

ZEUS AUTOMATION

OptiZeus SCADA

Operation Manual

Installation, configuration, screen building, plant operation, production and administration of the OptiZeus SCADA / HMI system

Software version 2.6.1.26

Edition 3.3 · 28 September 2026

Web optizeus.org

About this manual

This manual describes how to install, configure and operate **OptiZeus SCADA**: connecting it to PLCs and devices, defining tags, building synoptic screens, configuring alarms, historian, recipes, batches and reports, and administering users, security, backups and servers. It is the reference for the complete product; a plant-specific operator manual may be written from it for a particular installation.

Who should read it

Reader	Start with
System integrators and automation engineers who configure OptiZeus	Parts I–III, then the chapters for the features the plant uses
Supervisors and process engineers	Part IV (operation), Part V (recipes, batches, reports, KPIs)
Operators	Chapter 4 (the workspace), Chapter 16 (operating the plant), Chapter 17 (alarms)
IT and quality staff	Part VI (users, security, backups, servers, validation)

The manual assumes you know the basics of PLCs, industrial networks and process control, but not OptiZeus.

How the manual is organised

Part	Chapters	Contents
I Getting started	1–4	Architecture, installation and licensing, a complete first project, the workspace
II Configuration	5–9	Projects and deployment, PLC connections, tags and UDTs, the OPC UA server, scripts
III Screens	10–15	The synoptic builder, symbols, animations, super objects, layouts, advanced views
IV Operation	16–19	Operating the plant, alarms, events and audit trail, historian and trends
V Production	20–23	Recipes and batches, reports, KPI/OEE/SPC, operator tools
VI Administration	24–31	Users, security and Part 11, backups, servers, integration, diagnostics, troubleshooting, validation
Appendices	A–E	Keyboard, ports and files, licensing and limits, glossary, worked example projects

Conventions

Convention	Meaning
Bold	A menu item, tab, button or field exactly as OptiZeus labels it
Monospace	Tag names, addresses, file names, values you type
Configuration → Tags	A navigation path
Ctrl+S	A keyboard shortcut

Note: Information that helps you understand or use a feature.

Tip: A faster or better way to do something.

Important: Information you need to avoid a problem — lost configuration, a failed deployment, missing data.

WARNING: A risk to people, the plant or the product. Read it before acting.

Safety

WARNING: OptiZeus can write to PLCs. A setpoint, a command button, a recipe download or a script can start, stop or move equipment immediately. Before writing to a live plant, make sure the equipment is in a safe state and that the people near it know what you are about to do.

WARNING: A SCADA system is not a safety system. Emergency stops, interlocks and other safety functions must be implemented in hardware and in the PLC, independent of OptiZeus and of the network between them.

Important: Test new screens, scripts and alarm settings on a test project or a simulator before deploying them to a running plant.

Software version

This edition describes OptiZeus SCADA **version 2.6.1.26**. Screens and behaviour of other versions may differ. The version of a running system is shown in **Help** → **About OptiZeus** and returned by `GET /api/update/current-version`.

Related documents

Document	Subject
OptiZeus SCADA Technical Datasheet	Specifications
OptiZeus SCADA Cybersecurity Document	Security architecture and hardening
OptiZeus 21 CFR Part 11 Implementation Guide	Electronic records and signatures
GAMP 5 validation set (URS, FS, DS, risk assessment, IQ/OQ/PQ)	Computer system validation
Zeus PLC IDE Operation Manual	Programming the Zeus PLC controllers OptiZeus connects to

Revision history

Edition	Date	Change
2.x	April 2026	OptiZeus SCADA User Manual, version 2.5.4.47.
3.0	27 September 2026	Complete rewrite and expansion for version 2.6.1.23: quick-start tutorial, every connection type and dialog, UDTs, screens and animations reference, alarm pipelines and notifications, historian retention, ISA-88 batch, reports, KPI/OEE/SPC, administration, troubleshooting, validation, and worked example projects.
3.1	28 September 2026	Updated for 2.6.1.24 security release.
3.2	28 September 2026	Updated for 2.6.1.25: alarm delays and analog notifications, Clear All keeps history, faceplate/KPI/scheduler writes, retention 0 = keep, compression off, API error responses.
3.3	28 September 2026	Updated for 2.6.1.26: server-side weekly schedules, one write path for recipes/automation/pipelines/scripts, operator recipe download, shift report and alarm export, keyboard cancel, Modbus input addresses, connection password kept.

Contents

1. Product overview and architecture	14
1.1 What OptiZeus SCADA does	14
1.2 System architecture	14
1.3 Components	15
1.4 Supported protocols	16
1.5 How OptiZeus works with Zeus PLC controllers	17
1.6 Main functions at a glance	17
1.7 Licence editions	18
1.8 Roles	18
1.9 Where to go next	19
2. Installation, licensing and starting	20
2.1 Before you start	20
2.2 Installing on Windows	21
2.3 Running the server	24
2.4 The desktop application and kiosk stations	25
2.5 Licensing	27
2.6 Updating with a patch	28
2.7 HTTPS (SSL/TLS)	29
2.8 Installing on Linux	30
2.9 Uninstalling	31
2.10 Troubleshooting the installation	31
3. Quick start: your first project	33
3.1 The example	33
3.2 Step 1 — Create and activate the project	33
3.3 Step 2 — Add the PLC connection	34
3.4 Step 3 — Create the tags	34
3.5 Step 4 — Deploy the changes	35
3.6 Step 5 — Build the screen	35
3.7 Step 6 — Configure the alarms	36
3.8 Step 7 — Make the screen reachable	37
3.9 Step 8 — Operate the plant	37
3.10 Step 9 — View the trend	38
3.11 What you have built	38
4. The workspace: login, navigation, layouts and languages	
4.1 Opening OptiZeus	40
4.2 The login page	40
4.3 Screen elements after login	43
4.4 The header bar	43
4.5 Dashboard layouts	45
4.6 Languages and right-to-left display	47
4.7 The on-screen (touch) keyboard	48
4.8 Themes	48
4.9 Connection indications	49
4.10 Setup wizard, guided tours and help	50
4.11 Returning to the dashboard	51
5. Projects, databases, staging and deployment	53

5.1 What a project is	53
5.2 Where projects are stored	53
5.3 What lives in the project database	55
5.4 Managing projects	56
5.5 Staging and deployment	58
5.6 Server deployment (multi-server)	60
5.7 Worked example: taking a change into operation	62
6. PLC and device connections	63
6.1 Concepts	63
6.2 Supported protocols	63
6.3 Licence limits and permissions	64
6.4 The PLC Connections page	65
6.5 The connection dialog	66
6.6 Siemens S7	71
6.7 Modbus TCP	72
6.8 EtherNet/IP (Logix symbolic tags)	73
6.9 PCCC over EtherNet/IP (MicroLogix, SLC 500)	75
6.10 OPC UA client	76
6.11 Reading and writing: common behaviour	78
6.12 MQTT (client connection)	79
6.13 BACnet/IP	80
6.14 GE SRTP	81
6.15 Status, alarms and diagnostics	81
6.16 MQTT Sparkplug B publishing	82
6.17 Worked example: a Modbus TCP skid and an S7-1200	83
7. Tags, UDTs and instances	85
7.1 Concepts	85
7.2 Permissions and licence limits	85
7.3 Naming rules	86
7.4 The Tags tab	86
7.5 The tag form	87
7.6 Simulation tags	91
7.7 Calculated tags	92
7.8 UDTs and UDT instances	93
7.9 Browsing and monitoring tags	95
7.10 CSV import and export	96
7.11 REST access	97
7.12 Worked example: pump UDT with 3 instances	97
8. OPC UA server	99
8.1 Overview	99
8.2 Permissions	99
8.3 The OPC UA Server page	100
8.4 Address space	101
8.5 Security and certificates	102
8.6 Procedure: publish tags to an MES	102
8.7 Limits and known issues in 2.6.1.26	103
8.8 REST endpoints	104
9. Scripts, expressions and automation actions	105
9.1 Choosing the right tool	105
9.2 Event scripts	105
9.3 Expressions	109

9.4 Automation actions	111
9.5 Weekly scheduler	113
9.6 REST endpoints	114
10. The synoptic builder	117
10.1 Opening the builder	117
10.2 The synoptic screen list	118
10.3 The builder window	118
10.4 The canvas	120
10.5 Adding objects	121
10.6 Selecting and arranging objects	122
10.7 Screen properties	125
10.8 Saving and publishing	126
10.9 Popup screens	127
10.10 Importing drawings: CAD and SVG	128
10.11 Worked example: a tank overview screen	128
10.12 Limits and good practice	129
11. Symbols, shapes and the SVG library	131
11.1 Overview of object sources	131
11.2 The Object Library	131
11.3 Common properties (Props tab)	135
11.4 Appearance properties by object kind	136
11.5 The SVG library	138
11.6 Symbols from other HMI/SCADA projects	139
11.7 Worked example: a pump symbol with state colours	139
12. Animations and dynamic properties	141
12.1 Concepts	141
12.2 Adding an animation	141
12.3 Animation types	142
12.4 Summary of runtime support	147
12.5 Tag bindings list	148
12.6 Click actions (Action tab)	148
12.7 Object popups (Popup tab)	151
12.8 Expressions	152
12.9 Event scripts and attributes	153
12.10 Worked examples	154
12.11 Troubleshooting animations	155
13. Super objects, faceplates and templates	157
13.1 Concepts	157
13.2 The super object list	157
13.3 The super object builder	158
13.4 Placing super objects on screens	160
13.5 Variants	161
13.6 Super object instances	162
13.7 Faceplates	162
13.8 Worked example: a pump super object bound to a UDT	164
14. Layouts, viewers and navigation	165
14.1 How the display is organised	165
14.2 Display settings	165
14.3 The Layout Builder	166
14.4 Screen navigation	170

14.5 Station layouts and user layouts	171
14.6 Header navigation buttons	172
14.7 Pop-out windows and multiple monitors	172
14.8 Kiosk and full-screen operation	172
14.9 Data widgets on synoptic screens	173
14.10 Worked example: an operator station for Line 1	174
15. Digital twin, live video and advanced views	176
15.1 Digital Twin	176
15.2 Live video (IP cameras)	178
15.3 Screen Replay	179
15.4 Screen recording	180
15.5 Process Viewer and Process Status Badge	181
15.6 Server Topology	181
15.7 Pop-out screens and multi-monitor operation	182
15.8 Choosing the right view	182
16. Operating the plant: screens, controls and faceplates	
16.1 The operator's workspace	184
16.2 Reading a synoptic screen	185
16.3 Operating objects on a screen	186
16.4 Writing a setpoint or value	187
16.5 The touch keyboard	189
16.6 Faceplates	190
16.7 The Equipment Control panel	190
16.8 Roles and read-only operation	191
16.9 When communication is lost	191
16.10 Safe operating practice	191
16.11 Worked example: changing a tank level setpoint	192
17. Alarms	193
17.1 How alarms work in OptiZeus	193
17.2 The Alarms configuration tab	194
17.3 Severity and priority	198
17.4 Alarm states	198
17.5 The alarm panel and the alarm views	199
17.6 Acknowledging alarms	201
17.7 Shelving	202
17.8 Smart Alarm Grouping and flood detection	203
17.9 Global display settings	203
17.10 Notifications	203
17.11 Alarm pipelines	205
17.12 Alarms from external systems and Zeus PLC controllers	206
17.13 Alarm history	207
17.14 Worked example: Tank_101 level alarms	207
18. Events, sequence of events and the audit trail	
18.1 The three records compared	209
18.2 The event log	209
18.3 Sequence of events (SOE)	211
18.4 The audit trail	212
18.5 Retention	214
18.6 Worked example: a monthly audit review	214
19. Historian and trends	216

19.1 Architecture	216
19.2 Configuring the historian	217
19.3 Recording methods, deadband and compression	218
19.4 Storage tiers and retention	219
19.5 The Historian view	220
19.6 Period comparison	221
19.7 Correlation analysis	222
19.8 Export and printing	222
19.9 The Trends view and trend charts on screens	222
19.10 Performance notes	223
19.11 Worked example: historizing Tank_101 and reviewing a shift	224
20. Recipes and batch control (ISA-88)	226
20.1 Which tool to use	226
20.2 Recipe Manager overview	227
20.3 Creating and editing a parameter recipe	228
20.4 Download and upload	230
20.5 Recipes on screens	231
20.6 Recording the recipe in a batch record	232
20.7 Worked example: a 5-parameter mixing recipe	233
20.8 Batch Control (S88) overview	234
20.9 Configuring Batch Control	235
20.10 Running a batch (operator procedure)	238
20.11 Batch history and batch records	239
20.12 REST API summary	240
21. Reports	242
21.1 Report tools at a glance	242
21.2 Report Designer (template reports)	243
21.3 Visual Report Builder	244
21.4 Batch reports end to end	247
21.5 Shift Handover Report	247
21.6 Scheduled (automatic) reports	248
21.7 Exporting and printing	249
21.8 Worked example: a daily shift report	250
22. KPI, OEE and SPC dashboards	251
22.1 KPI Dashboard	251
22.2 OEE & Downtime Tracking	253
22.3 SPC components	255
22.4 Operator dashboard components	257
22.5 Data logging for dashboards and reports	257
22.6 Worked example: OEE for a filling machine	257
23. Operator tools: procedures, notepad, voice, AI assistant and anomaly detection	
23.1 Procedure Automation (electronic SOPs)	259
23.2 Operator Notepad	261
23.3 Voice Commands	262
23.4 AI Assistant	263
23.5 AI Anomaly Detection	264
23.6 Guided Tours	265
23.7 REST API summary	266
24. Users, roles, MFA and directory integration	268
24.1 Concepts	268

24.2 The default accounts	268
24.3 Roles and permissions	269
24.4 Security areas	272
24.5 Managing user accounts	272
24.6 Password policy and lockout	274
24.7 Sessions, time-outs and logout	276
24.8 Two-factor authentication (TOTP)	276
24.9 Single sign-on with OpenID Connect	277
24.10 Active Directory / LDAP	278
24.11 Connected clients and force disconnect	279
24.12 Worked example — access plan for a small plant	280
25. Security, electronic signatures and 21 CFR Part 11	
25.1 Security architecture at a glance	281
25.2 Installation secrets	281
25.3 HTTPS and secure WebSocket	282
25.4 Hardening checklist	282
25.5 Requiring a reason and re-authentication for writes	283
25.6 Electronic signatures	284
25.7 The audit trail	286
25.8 Verifying audit-trail integrity	286
25.9 Mapping to 21 CFR Part 11	287
25.10 Worked example — configuring a GMP mixing line	288
26. Backup, restore and maintenance	290
26.1 What must be protected	290
26.2 The Backup & Restore page	290
26.3 Scheduled (automatic) backups	291
26.4 Manual backup and restore	292
26.5 Historian archive backup	294
26.6 Service control	294
26.7 Database maintenance	295
26.8 Log files and rotation	296
26.9 Installing a patch or upgrade	296
26.10 The many-databases trap	298
26.11 Recommended maintenance schedule	298
27. Servers, multi-site, redundancy and store-and-forward	
27.1 Overview and licensing	299
27.2 Multi-server deployment	299
27.3 Server redundancy and failover	301
27.4 Multi-site dashboard	303
27.5 Store & forward	303
27.6 MQTT Sparkplug B publishing	304
27.7 Edge gateway	305
27.8 Offline mode (browser client)	305
27.9 Auto-update	306
27.10 Choosing an architecture	306
28. Integration: REST API, webhooks, MQTT, ERP	307
28.1 Integration options at a glance	307
28.2 REST API basics	307
28.3 Live data	309
28.4 Exporting history, alarms and events	310
28.5 Webhooks	312

28.6 MQTT	314
28.7 SAP / ERP connector	314
28.8 OPC UA server	315
29. System logs, performance and diagnostics	316
29.1 Where to look first	316
29.2 System Logs — Event Logs	316
29.3 System Logs — Console Logs	318
29.4 Version information	318
29.5 Health and diagnostic endpoints	319
29.6 Performance monitor	320
29.7 Start-up messages	321
29.8 Background workers and communication diagnostics	322
29.9 Collecting data for a support case	323
30. Troubleshooting	324
30.1 Values show XXXX , --- or tags are red	324
30.2 PLC shows disconnected	325
30.3 Login problems	326
30.4 Slow screens or configuration pages	327
30.5 Reports and history	327
30.6 Service does not start or keeps restarting	328
30.7 Port conflicts and firewall	329
30.8 Licence problems	329
30.9 Backup, maintenance and integration	330
30.10 When to contact support	331
31. Validation (GAMP 5)	332
31.1 GAMP 5 classification	332
31.2 Validation approach	332
31.3 The supplied validation document set	333
31.4 Installation Qualification (IQ) — outline	334
31.5 Operational Qualification (OQ) — outline	335
31.6 Performance Qualification (PQ) — outline	336
31.7 What the product provides — and gaps to cover by procedure	336
31.8 Maintaining the validated state	337
31.9 Worked example — validation deliverables for a small GMP line	337
Appendix A. Keyboard shortcuts and touch operation	
A.1 Conventions	340
A.2 Synoptic screen editor (Viewer Builder)	340
A.3 Super object and faceplate builders	341
A.4 Other editors	342
A.5 Runtime screens and dialogs	343
A.6 Touch-screen operation	345
A.7 On-screen keyboards	345
A.8 Kiosk operation	347
A.9 Quick reference card	350
Appendix B. Ports, services and files	351
B.1 Listening ports	351
B.2 Outbound connections	352
B.3 The Windows service	353
B.4 Folder structure	354
B.5 Configuration files	356

B.6 Databases	358
B.7 Logs	360
B.8 Checklist for a new server	360
Appendix C. Licence tiers, add-ons and limits	361
C.1 Licence editions	361
C.2 Add-on modules	362
C.3 Trial mode	363
C.4 Hardware ID	363
C.5 Licence files	364
C.6 Activating, moving and renewing a licence	364
C.7 Technical limits	366
Appendix D. Glossary	371
D.1 Terms	371
D.2 Abbreviations	375
Appendix E. Worked example projects	378
E.0 Common procedures	378
E.1 Tank farm level monitoring with alarms	378
E.2 Pump station with duty/standby and run-hour reporting	382
E.3 Batch mixer with recipe and batch report	385
E.4 OEE dashboard for a packaging line	390
E.5 Connecting a Zeus PLC controller over OPC UA and EtherNet/IP	392
E.6 Alarm notification by e-mail and WhatsApp, with escalation	395

OPTIZEUS SCADA · OPERATION MANUAL

Part I — Getting started

1. Product overview and architecture

This chapter explains what OptiZeus SCADA is, which software components make up an installation, how they communicate, and which features each licence edition includes. Read it before you plan an installation. System integrators and engineers need all of it; supervisors and operators can read 1.1 and 1.6 and skip the rest.

1.1 What OptiZeus SCADA does

OptiZeus SCADA is a supervisory control and data acquisition (SCADA) and human-machine interface (HMI) system. One OptiZeus server:

- reads and writes process values in PLCs and other devices over industrial protocols (OPC UA, Modbus TCP, Siemens S7, EtherNet/IP, PCCC, MQTT, BACnet/IP, GE SRTP);
- keeps a live image of every configured **tag** (process variable) in memory and pushes changes to all connected clients in real time;
- evaluates alarm conditions, keeps the alarm and event history and sends notifications;
- records tag values in a historian and produces trends and reports;
- serves the operator and engineering user interface to web browsers and to the OptiZeus desktop application.

All configuration — PLC connections, tags, screens, alarms, recipes, users — is done in the same user interface, in the **Config** view. There is no separate engineering tool to install.

Version covered by this manual: **2.6.1.26** (build date 2026-09-28). The server reports its version at `GET /api/update/current-version`.

1.2 System architecture



The server is a single Node.js process. Clients hold no process data of their own: every value on a screen comes from the server over a WebSocket connection, and every operator command goes back to the server, which writes it to the PLC.

1.3 Components

1.3.1 The OptiZeus server (backend)

Item	Value
Program	<code>backend\server.js</code> , run by Node.js 24
Default install folder	<code>C:\OptiZeus\backend</code> (Windows), <code>/opt/optizeus/backend</code> (Linux)
Windows service	<code>OptiZeus</code> , managed by the WinSW wrapper <code>backend\daemon\optimizeus.exe</code> (Chapter 2)
Linux service	systemd unit <code>optimizeus</code>
HTTP port	3004 (environment variable <code>PORT</code>)
HTTPS port	3443 (environment variable <code>HTTPS_PORT</code>); only opened when <code>ssl-key.pem</code> and <code>ssl-cert.pem</code> exist in the backend folder
Health check	<code>GET /api/health</code> returns <code>{"status":"ok"}</code> without logging in

The server contains the REST API (several hundred routes, see Chapter 28), the WebSocket server that pushes tag values, alarms and events, the alarm engine, the staging and deployment service, the report scheduler, the notification services (e-mail, SMS, WhatsApp, Telegram, voice), the optional OPC UA server and the static web server that delivers the user interface.

Note: Some customer installations run the server from the copy inside the desktop application folder, `C:\OptiZeus\electron\resources\backend\`, instead of `C:\OptiZeus\backend\`. To see which copy is running, open `C:\OptiZeus\backend\daemon\optimizeus.xml` and read the `<argument>` line. Patches must reach the copy that runs (Chapter 2).

1.3.2 The web client

The user interface is a single-page web application that the server delivers from `backend\public` (fallback `backend\frontend`). Any workstation with a current Chromium-based browser or Firefox opens it at:

- `http://<server>:3004`, or
- `https://<server>:3443` when HTTPS is configured.

Nothing has to be installed on a browser workstation. The client installer (Chapter 2) only adds a launcher page and desktop shortcuts.

1.3.3 The desktop application (Electron)

`C:\OptiZeus\electron\OptiZeus SCADA.exe` is a dedicated window around the same web client. Compared with a browser it adds:

- kiosk mode (full screen, no window frame, exit only by a key combination and optional password);
- desktop lock (blocks the Windows key, Alt+Tab and the taskbar);

- a system-tray icon, auto-start with Windows and a splash screen;
- automatic recovery: if the page crashes or fails to load it reloads, and the window is kept awake for unattended stations.

On start the desktop application checks `http://localhost:3004/api/health`. If no server answers, it starts its own server from `electron\resources\backend`. It always clears the stored login on start, so every session begins at the login page. Settings are described in Chapter 2 and Chapter 4.

1.3.4 Worker threads

Two jobs run in their own threads so that they never delay the user interface:

Worker	Job
EtherNet/IP worker (<code>workers\eipWorker.js</code>)	Polls Logix / EtherNet/IP PLCs. Each PLC gets its own independent polling loop, and results are sent back to the main thread as blocks of values with the measured cycle time.
Historian worker (<code>workers\historianWorker.js</code>)	Receives tag values from the main thread, applies the recording method (interval, on change, or both), deadband and swinging-door compression, buffers the values and writes them to the binary archive in the project's <code>historian-archives</code> folder (17 bytes per sample: time, value and quality). It flushes every 5 seconds by default.

1.3.5 Databases and project folders

Each **project** is a folder with its own SQLite database `scada.db` (WAL journal mode, so you also see `scada.db-wal` and `scada.db-shm`) and a set of JSON configuration files. Only one project is active at a time. The **Default** project is the backend folder itself; other projects are in `backend\projects\<name>`. Chapter 5 explains projects in detail.

Note: The SVG symbol library (Chapter 11) is always stored in the database of the Default project, `backend\scada.db`, and is shared by all projects.

1.4 Supported protocols

Protocol	UI name	Default port	Used for
OPC UA (client)	OPC UA	4840	Any OPC UA server; node browsing; optional OPC UA Alarms & Conditions subscription
Modbus TCP	Modbus TCP	502	Modbus devices and PLCs over Ethernet (unit ID 1–247)
Siemens S7 (ISO-on-TCP)	Siemens S7 (ISO-on-TCP)	102	S7-300/400/1200/1500 with rack and slot
EtherNet/IP (CIP)	EtherNet/IP (CIP)	44818	Allen-Bradley Logix controllers, tag browsing, 1–6 parallel connections
PCCC over EtherNet/IP	PCCC over EtherNet/IP	44818	Allen-Bradley MicroLogix and SLC 500 (addresses such as <code>N7:0</code> , <code>F8:0</code> , <code>B3:0/5</code> , <code>T4:0.ACC</code>)
MQTT	MQTT	1883 (8883 with TLS)	Topic subscription; JSON payloads with <code>topic::jsonPath</code>

Protocol	UI name	Default port	Used for
BACnet/IP	BACnet/IP	47808	Building automation objects (<code>analogInput:0</code> , <code>bv:3:presentValue</code>)
GE SRTP	GE SRTP (SNP-X)	18245	GE PACSystems and Series 90 (<code>%R</code> , <code>%AI</code> , <code>%AQ</code> , <code>%I</code> , <code>%Q</code> , <code>%M</code>)

The server can also act as a data source for other systems: OPC UA server (Chapter 8), MQTT Sparkplug B publisher (Chapter 28), REST API, webhooks and SQL database logging.

Important: The PLC brand list also offers models whose only protocol is Mitsubishi MELSEC, Omron FINS or Beckhoff ADS. Version 2.6.1.26 has no driver for these three protocols; a connection that uses them fails with "Unsupported server type". Connect such PLCs through OPC UA or Modbus TCP. Modbus RTU and other serial connections are not supported in this version.

Protocol settings and addressing are covered in Chapter 6.

1.5 How OptiZeus works with Zeus PLC controllers

OptiZeus has no dedicated driver for Zeus PLC controllers in version 2.6.1.26. A Zeus PLC is connected like any other controller, through a standard protocol that the controller exposes (for example OPC UA or Modbus TCP), with **Generic / Other** as the brand in the connection dialog. Map the controller variables to OptiZeus tags as described in Chapter 7. Check which protocols your controller firmware has enabled before you design the tag list.

1.6 Main functions at a glance

Area	Functions	Chapter
Configuration	Projects, PLC connections, tags and folders, UDTs and instances, calculated / memory / simulated tags, staging and deployment	5–9
Screens	Synoptic builder with a large P&ID object library, SVG library, animations, click actions, super objects, faceplates, popups, layouts	10–15
Operation	Dashboard, synoptic screens, operator controls, faceplates, touch keyboard	16
Alarms	Analog (HH/H/L/LL), Boolean, bit and communication alarms, groups and areas, acknowledgment with reasons, popups and sounds, notification channels, pipelines	17
History	Event log, sequence of events, audit trail, historian with raw and aggregated retrieval, trend templates, screen replay	18–19
Production	Recipes, ISA-88 batch control (add-on), reports and report designer, shift reports, KPI, OEE, SPC	20–22
Operator tools	Procedures, notepad, voice commands, AI assistant, anomaly detection	23
Administration	Users and roles, MFA, directory login, electronic signatures, backup, multi-server deployment, redundancy, store-and-forward, integration, logs	24–29

1.7 Licence editions

A licence has one edition and may carry add-on modules. The limits are defined in the server file `backend\license.js`.

	Starter	Basic	Professional	Professional Max	Enterprise
Tags	100	1,000	5,000	10,000	20,000
Concurrent clients	1	2	5	5	10
PLC connections	1	3	8	8	20
External data sources	1	3	10	10	12
Screens	5	10	50	50	unlimited
Users	2	5	20	20	unlimited
Features	dashboard, alarms, events, synoptic	+ trends, datalog	+ reports, recipes	+ extended notifications (SMS, WhatsApp, Telegram)	+ SQL Server, API, custom drivers, redundancy, multi-server

Add-on	Name
<code>batch-control</code>	Batch Control (S88)
<code>spc</code>	SPC / Statistical Process Control
<code>ai-assistant</code>	AI Assistant
<code>opcua-server</code>	OPC UA Server

Note: In version 2.6.1.26 the server enforces the client, PLC-connection and tag limits, the Batch Control add-on and the multi-server feature. The other limits and flags are recorded in the licence but not yet checked. Appendix C lists exactly what is enforced. A new installation runs a 7-day trial with Starter limits (Chapter 2).

1.8 Roles

OptiZeus has three built-in roles. Roles can be extended with security letters and kiosk mode (Chapter 24).

Role	Typical user	Can do
<code>admin</code>	Engineer, system administrator	Everything, including Config , deployment and user management
<code>operator</code>	Plant operator	View screens, acknowledge alarms, operate controls and write tag values
<code>viewer</code>	Manager, visitor	View only

1.9 Where to go next

- To install the server, go to Chapter 2.
- To build a first working project in about an hour, follow Chapter 3.
- To learn the screen layout, header and navigation, read Chapter 4.

2. Installation, licensing and starting

This chapter describes how to install the OptiZeus server and client workstations on Windows and Linux, how to run the server as a service, how to set up HTTPS, how to activate a licence, how to update an installation with a patch and how to remove it. It is written for the engineer or IT administrator who installs the system. Operators do not need it.

2.1 Before you start

2.1.1 Server requirements

Item	Requirement
Operating system	Windows 10/11 (64-bit) or Windows Server; or a Debian/Ubuntu Linux with systemd and <code>apt</code> (the Linux setup uses the NodeSource <code>apt</code> repository)
Node.js	Version 24. The Windows release contains <code>node-v24.12.0-x64.msi</code> and installs it when needed. The bundled <code>node_modules</code> are built for Node 24; an older Node.js is replaced.
Rights	Local administrator (Windows) or root (Linux) for installation
Disk	Install folder <code>C:\OptiZeus</code> (fixed; the setup does not ask for a folder) plus room for the historian archive, backups and logs
Network	Reachable from every client on TCP 3004 and 3443; able to reach every PLC on its protocol port (Chapter 6)
Clock	Synchronised with a time server — alarm, event and historian timestamps come from the server clock

The following sizing figures come from the previous manual edition and have not been re-measured for 2.6.1.26. Treat them as a starting point.

	Minimum	Recommended
CPU	2 cores	4 or more cores
RAM	4 GB	8 GB or more (16 GB above 10,000 tags)
Disk	10 GB	50 GB SSD or more

2.1.2 Client requirements

Client	Requirement
Browser	Current Chrome, Edge or Firefox. JavaScript and WebSockets must not be blocked by a proxy.
Desktop application	Windows 10/11 64-bit; about 2 GB free RAM for the embedded browser engine
Display	Screens are designed in pixels; 1920 × 1080 is a practical minimum for engineering work

2.1.3 Ports

Port	Direction	Purpose
TCP 3004	Clients → server	HTTP and WebSocket (<code>ws://</code>)
TCP 3443	Clients → server	HTTPS and secure WebSocket (<code>wss://</code>), only when certificates exist
TCP 4840	OPC UA clients → server	OptiZeus OPC UA server, only when enabled (Chapter 8)
PLC ports	Server → devices	4840 OPC UA, 502 Modbus TCP, 102 S7, 44818 EtherNet/IP and PCCC, 1883/8883 MQTT, 47808/UDP BACnet, 18245 SRTP

Important: The installers do not create Windows Firewall rules. Open the ports yourself on the server (see 2.2.4). Without the rules the server works on the server PC itself but no other workstation can connect.

2.1.4 What is in the release

The full release is a ZIP named `OptiZeus-SCADA-v2.6.1.26-full.zip`. Extract it to a local folder (not to `C:\OptiZeus` itself) before you run anything.

Item	Content
<code>SETUP.bat</code>	Windows installer, server and client modes
<code>INSTALL-PATCH.bat</code>	Windows updater (see 2.6)
<code>UNINSTALL.bat</code>	Windows uninstaller
<code>setup.sh</code> , <code>install-patch.sh</code> , <code>uninstall.sh</code>	The Linux equivalents
<code>node-v24.12.0-x64.msi</code>	Offline Node.js installer
<code>backend\</code>	Server program, drivers, services, workers, the WinSW service wrapper in <code>backend\daemon\</code> , and the pre-built web client
<code>electron\</code>	Desktop application (<code>OptiZeus SCADA.exe</code>)
<code>client\optizeus-launcher.html</code>	Launcher page used by the desktop shortcuts
<code>scripts\</code>	Service helper scripts
<code>docs\</code>	Documentation

2.2 Installing on Windows

2.2.1 Server installation

To install a server:

1. Log on to the server PC as a local administrator.
2. Extract the release ZIP, for example to `C:\Install\OptiZeus-2.6.1.26`.

3. Right-click `SETUP.bat` and choose **Run as administrator**. If you start it without administrator rights it stops with "Administrator privileges required".
4. At **Select installation mode**, type **1 (SERVER + CLIENT)** and press Enter.
5. Wait while the setup runs its ten steps (table below). It writes a log to `%TEMP%\optizeus-setup.log` and `setup.log` beside the batch file, and copies it to `C:\OptiZeus\install.log` at the end.
6. At the end the setup waits 8 seconds and checks `https://localhost:3443/api/health`, then `http://localhost:3004/api/health`. "Server is running" means the installation succeeded.
7. Open the firewall ports (2.2.4).
8. Open `https://localhost:3443` (or `http://localhost:3004`) on the server, accept the licence agreement and activate the licence or start the trial (2.5).
9. Log in with the default administrator account. The login asks you to replace the factory default password before you continue (**Change the Default Password**, Chapter 4).

Step	What the setup does
1/10 Node.js	Uses the installed Node.js if its major version is 24 or higher. Otherwise installs the bundled MSI silently; without the MSI it tries to download Node.js 24 from nodejs.org. If neither works it stops.
2/10 Stop	Stops the services <code>OptiZeus</code> and <code>OptiZeusBackend</code> and ends every <code>node.exe</code> process on the PC.
3/10 Folders	Creates <code>C:\OptiZeus\backend</code> and its sub-folders (<code>services</code> , <code>drivers</code> , <code>middleware</code> , <code>utils</code> , <code>db</code> , <code>scripts</code> , <code>projects</code> , <code>videos</code> , <code>public</code> , <code>frontend</code> , <code>daemon</code>).
4/10 Backend	Copies the backend files. <code>node_modules</code> is copied only on a fresh install; an existing <code>node_modules</code> is kept. Copies <code>UNINSTALL.bat</code> to <code>C:\OptiZeus</code> .
5/10 Web client	Deletes the old <code>public\static</code> and <code>frontend\static</code> folders and copies the new web client into both.
6/10 Dependencies	If <code>node_modules</code> is missing, runs <code>npm ci --production</code> (needs Internet access).
7/10 HTTPS	If <code>ssl-key.pem</code> does not exist, creates a self-signed certificate for <code>localhost</code> / <code>127.0.0.1</code> , valid 10 years, with OpenSSL or, if OpenSSL is missing, with Node.js. Adds the certificate to the Windows trusted root store of the server PC.
8/10 Service	Removes any old <code>OptiZeus</code> service, writes <code>backend\daemon\optizeus.xml</code> and registers the service with WinSW. If that fails, creates a scheduled task OptiZeus SCADA that starts the server at boot as SYSTEM.
9/10 Shortcuts	Installs the launcher page in <code>C:\OptiZeus\client</code> , copies the desktop application to <code>C:\OptiZeus\electron</code> and creates the desktop shortcuts OptiZeus SCADA , OptiZeus SCADA (App) and Uninstall OptiZeus SCADA for the user who ran the setup.
10/10 Start	Starts the service (or the scheduled task, or, as a last resort, a console window running <code>node server.js</code>).

WARNING: Step 2 ends every Node.js process on the computer, not only OptiZeus. Do not run the setup on a PC where other Node.js applications must keep running, and never during production on a running plant server without a planned stop.

Note: The setup banner of the 2.6.1.26 release still reads "v2.6.1.8". This is only a label; the files installed are those of the release.

2.2.2 Client workstation installation

A client workstation needs no server components. To install one:

1. Extract the release ZIP on the workstation.
2. Run `SETUP.bat` as administrator and type `2 (CLIENT ONLY)`.
3. Enter the server addresses when asked:

Prompt	Example	Notes
Primary Server HTTPS	<code>https://192.168.1.10:3443</code>	
Primary Server HTTP	<code>http://192.168.1.10:3004</code>	
Backup Server HTTPS	(empty)	Only with a redundant server pair
Backup Server HTTP	(empty)	

4. The setup writes the addresses into `C:\OptiZeus\client\optizeus-launcher.html`, copies the desktop application to `C:\OptiZeus\electron` and creates the shortcuts **OptiZeus SCADA** (launcher) and **OptiZeus SCADA (App)**.

The launcher tries the servers in this order and opens the first one that answers: primary HTTPS, backup HTTPS, primary HTTP, backup HTTP.

Note: The desktop application on a client-only workstation connects to `http://localhost:3004`. If no server runs locally it tries to start the server copy inside `electron\resources\backend`. On a pure client workstation use the **OptiZeus SCADA** launcher shortcut or a browser, or point the application at the server with the **Server Settings** link on the login page (Chapter 4).

2.2.3 Installed folders and files (Windows)

Path	Content
<code>C:\OptiZeus\backend\</code>	Server; also the Default project (its <code>scada.db</code> and JSON files)
<code>C:\OptiZeus\backend\projects\<name>\</code>	Other projects (Chapter 5)
<code>C:\OptiZeus\backend\daemon\</code>	<code>optizeus.exe</code> (WinSW), <code>optizeus.xml</code> , service logs <code>optizeus.out.log</code> , <code>optizeus.err.log</code> , <code>optizeus.wrapper.log</code> (rotated)
<code>C:\OptiZeus\backend\ssl-key.pem</code> , <code>ssl-cert.pem</code>	HTTPS certificate
<code>C:\OptiZeus\backend\.jwt-secret</code> , <code>.hmac-secret</code>	Per-installation secrets for login tokens and record integrity. Created on first start. Never copy them to another site.
<code>C:\OptiZeus\license.dat</code> , <code>agreement.dat</code> , <code>grace-period.dat</code>	Licence state (one level above the backend folder that runs)

Path	Content
C:\OptiZeus\electron\	Desktop application
C:\OptiZeus\client\	Launcher page
C:\OptiZeus\install.log	Setup log

2.2.4 Opening the firewall

To allow clients to connect, run these commands in an administrator command prompt on the server. The rule names match the names that `UNINSTALL.bat` removes.

```
netsh advfirewall firewall add rule name="OptiZeus HTTP" dir=in action=allow protocol=TCP localport=3004
netsh advfirewall firewall add rule name="OptiZeus HTTPS" dir=in action=allow protocol=TCP
localport=3443
```

If you enable the OptiZeus OPC UA server (Chapter 8), also open its port:

```
netsh advfirewall firewall add rule name="OptiZeus SCADA" dir=in action=allow protocol=TCP
localport=4840
```

Restrict the rules to the plant network (`remoteip=` parameter) according to your site security policy (Chapter 25).

2.3 Running the server

2.3.1 As a Windows service (normal operation)

The setup registers the server as the Windows service **OptiZeus** with the WinSW wrapper. The generated `optizeus.xml` :

```
<service>
  <id>OptiZeus</id>
  <name>OptiZeus</name>
  <description>OptiZeus Industrial Automation System</description>
  <executable>C:\Program Files\nodejs\node.exe</executable>
  <argument>C:\OptiZeus\backend\server.js</argument>
  <logmode>rotate</logmode>
  <stoptimeout>30sec</stoptimeout>
  <onfailure action="restart" delay="5sec"/>
  <onfailure action="restart" delay="10sec"/>
  <onfailure action="restart" delay="30sec"/>
  <env name="NODE_ENV" value="production"/>
  <workingdirectory>C:\OptiZeus\backend</workingdirectory>
</service>
```

The service starts automatically at boot and restarts after a crash (after 5, 10 and 30 seconds).

Task	Command (administrator prompt)
Start	<code>net start OptiZeus</code>
Stop	<code>net stop OptiZeus</code>

Task	Command (administrator prompt)
Restart	<code>net stop OptiZeus & net start OptiZeus</code>
Status	<code>sc query OptiZeus (STATE 4 = RUNNING)</code>

The same functions are in the user interface under **Config** → **Admin** → **Startup Mode** → **Server Startup Mode**:

Element	Meaning
Windows Service Status	Service Running, Service Stopped or not installed, with Refresh Status
Service Registration	Register Service / Unregister Service, Start Service, Stop Service, Restart Service
Startup Mode Preference	Normal Mode (standard desktop process) or Service Mode
Service Configuration	Auto-start on system boot, Restart on failure, Start Delay (seconds) (0–300), Max Restart Attempts (1–10); saved with Save Startup Configuration to <code>backend\service-config.json</code>

Important: Register Service in the user interface, and `backend\scripts\install-service.bat`, register the service with a different wrapper (node-windows) than the setup (WinSW). Use them only if the setup could not register the service. Do not mix the two methods on one PC; to re-register, run `SETUP.bat` or `INSTALL-PATCH.bat` again. Stopping the service from the user interface also stops the server that serves the page — you lose the connection until the service is started again.

2.3.2 As a scheduled task (fallback)

If WinSW registration fails, the setup creates the scheduled task **OptiZeus SCADA** (at startup, as SYSTEM, highest privileges, restart 3 times at 1-minute intervals). Start it with `schtasks /run /tn "OptiZeus SCADA"`.

2.3.3 Manually from a console

For troubleshooting, stop the service and run the server in a console:

```
cd /d C:\OptiZeus\backend
node server.js
```

The console shows every log line with a local timestamp. The server reports `OptiZeus Server running on http://localhost:3004` and, with certificates, `OptiZeus HTTPS Server running on https://localhost:3443`. Close the console with Ctrl+C and start the service again.

2.4 The desktop application and kiosk stations

2.4.1 Starting the desktop application

Double-click **OptiZeus SCADA (App)**. The application shows a splash screen, checks that the server answers on port 3004 (starting its own server copy if necessary, see 1.3.3), loads the user interface, clears any stored login and shows the login page.

2.4.2 Desktop application settings

Settings are edited in **Config** → **Admin** → **Clients & Shutdown** → **Electron Client** and saved with **Save Configuration**. They take effect when the desktop application is restarted.

Section	Field	Meaning	Default
Window Settings	Width (px), Height (px)	Window size; Quick Presets fill common resolutions	1600 × 1000
	Min Width (px), Min Height (px)	Smallest window size	1024 × 768
	Start in Fullscreen	Full screen at start	Off
	Start Maximized (recommended if not using fullscreen)		On
Kiosk Mode	Enable Kiosk Mode	Full screen, no frame, no menu	Off
	Exit Password	Password that the exit dialog asks for	Empty
	Exit Key Combination	Key combination that closes the kiosk	Ctrl+Shift+Q
Menu Bar	Show Menu Bar, Show DevTools Option		On, Off
System Tray	Enable System Tray Icon, Minimize to Tray on Close, Show Notification Balloons		On, On, On
Startup	Auto-start with Windows	Adds the application to the Windows Run key of the current user	Off
	Splash Screen Duration (ms)	0–10000	1500
	Default Screen	Dashboard, Alarms, Trends or Synoptic	Dashboard
Security	Allow Opening External Links		Off
	Disable Right-Click Context Menu		Off
	Disable Browser Keyboard Shortcuts (F5, Ctrl+R, etc.)		Off
	Prevent Application Close (blocks close button, Alt+F4)		Off
	Lock Desktop (blocks Win key, Alt+Tab, taskbar access)	Hides the taskbar and disables the Windows key while the application runs; restored on exit	Off
Interface Settings	Enable Touch Screen Mode (shows on-screen keyboard for value entry)	See Chapter 4	Off

Note: The settings are stored in `electron-config.json` in the folder of the active project, but the desktop application reads `electron-config.json` from `electron\resources\backend\`. On an installation where the service runs `C:\OptiZeus\backend` (or where a project other than Default is active), copy the saved file to `C:\OptiZeus\electron\resources\backend\electron-config.json` so that the application uses it.

2.4.3 Setting up a kiosk station

To set up an operator station that cannot be left:

1. Create an operator account whose role has kiosk mode enabled (Chapter 24).
2. In **Electron Client**, tick **Enable Kiosk Mode**, set an **Exit Password** and, if required, tick **Lock Desktop**, **Disable Browser Keyboard Shortcuts** and **Prevent Application Close**.
3. Tick **Auto-start with Windows**, click **Save Configuration** and copy the file as described in the note above.
4. Restart the desktop application and check that the exit key combination asks for the password.

In a browser, a user whose role has kiosk mode must work full screen. If the browser leaves full screen, a **KIOSK MODE** overlay covers the page until the user clicks **Enter Fullscreen**.

2.5 Licensing

Licensing is described fully in Appendix C. This section gives the procedures an installer needs.

2.5.1 First start: agreement, trial or activation

When no licence or trial exists, the browser shows the licence dialog before the login page.

1. **License Agreement**: scroll to the end of the text (the check box is disabled until you do), tick **I have read and agree to the terms of the License Agreement** and click **Accept and Continue**.
2. **Activate Your License** offers two paths:
 - **Try OptiZeus Free → Start Free Trial** starts a 7-day trial immediately; or
 - **Activate with License File**: click **Copy** next to the Hardware ID and send it to your vendor.

The Hardware ID has the form `xxxx-xxxx-xxxx-xxxx-xxxx`. It is derived from the system disk, motherboard, processor and computer name (Appendix C.4).

Important: Do not rename the server PC, replace its system disk or clone it after you have sent the Hardware ID. The licence would no longer match.

2.5.2 Activating a licence file

The vendor returns a `.lic` file. To activate it:

1. In the licence dialog, click **Select License File (.lic)**, choose the file and click **Activate License**; or, when already logged in as administrator, open **Config → Admin → License Activation**, click **Choose License File...**, select the file (`.lic`, `.license` or `.dat`, up to 1 MB) and click **Activate License**.
2. The dialog shows **License Activated!** with the customer, tier and expiry date. The trial ends.

License Activation also shows the **Current License Status: Status** (Active / Not Activated), **Edition**, **Expires** and **Licensed To**.

If activation fails, the message tells you why (hardware mismatch, invalid signature, deactivated licence, expired licence). See Appendix C.6 for the full list.

2.5.3 Trial period

- The trial lasts 7 days from the first start of the server. The top of the screen shows **Trial**: with the remaining time and an **Activate License** link.

- During the trial the server applies the Starter limits: 1 concurrent client, 1 PLC connection, 100 tags.
- A trial cannot be restarted. Running the setup or a patch over an existing installation keeps the trial state (`grace-period.dat`). Damaging or replacing that file ends the trial.
- When the trial expires, creating PLC connections, tags and screens and writing tags through the API are refused, but communication, alarms and the historian keep running (Appendix C.3).

Note: The licence dialog shows a tier summary ("Basic: 50 Tags, 5 Screens, 3 Users", "Professional: 200 Tags ..."). These figures are out of date; the limits in Chapter 1.7 and Appendix C are the ones the server applies.

2.5.4 Re-binding after a hardware change

If a valid licence stops working after a Windows update, a network-card change or a computer rename, the licence dialog offers **Already had a license that stopped working? → Re-bind license to this machine**. The server checks that the licence file was genuinely issued and not expired, and binds it to the current Hardware ID. This works only for the licence file already on the machine.

2.5.5 Moving a licence to another computer

1. Open **Config → Admin → License Activation**.
2. Under **Transfer License to Another Machine**, click **Deactivate License**. You must be logged in as an administrator (since 2.6.1.24 the server refuses deactivation from any other session).
3. Copy the **Deactivation Code (Copy)** or click **Download Return File**, and send it to your vendor with the new computer's Hardware ID.
4. The deactivated licence cannot be activated again on this computer. The computer returns to trial mode.

2.5.6 Renewal

A licence is normally issued for 365 days. Ask the vendor for a renewed `.lic` file before the expiry date and activate it as in 2.5.2.

2.6 Updating with a patch

Updates are delivered as a patch ZIP, for example `patch-v2.6.1.26.zip`, containing `INSTALL-PATCH.bat` and a `backend\` folder (and sometimes `electron\`). A patch does not touch the projects, databases, licence or certificates.

To install a patch:

1. Plan a stop: the server is stopped for one to two minutes and operators lose their screens meanwhile.
2. Make a backup of the active project (Chapter 26).
3. Extract the patch ZIP on the server and run `INSTALL-PATCH.bat` as administrator.
4. If OptiZeus is not in `C:\OptiZeus`, the patch asks for the installation path.
5. The patch runs the steps below and ends with "Patch v2.6.1.26 installed successfully!" and a list of changes.
6. Reload the page in every browser (Ctrl+F5) and restart every desktop application.

Step	Action
[1/8]	Stops the <code>OptiZeus</code> service
[2/8]	Backs up <code>server.js</code> , <code>license.js</code> , <code>drivers</code> and <code>services</code> to <code>backend\backup-pre-<version></code>

Step	Action
[3/8]	Copies <code>server.js</code> and <code>license.js</code>
[4/8]	Copies <code>drivers</code> , <code>services</code> , <code>middleware</code> , <code>utils</code> , <code>workers</code> , <code>scripts</code> and the WinSW binary
[5/8]	Installs <code>super-objects.json</code> only if none exists (your super objects are kept)
[6/8]	Replaces the web client in <code>backend\public</code> and <code>backend\frontend</code>
[6.4]	Replaces <code>electron\resources\app.asar</code> if the patch contains it; closes the running desktop application first
[6.45]	If <code>C:\OptiZeus\electron\resources\backend\server.js</code> exists, updates that server copy too
[6.5], [6.6]	Adds the OptiZeus pipe and callout symbol sets to the SVG library (existing entries are skipped)
[7/8]	Registers the service with WinSW if it is missing (keeps an existing <code>optizeus.xml</code>)
[8/8]	Starts the service and checks that it is RUNNING

Important: On customer installations the running server is often the copy in `C:\OptiZeus\electron\resources\backend\`. Step 6.45 updates it automatically. If you copy files by hand instead of running the patch, copy them to the folder named in the `<argument>` line of `backend\daemon\optizeus.xml` — otherwise the update has no effect.

Note: If you need to go back, stop the service, copy the files from `backend\backup-pre-<version>` back into `backend\` and start the service. The web client is not backed up; reinstall it from the previous release.

2.6.1 Checking the installed version

In a browser, open `http://<server>:3004/api/update/current-version`. The reply shows the version and build date, for example:

```
{ "version": "2.6.1.26", "buildDate": "2026-09-28" }
```

2.7 HTTPS (SSL/TLS)

The server opens the HTTPS port 3443 only when both `ssl-key.pem` and `ssl-cert.pem` are present in the folder of the running `server.js` (not the project folder). The certificate made by the setup is self-signed and valid only for the names `localhost` and `127.0.0.1`, and it is trusted only on the server PC. Other workstations show a certificate warning.

To use a certificate from your company certificate authority:

1. Obtain a certificate whose subject alternative names include the server host name and IP address that clients use.
2. Save the private key as `ssl-key.pem` and the certificate (with the intermediate certificates appended) as `ssl-cert.pem`, both in PEM format without a password.
3. Replace the two files in `C:\OptiZeus\backend\` (or in the folder of the running server copy).
4. Restart the service. The service log should show `[SSL] HTTPS server created with certificates from ...`.
5. On each client, open `https://<server-name>:3443` and check that the browser shows no warning.

HTTP on port 3004 stays open. If your policy requires HTTPS only, block TCP 3004 at the firewall for all clients. When OptiZeus runs behind a reverse proxy that terminates TLS, requests arriving with `X-Forwarded-Proto: http` are redirected to HTTPS.

2.8 Installing on Linux

2.8.1 Server installation

1. Copy the release to the Linux server and extract it.
2. Run:

```
sudo bash setup.sh
```

3. Type `1` for **SERVER + CLIENT**.
4. The script:
 - installs Node.js 24 from NodeSource if Node.js is missing or older than 20 (offline: place `node-v24.12.0-linux-x64.tar.xz` beside `setup.sh`);
 - stops an existing `optizeus` service;
 - copies the backend to `/opt/optizeus/backend` and the web client to `public/` and `frontend/`;
 - copies `node_modules` and runs `npm rebuild`, or runs `npm install --production`;
 - creates a self-signed certificate;
 - writes `/etc/systemd/system/optizeus.service` (runs as `root`, `Restart=always`, 5 s delay) and enables it;
 - creates a desktop launcher and starts the service.

Note: The bundled `node_modules` contain Windows native modules. On Linux, `npm rebuild` or `npm install` needs Internet access (or a local package mirror) to build the native drivers (SQLite, Siemens S7).

2.8.2 Managing the Linux service

Task	Command
Start	<code>systemctl start optizeus</code>
Stop	<code>systemctl stop optizeus</code>
Status	<code>systemctl status optizeus</code>
Log (live)	<code>journalctl -u optizeus -f</code>

Open TCP 3004 and 3443 in the Linux firewall (for example `ufw allow 3004/tcp` and `ufw allow 3443/tcp`).

2.8.3 Linux client

`sudo bash setup.sh`, option `2`, asks for the same server addresses as the Windows client setup and creates a desktop launcher that opens `/opt/optizeus/client/optizeus-launcher.html`.

2.8.4 Linux patch

Run `sudo bash install-patch.sh` from the extracted patch. It stops the service, backs up `server.js`, `license.js` and `package.json`, copies the backend modules and web client and starts the service again.

Note: If the systemd unit is missing, `install-patch.sh` creates one that runs as user `optizeus`. The setup does not create that user and installs the files as root. Create the user and give it ownership of `/opt/optizeus`, or change `User=` to `root`, before you start such a unit.

2.9 Uninstalling

2.9.1 Windows

WARNING: Uninstalling deletes `C:\OptiZeus` completely, including every project, database, historian archive, backup stored under the install folder and the licence. Export the projects (Chapter 5) and deactivate the licence (2.5.5) first if you intend to reuse them.

1. Run `C:\OptiZeus\UNINSTALL.bat` as administrator (or the **Uninstall OptiZeus SCADA** desktop shortcut).
2. Type `YES` to confirm.
3. The script stops and removes the services `OptiZeus` and `OptiZeusBackend`, ends every `node.exe` process, deletes the scheduled tasks, the desktop shortcuts, the firewall rules named **OptiZeus HTTP**, **OptiZeus HTTPS** and **OptiZeus SCADA**, and the folder `C:\OptiZeus`.
4. Node.js is not removed. Remove it with **Apps & features** if no other program needs it.

Note: The uninstaller tries to remove the trusted certificate by the name "OptiZeus SCADA". A certificate created by OpenSSL has the name `localhost` and stays in the trusted root store. Remove it with `certmgr.msc` (Trusted Root Certification Authorities) if required.

2.9.2 Linux

Run `sudo bash /opt/optizeus/uninstall.sh` and answer `y`. It stops and disables the service, removes the unit file, deletes `/opt/optizeus` and the desktop launcher.

2.10 Troubleshooting the installation

Symptom	Cause and remedy
"Cannot install Node.js. No bundled MSI and no internet."	Place <code>node-v24.12.0-x64.msi</code> beside <code>SETUP.bat</code> and run it again.
"Node.js installation may require a restart."	Restart the PC and run the setup again.
Page shows OptiZeus front-end assets not found	<code>backend\public\index.html</code> is missing. Re-extract the full release and run the setup or patch again; no <code>npm</code> build is needed on site.
Clients cannot connect, server PC can	Firewall rules missing (2.2.4).
Browser warning on HTTPS	Self-signed certificate; install a proper certificate (2.7).

Symptom	Cause and remedy
Service does not reach RUNNING	Read <code>backend\daemon\optizeus.err.log</code> ; run <code>node server.js</code> in a console (2.3.3) to see the error.
"Client limit reached (N clients)" at login	All licensed client seats are in use (Appendix C.1.2). Close unused tabs, or disconnect sessions in Config → Admin → Clients & Shutdown .
"Trial period has expired (7 days). Please activate a license."	Activate a licence (2.5.2).

3. Quick start: your first project

This chapter takes you through a complete small project from an empty installation to an operating screen: a storage tank Tank_101 with a level transmitter and high/low alarms, and a transfer pump Pump_P201 with a run command, run feedback and fault feedback. It is written for the engineer who configures OptiZeus for the first time. Every step points to the chapter that explains the function in depth. Allow about one hour.

3.1 The example

Item	Signal	Device address (Modbus TCP)	OptiZeus tag
Tank_101 level transmitter LT-101	0–10000 raw = 0–100 %	Holding register 40001	Tank_101_Level
Pump_P201 run command	Coil	00001	Pump_P201_Cmd
Pump_P201 run feedback	Discrete input	10001	Pump_P201_Run
Pump_P201 fault	Discrete input	10002	Pump_P201_Fault

The PLC is a Modbus TCP device at 192.168.1.50, port 502, unit ID 1. If you use a different protocol, choose it in step 3.3 and use its address format (Chapter 6).

Tip: No PLC available? Create the tags with **Data Source = Simulation** instead of **PLC / External Device** (step 3.4). A **Sine Wave** between 0 and 100 makes a good tank level, and **Toggle (alternating)** makes a Boolean change state on its own.

WARNING: The pump command in this example writes to a real output. Use a test PLC or a PLC in a safe state. Never test click actions against equipment that can start on a live plant.

3.2 Step 1 — Create and activate the project

To create a project for the example:

1. Log in as administrator (Chapter 4) and click **Config** in the header.
2. Click the **Projects** tab.
3. Under **Create New Project**, type `Line1_Demo` and click **Create Project**. The message "Project created successfully" appears and the project is listed under **Projects**.
4. In the **Projects** list, click **Switch** on the row of `Line1_Demo`.
5. Confirm "Switch to project "Line1_Demo"? The server will restart and all clients will be disconnected briefly."
6. The page reloads after about 10 seconds. Log in again. The **Projects** tab now shows **Active Project: Line1_Demo**.

A new project starts with an empty database and default settings. Its users are separate from those of other projects, so the default accounts exist again in the new project — change their passwords (Chapter 4).

Note: You can also build the example in the **Default** project; then skip this step. Projects are explained in Chapter 5.

3.3 Step 2 — Add the PLC connection

To add the Modbus TCP connection:

1. In **Config**, click the **PLC Connections** tab and then + **Add PLC Connection** (or + **Add Your First PLC**).
2. Fill in the dialog:

Field	Value
Connection Name *	PLC_Line1
PLC Brand *	Generic / Other
Model *	Modbus TCP Device
Protocol *	Modbus TCP
Host / IP Address *	192.168.1.50
Port *	502
Unit ID (Slave Address)	1
Enabled	ticked

3. Click **Test Connection**. A green line with ✓ confirms that the PLC answers; a red X shows the error.
4. Click **Add Connection**.

The connection appears in the list under its brand with a coloured status dot and a ► **Connect** / ▲ **Disconnect** button. OptiZeus also creates a communication alarm for the connection, so a lost PLC link raises an alarm automatically.

Note: PLC connections take effect immediately; they do not go through staging.

3.4 Step 3 — Create the tags

To create the tank level tag:

1. Click the **Tags** tab and then + **Add New Tag**.
2. Fill in the form:

Field	Value
Tag Name *	Tank_101_Level1 (letters, numbers and underscores only; cannot be changed later)
Folder	Tank_101
Tag Type *	Analog
Historian → Enable Recording	ticked; Recording Method Interval, Interval (ms) 1000
Data Source *	PLC / External Device
Label	Tank 101 level1
PLC Connection	PLC_Line1 (Modbus TCP)

Field	Value
PLC Address	40001
Unit *	%
Min Value / Max Value	0 / 100
Linear Scaling (Raw → Engineering)	ticked; Raw Range (PLC) Min 0, Max 10000; Engineering Range Min 0, Max 100; Clamp output to engineering range ticked

3. Check the **Preview** line: "Raw 0 → 0 % | Raw 5000 → 50.00 % | Raw 10000 → 100 %".

4. Click **Add Tag**.

Create the three pump tags the same way:

Tag Name	Folder	Tag Type	PLC Address	Label
Pump_P201_Cmd	Pump_P201	Bool	00001	Pump P201 run command
Pump_P201_Run	Pump_P201	Bool	10001	Pump P201 running
Pump_P201_Fault	Pump_P201	Bool	10002	Pump P201 fault

Boolean tags have no unit, range or scaling fields. Tick **Historian → Enable Recording** with **Recording Method On Change (Deadband)** for Pump_P201_Run so that the pump state can be trended.

3.5 Step 4 — Deploy the changes

With staging on (the default), new tags do not reach the running system until you deploy them. The **Deployment** tab blinks and shows the number of pending changes, and a floating **Deploy** button appears.

To deploy:

1. Click the **Deployment** tab (or the **Deploy** button).
2. Check the list under **Pending Changes**: four lines "Tag: ..." marked + (create).
3. In **Deploy Description (Optional)**, type Tank_101 and Pump_P201 tags.
4. Click **Deploy 4 Changes**.
5. The pending list empties and **Deploy History** shows "4 changes deployed – Tank_101 and Pump_P201 tags" with status **success**.

The project version increases by one patch step with every deploy (for example 1.0.0 → 1.0.1). Staging is described in Chapter 5.

To check the values, open the **Tags** tab: the tags now show live values. A value shown as xxxx means the tag has bad quality — the PLC is not connected or the address is wrong (Chapter 30).

3.6 Step 5 — Build the screen

To create the screen:

1. Click the **Viewer Builder** tab. Under **Synoptic Screens (Viewer Builder)**, click + **Create New Screen**.

- In the top bar of the builder, type `Tank_101 Transfer` in **Screen Name *** and `Tank farm transfer` in **Description**.

To place the tank and show its level:

- In the **Object Library** on the left, click the **Tanks** category and drag **Vertical Tank** onto the canvas.
- With the tank selected, go to the **Properties** panel on the right, find  **Animations (0)** and click **+ Add Animation**.
- Select the tag `Tank_101_Level`.
- In **Animation Type**, choose **Fill Level** (under Analog). Keep minimum `0`, maximum `100` and fill direction bottom-up.
- From the **Text** category, drag **Value Display** below the tank. Add an animation on it with tag `Tank_101_Level` and type **Value Display — Number (decimals, prefix/suffix)**; set 1 decimal and suffix `%`.

To place the pump and show its state:

- From the **Pumps** category, drag **Centrifugal Pump** to the right of the tank.
- Add an animation: tag `Pump_P201_Run`, type **Color Change (On/Off)**. Set the colour when FALSE to grey and when TRUE to green.
- Add a second animation: tag `Pump_P201_Fault`, type **Blink when True** (red/yellow).

To let the operator start and stop the pump:

- With the pump selected, go to **Click Action** and choose **Toggle Tag (Bool)**.
- In **Target Tag**, select `Pump_P201_Cmd`.
- Tick **Require Confirmation** so that the operator must confirm each command.
- Leave **Security Letter (A-Z)** at **No Security Required** for now (Chapter 24 explains security letters).

To connect the objects visually:

- From **Piping**, drag a **Horizontal Pipe** between the tank outlet and the pump, and resize it.
- From **Text**, drag a **Text Label** above each object and type `TK-101` and `P-201`.
- Click  **Save**.

The screen is saved to staging. Deploy it as in 3.5 ("Screen: ..." appears in the pending list). The synoptic builder is described in Chapters 10–12.

3.7 Step 6 — Configure the alarms

To create the tank level alarms:

- Click the **Alarms** tab, sub-tab **Alarm Configurations**, and then **+ Create Alarm**.
- Fill in the form:

Field	Value
Alarm Name *	<code>Tank_101_Level_Alarm</code>
Tag *	<code>Tank_101_Level</code>
Alarm Type *	Analog (Threshold-based)
Description	Tank 101 level

Field	Value
Enabled	ticked
Deadband (for alarm clearing)	2
High-High Alarm (Critical)	ticked; Threshold 90 , Severity Critical, Delay (sec) 0 , Message TK-101 level very high
High Alarm (Warning)	ticked; Threshold 80 , Severity Warning, Delay (sec) 5 , Message TK-101 level high
Low Alarm (Warning)	ticked; Threshold 20 , Severity Warning, Delay (sec) 5
Low-Low Alarm (Critical)	ticked; Threshold 10 , Severity Critical, Delay (sec) 0

3. Click **Create Alarm**.

To alarm the pump fault:

- Click **+ Create Alarm** again. **Alarm Name** Pump_P201_Fault_Alarm , **Tag** Pump_P201_Fault , **Alarm Type** Boolean (On/Off), **Trigger Alarm When** Value is ON (True/1), **Severity (On Trigger)** Critical, **Alarm Message (On)** P-201 fault , **Related Analog Tag (for Trend View)** Tank_101_Level .
- Click **Create Alarm** and deploy both alarms (3.5).

Alarms are described in Chapter 17.

3.8 Step 7 — Make the screen reachable

Operators open screens from the dashboard. The dashboard's screen list or **Screen Navigation** panel lists every deployed screen automatically. To also put the screen on the header:

- Click the **Display** tab.
- Under **Navigation Buttons**, tick the entry  **Tank_101 Transfer**.
- Click **← Back to Dashboard**. The header now has a **Tank_101 Transfer** button.

Layouts and navigation are described in Chapters 4 and 14.

3.9 Step 8 — Operate the plant

Log out and log in as an operator (a user with the `operator` role; Chapter 24 shows how to create one).

To start the pump:

- Click **Tank_101 Transfer** in the header, or select the screen in the dashboard.
- Check the tank level value and fill.
- Click the pump. Confirm the command.
- OptiZeus writes TRUE to `Pump_P201_Cmd` . When the PLC switches the pump on, `Pump_P201_Run` becomes TRUE and the pump turns green.
- Click the pump again and confirm to stop it.

Note: Users with the `viewer` role cannot write values; the click has no effect for them.

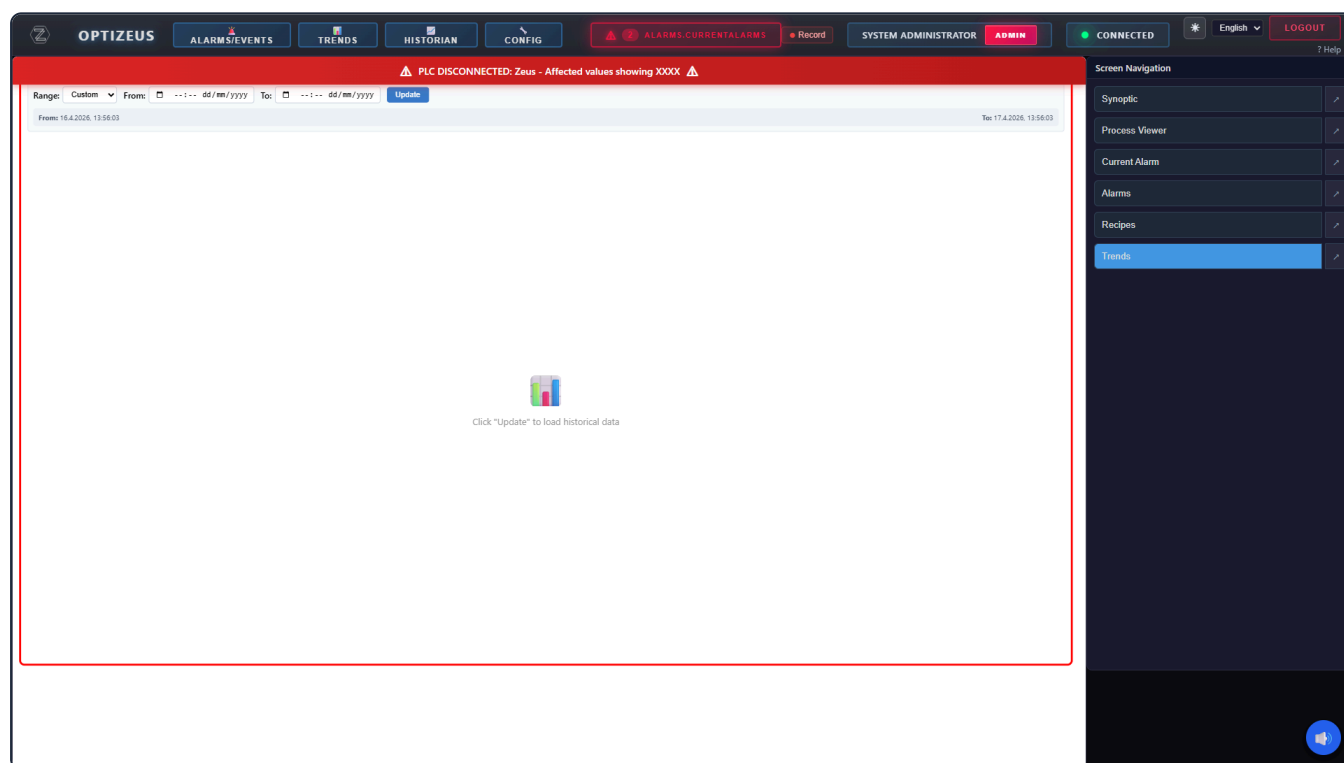
To handle an alarm:

1. When the level passes 80 %, the alarm counter button in the header (▲ with the number of current alarms) changes colour, a popup appears and an alarm sound plays (default settings).
2. Click the alarm counter button to open the list of current alarms.
3. Click **Acknowledge** on the alarm, choose or type a reason (at least 10 characters by default) and confirm.
4. The alarm stays in the list as acknowledged until the level falls below 78 % (threshold minus deadband); then it clears.

3.10 Step 9 — View the trend

1. Click **Historian** in the header.
2. In the **Tags** list on the left, search for `Tank_101` and click `Tank_101_Level`.
3. Choose the time range **1h**. The chart shows the level recorded since step 4.
4. Add `Pump_P201_Run` to compare pump state and level. Use **+ Add Pane** to show it in a separate pane.
5. Click **Alarms** to mark alarm times on the chart.

The historian is described in Chapter 19.



Historian trend

3.11 What you have built

Configured	Where	Chapter
Project <code>Line1_Demo</code>	Config → Projects	5
Modbus TCP connection <code>PLC_Line1</code> with automatic communication alarm	Config → PLC Connections	6
Four tags with scaling and historian recording	Config → Tags	7

Configured	Where	Chapter
Screen Tank_101 Transfer with fill, value, colour, blink and toggle	Config → Viewer Builder	10–12
Analog HH/H/L/LL alarm and Boolean fault alarm	Config → Alarms	17
Header navigation button	Config → Display	14

Next steps: turn the tank and pump into reusable super objects (Chapter 13), add a recipe (Chapter 20) and a daily shift report (Chapter 21).

4. The workspace: login, navigation, layouts and languages

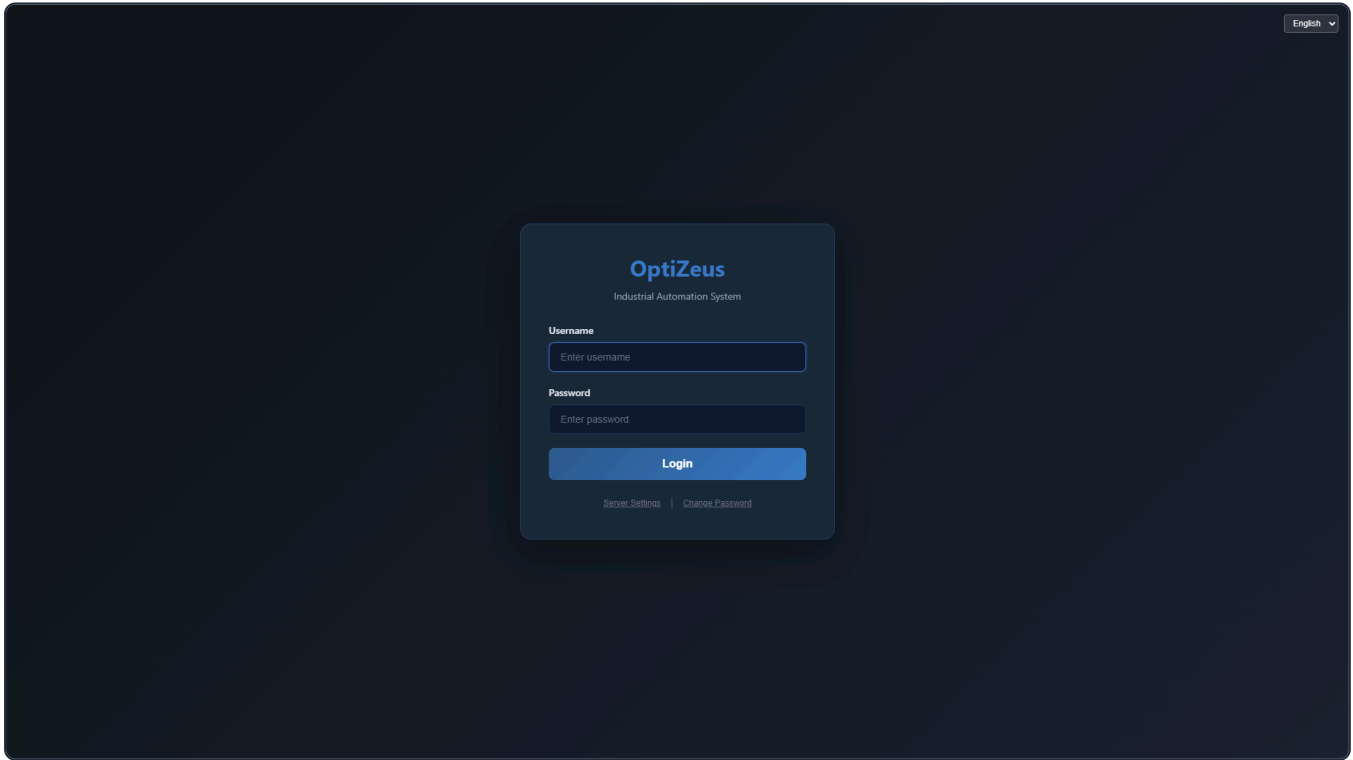
This chapter describes what every user sees after opening OptiZeus: the login page, password and two-factor prompts, the header bar with its buttons and indicators, the navigation bar, the dashboard layout, the language and theme selectors, the on-screen keyboard and the indications that appear when the server or a PLC is not connected. Operators and supervisors should read 4.1–4.4 and 4.8–4.9; engineers who design the workspace need all sections.

4.1 Opening OptiZeus

Client	How to open
Browser	<code>http://<server>:3004</code> or <code>https://<server>:3443</code>
Launcher shortcut OptiZeus SCADA	Tries the configured servers in order and opens the first that answers (Chapter 2)
Desktop application OptiZeus SCADA (App)	Opens the login page; a stored login is always cleared at start

When the page opens and no licence or trial exists, the licence dialog appears first (Chapter 2). When the page cannot reach a server, the login page shows **Server Configuration** instead (4.2.3).

4.2 The login page



Login page

Element	Meaning
Username	User name; not case-sensitive
Password	Password
Login	Logs in. Shows Logging in... while waiting.
 On-Screen Keyboard	Opens the on-screen keyboard for the selected field (4.7)
SSO button	Shown only when single sign-on (OpenID Connect) is configured; the label is set by the administrator, default Sign in with SSO (Chapter 24)
Server Settings	Opens Server Configuration to point this client at another server (4.2.3)
Change Password	Changes the password of a named account without logging in (4.2.4)

4.2.1 Logging in

To log in:

1. Type your **Username** and **Password**.
2. Click **Login** or press Enter.
3. If your account uses two-factor authentication, the page asks for **Two-Factor Authentication Code**. Type the 6-digit code from your authenticator app and click **Verify & Login**. **Back to Login** returns to the password step.

Message	Meaning
Invalid credentials	Wrong user name or password. The attempt is written to the event log.
Account is disabled. Contact administrator.	The account is disabled (Chapter 24).
Invalid MFA code	The code was wrong or expired. Check the clock of the phone.
Too many login attempts. Try again later.	After 5 failed attempts within 15 minutes from one computer, logins from that computer are blocked for 30 minutes (defaults; Chapter 25).
Client limit reached (N clients)...	All licensed client seats are in use (Appendix C).
Connection error. Please check if the server is running.	The server did not answer.

4.2.2 The first login

A new installation (and every new project) contains three default accounts: an administrator, an operator and a viewer, each with a well-known password given in the release documentation.

When you log in to one of these accounts with its default password, the login shows **Change the Default Password** with the text "This account still uses its factory default password, which is published. You must set your own password before continuing." The dialog cannot be cancelled: enter the default password as **Current Password**, choose a new one (4.2.4) and click **Change Password**; the session then starts. (Since version 2.6.1.24; not on a server running in demo mode.)

Important: The change is forced only for an account that actually logs in. Immediately after installation, log in with the default administrator account and set its password. Then change the passwords of the default operator and viewer accounts, or disable them, in **Config → Admin → Users** (Chapter 24). Leaving the default passwords in place is the most common security weakness of an installation.

A forced change also happens when a password has expired. Every local account has a password lifetime (default 180 days from the last change). After it, the login shows **Password Expired - Change Required** with the text "Your password has expired. You must change it before continuing." The dialog cannot be cancelled. Accounts that log in through Active Directory are exempt.

4.2.3 Server settings (client workstations)

The login page checks the server on the same address. If none answers within 3 seconds and no server was configured before, it shows **Server Configuration**:

Field	Meaning	Default
Server IP Address	IP address or name of the OptiZeus server, e.g. 192.168.1.100	—
Server Port		3004
Test Connection	Checks the server; reports Connection successful! , Connection timed out. Check the IP and port. or Cannot reach server. Check IP address and ensure the server is running.	
Save & Connect	Stores the address in this browser and reloads the page	

The address is stored in the browser (per workstation and browser profile). To change it later, click **Server Settings** on the login page.

4.2.4 Changing your password

To change your password:

1. Click **Change Password** on the login page, or the password option in the application after you have logged in.
2. On the login page, check **User Name**: it is filled in from the login form and names the account whose password is changed. After login the dialog always changes your own password and has no **User Name** field.
3. Enter **Current Password**, **New Password** and **Confirm New Password**.
4. The dialog lists the **Password Requirements: Minimum 8 characters, At least one uppercase letter, At least one lowercase letter, At least one number, At least one special character** (defaults; the administrator sets the policy, Chapter 25). Previously used passwords may be refused (password history).
5. Confirm. After a change from the login page the message "Password changed! Please login with your new password." appears.

4.2.5 Two-factor authentication prompt

When the administrator requires two-factor authentication (MFA) for all users and your account has none yet, the dialog **Setup Two-Factor Authentication** opens directly after login with the text "Two-factor authentication is required by your administrator. You must complete setup to continue. Canceling will log you out."

To set up MFA:

1. **Step 1: Scan QR Code** — scan the code with an authenticator app (any TOTP app).
2. **Step 2: Verify Code** — type the 6-digit code shown by the app.
3. Click **Verify & Enable MFA**. **MFA Enabled Successfully!** confirms it.

Cancel & Logout logs you out. From then on every login asks for the code (4.2.1). MFA administration is described in Chapter 24.

4.2.6 Automatic logout

The administrator can set an automatic logoff after a period without mouse, keyboard or touch activity, and can hide the **Logout** button (for example on shared kiosk stations). Both settings are in the security settings (Chapter 25). A login token is valid for at most 24 hours.

4.3 Screen elements after login

After login OptiZeus shows the **dashboard**: the header bar at the top and, below it, the dashboard panels defined by the active layout (4.5).



Dashboard

4.4 The header bar



Header and navigation bar

From left to right:

Element	Meaning
Logo	Company logo (Config → Admin → Branding or the layout settings)

Element	Meaning
Title	Application title, shown in capitals, default OptiZeus
Navigation buttons	Open views and screens (4.4.1)
Alarm counter (⚠ with a number)	Number of current alarms. Red when unacknowledged alarms exist, another colour when all are acknowledged. Click it to open the current-alarm list (Chapter 17).
● Record	Records the screen to a video file; while recording shows ● REC with the elapsed time and pause, resume and ■ Stop buttons. Stop downloads the recording. Can be hidden in the display settings.
Pending Deploy	Administrators only: number of configuration changes waiting for deployment; click to open Staging & Deployment (Chapter 5)
User information	Full name (or user name) and role in capitals, e.g. OPERATOR
Connection status	Green dot and Connected , or red dot and Disconnected (4.9). With server redundancy a badge PRIMARY (green) or BACKUP (red) shows which server serves this client.
Theme button (☀ / 🌙)	Switches between dark and light theme
Language selector	Changes the display language (4.6)
Logout	Ends the session (can be hidden by the administrator)
? Help	Opens the online help (4.10)

Each element except the theme button and the language selector can be hidden in the layout settings (4.5).

4.4.1 Navigation buttons

A navigation button opens a view or a synoptic screen. Without a custom configuration the header shows three buttons:

Button	Opens	Visible to
 Alarms/Events	Alarm and event views (Chapters 17–18)	All
 Historian	Historian (Chapter 19)	All
 Config	Configuration (engineering)	Administrators only

A new project created in **Config** → **Projects** also starts with an **OEE** button.

The views that can be placed on a navigation button:

Button label	View
Alarms/Events	Alarm and event lists
Trends	Tag trends
Historian	Historian
Screen Replay	Replay of past screen states
Shift Report	Shift handover report
KPI Dashboard	KPI dashboard

Button label	View
Config	Configuration (admin only)
Audit Trail	Audit trail (admin only)
Batch Control	Batch execution
Help	Online help
OEE	OEE dashboard
Smart Alarms	Smart alarm grouping
Notepad	Operator notepad
SOE	Sequence of events
Procedures	Operating procedures
Voice	Voice commands
any synoptic screen	That screen

Buttons can be collected in **groups** (and sub-groups). Clicking a group opens a drop-down list of its buttons. A button or group can be bound to a tag or to an alarm group: when the bound tag is active, or the alarm group has active alarms, the button blinks red and shows a count badge. This lets operators see in which area an alarm is without opening the alarm list.

4.4.2 Customising the navigation buttons

To choose which buttons appear:

1. Open **Config** → **Display**.
2. Under **Navigation Buttons**, tick the views and screens that must appear in the header. The change is saved immediately, for your user layout and, if one exists, for this station's layout.
3. For icons, labels, order, groups and tag or alarm-group bindings, use **Open Layout Builder** →  **Header** → **Navigation Buttons** (4.5.2). Each row there has the columns **Visible**, **Icon**, **Label**, **Action** and a **Remove** button.

4.5 Dashboard layouts

The area below the header is arranged by the **layout**. A layout defines the layout mode, the panels, the header settings and the navigation buttons.

4.5.1 Layout modes

Select the mode in **Config** → **Display** → **Layout Mode**:

Mode	Arrangement
Custom Layout	Free arrangement designed in the Layout Builder (4.5.2); the default for new projects
Single Column	All panels stacked

Mode	Arrangement
2 Columns	Synoptic viewer on the left, panels in a right column of adjustable width
3 Columns	Left, centre and right columns; widths in Panel Widths (Left Panel Width (px), Center Panel Width (px), Right Panel Width (px), Panel Gap (px))
4 Columns	Four columns of adjustable width

For the column modes, **Panel Visibility** switches individual panels on and off and **Tag Columns (Auto, 1–6)** sets the number of columns in the tag panel.

4.5.2 The Layout Builder

To design a custom layout:

1. Open **Config** → **Display** and click  **Open Layout Builder**.
2. Type a name in **Layout Name**.
3. On the  **Panels** tab, drag panels from **Panel Library** onto the canvas, then move and resize them. Positions and sizes are in percent of the screen, so the layout adapts to the monitor.
4. Select a panel to edit it in **Properties: Panel Info** (title), **Position & Size**, **Panel Configuration** (for example the synoptic screen shown by a **Synoptic View** panel, or which screens a **Screen Navigation** panel lists and in which order).
5. On the  **Header** tab, set **General**, **Logo & Title**, **Navigation Buttons**, **Elements Visibility**, **Header Instruments** (live values in the header) and **Panel Menu Bars**. In **Elements Visibility** you can also pre-select an alarm group for the layout, so that operators of this station see only their area's alarms, and protect that filter with a security letter.
6. Use  **Preview** to check the result,  **Presets** to start from a prepared arrangement, and **Undo (Ctrl+Z)** / **Redo (Ctrl+Y)**,  **Duplicate**,  **Front**,  **Back**,  **Delete** while editing. **Page Background** and **Layout Description** are at the bottom of the properties.
7. Click **Save Layout**. To make the layout apply to this workstation only, choose the option to also save it as a station layout (4.5.3).

Panel types:

Panel	Shows
Alarm Panel	Current alarms with acknowledgment
Event Log	Recent events
Trend Chart	Live trend
Synoptic View	A synoptic screen
Tag List, Tag Grid	Live tag values; click a tag for its faceplate
Gauge Panel	Gauges
Datalog Viewer	Logged data
System Status	Server and communication status
User Info	Logged-in user
Clock	Date and time

Panel	Shows
Notifications	Notification messages
Screen Navigation	Buttons for synoptic screens, optionally grouped; can be docked to a screen edge
Quick Links	Shortcut buttons
Breadcrumb	Navigation path

Layouts in depth are covered in Chapter 14.

4.5.3 Station layouts

A layout can belong to a user or to a workstation (station). When a user logs in, OptiZeus uses the first of:

1. the **station layout** of this workstation;
2. the **user layout** of the user;
3. the project's default layout.

A workstation is identified by its IP address: the server gives every client an ID of the form `client_192_168_1_21`. The ID stays the same when the browser cache is cleared, but two browsers on one PC count as the same station, and a PC that changes its IP address becomes a new station.

To manage station layouts, open **Config → Admin → Clients & Shutdown → Station Layouts (Station Layout Manager)**. The table lists **Station**, **IP**, **Nav Buttons**, **Updated** and **Actions**; the current workstation is marked **THIS STATION**. **Edit** opens the Layout Builder for that station; **Remove** deletes the station layout so that the station falls back to the user layout.

Tip: Use station layouts for fixed operator stations (control room, field panel, supervisor office), so that the same operator sees the layout that fits the monitor and area of the station at which he or she logs in.

4.6 Languages and right-to-left display

4.6.1 Changing the language

Select the language in the language selector in the header. The interface changes immediately. The choice is kept in this browser and is also recorded in your user account, but in 2.6.1.26 a login from another workstation does not apply it automatically — select the language there once.

Language	Direction	Delivered translation
English	Left to right	Complete (reference)
עברית (Hebrew)	Right to left	Most of the standard texts
العربية (Arabic)	Right to left	Most of the standard texts
Ελληνικά (Greek)	Left to right	Partial

For Hebrew and Arabic the whole page is mirrored (right-to-left). The configuration pages stay left-to-right so that tag names, addresses and expressions remain readable.

Note: Texts without a translation are shown in English. Many texts inside the configuration pages and dialogs are not yet translatable in 2.6.1.26 and always appear in English. Screen texts that you type yourself (labels on synoptic screens, alarm messages) are shown as entered; alarm messages can be translated in the **Language** tab.

4.6.2 Adding a language and editing translations

Administrators manage languages in **Config** → **Language** (**Language Settings**):

Area	Function
System Language	Default language
Available Languages	Enabled languages; Add Language with Code (e.g. fr), Name (e.g. French), Native (e.g. Français) and Direction (LTR / RTL); English cannot be deleted
Translation Editor	Table Translation Key , English Value , Translated Value , with Search Translations , Translation Coverage (translated / untranslated) and Save Translations
Alarm Message Translations	Translations of alarm messages
Export / Import	Exchange translations as a file, e.g. for a translator

Translations saved here override the delivered ones.

Note: Right-to-left mirroring is applied only to the language codes he and ar . A language added with direction RTL under another code is displayed left-to-right in 2.6.1.26.

4.7 The on-screen (touch) keyboard

On touch-screen stations OptiZeus shows its own keyboard whenever an input field is selected: a numeric keypad for numbers, dates and times, and a full keyboard for text. On the login page the keyboard opens automatically on touch devices and can always be opened with  **On-Screen Keyboard**. Pressing Enter on a physical keyboard while the on-screen keyboard is open submits the login.

To switch touch-screen mode on for a workstation:

1. Open **Config** → **Admin** → **Clients & Shutdown** → **Connected Clients**.
2. In the row of the workstation (columns **Client Name**, **Client ID**, **IP Address**, **Touch Screen**, **Last Seen**, **Actions**), tick **Touch Screen**.

The setting belongs to the workstation and is applied the next time it connects. For the desktop application, **Enable Touch Screen Mode** in the Electron client settings does the same (Chapter 2). Touch operation is described further in Appendix A.

4.8 Themes

The  /  button in the header switches between the dark and light themes. The choice is stored in the browser and in the settings of the workstation. Synoptic screens keep the colours they were designed with.

4.9 Connection indications

OptiZeus never shows an old value as if it were current. When data is not available, the display says so.

Indication	Where	Meaning	Action
Connected (green dot)	Header	The client receives live data	—
Disconnected (red dot)	Header	The WebSocket link to the server is lost; the client reconnects automatically	Check the network and the server
Alarm "Communication Lost - Server disconnected" (critical)	Alarm list	Added by the client while the server link is lost	Clears when the link returns
SERVER CONNECTION LOST - All values showing XXXX	Red banner on synoptic screens	No values can be trusted	As above
PLC DISCONNECTED: PLC_Line1 - Affected values showing XXXX	Red banner on synoptic screens	Named PLC connections are down; the tags of other PLCs are still live	Check the PLC and network; see the communication alarm
XXXX in a value field	Screens, faceplates, gauges	The tag has bad quality: server link lost, PLC not connected or the address cannot be read	Chapter 30
---	Screens	The tag exists but has no value yet (for example a memory tag that was never written)	—
Server is starting up, loading tags... Please wait.	Blue banner at the top	The server has just started and is loading the tag database	Wait
PRIMARY / BACKUP	Header	With server redundancy, which server serves the client	Chapter 27

Note: Memory, calculated and simulated tags do not depend on a PLC. They keep showing values when a PLC is disconnected and show `xxxx` only when the server link itself is lost.

WARNING: While a screen shows `xxxx`, the process is not being supervised from that screen, and alarms of the affected PLC cannot be raised. Treat a lasting communication loss as a plant upset and follow the site procedure for loss of view.

Other messages that can appear:

Message	Meaning
New Configuration Deployed — "A new version of the system configuration has been deployed. Would you like to refresh to get the latest version?"	An engineer has deployed changes (Chapter 5). Refresh Now reloads the page; Later keeps the current view.
"You have been disconnected by the administrator."	An administrator ended the session from Clients & Shutdown .
"Server is shutting down: ..."	The server is being stopped.

Message	Meaning
KIOSK MODE — "This workstation is configured for kiosk mode. You must operate in fullscreen mode."	A kiosk user left full screen; click Enter Fullscreen .
Trial: ... remaining / Expired - License required	Trial licence (Chapter 2). Activate License opens the activation page.

4.10 Setup wizard, guided tours and help

4.10.1 Setup wizard

After an administrator logs in to a project that is not fully configured, the dialog **Welcome to OptiZeus SCADA** — "Let's set up your system step by step" — shows the progress in percent and six steps:

Step	Marked done when
Create Project	Always
Add PLC Connection	At least one PLC connection exists
Import or Create Tags	At least one tag exists
Create Synoptic Screen	At least one screen exists
Create Operator Users	A user other than the default administrator exists
Configure Alarms	At least one tag exists

Clicking an open step (**Go** →) opens **Config**; choose the matching tab yourself. **Close & Setup Later** closes the dialog until the next login; **Don't show again** switches it off in this browser; when all steps are done the button reads **Start Using OptiZeus**. The same checklist is available at any time in **Config** → **Admin** → **Setup Wizard**.

Note: The wizard counts the default operator and viewer accounts as "operator users" and treats alarms as configured as soon as tags exist, so these two steps can show **Done** before you have done them. Use the wizard as a reminder, not as a check of completeness.

4.10.2 Guided tours

? Help → **Guided Tours** lists step-by-step tours that highlight the elements on the real screens: Dashboard Overview, Tags & Monitoring, User Defined Types (UDT), Synoptic Screens (HMI), Super Objects, Alarms & Notifications, Recipe Manager, Process Viewer, PLC Communication, Trends & History, Batch Control (ISA-88), Security & CFR 21 Part 11, Project Management, Staging & Deployment, OPC-UA Server, Data Storage & Historian, Reports & Visual Builder, Display & Layout Settings, AI Assistant, SVG Library, Data Import, Floor Display Mode, User Management, Notification Channels, Synoptic Builder Advanced and Keyboard Shortcuts. A tour opens the view it needs by itself; close it at any time.

4.10.3 Online help

? Help opens **OptiZeus System - User Manual**, the built-in help with sections for every area (Overview, Dashboard, Synoptic Screens, Super Objects, Alarms & Notifications, Trends & History, Tags & Monitoring, Recipe Manager, Configuration, Backup & Restore, User Roles, Security & CFR 21 Part 11, Touch Screen Mode, Keyboard

Shortcuts, Troubleshooting, Video Tutorials, Guided Tours and more), with links to the PDF manuals. **Back to Dashboard** returns to the dashboard.

4.11 Returning to the dashboard

Every full-page view (Config, Historian, Alarms/Events, Help and so on) has a back button in its top-left corner (← **Back to Dashboard** or **Back**) that returns to the dashboard. A synoptic screen opened full page returns with ← **Back to Dashboard** as well.

OPTIZEUS SCADA · OPERATION MANUAL

Part II — Configuration

5. Projects, databases, staging and deployment

This chapter explains how OptiZeus organises a configuration into projects, where each project keeps its data, how to create, switch, copy, export, import and delete projects, and how configuration changes are staged and deployed to the running system or to other servers. It is written for engineers and administrators. Operators only need to know 5.5.6: what happens on their screen when an engineer deploys.

5.1 What a project is

A **project** is a complete, self-contained configuration: PLC connections, tags, UDTs, screens, super objects, alarms, recipes, users, reports, settings — and the history the running system records for it (alarms, events, audit trail, historian). One OptiZeus server runs exactly one project at a time, the **active project**.

Typical uses:

Use	Example
One project per plant or line	<code>Line1</code> , <code>Line2</code> on a shared engineering PC
Development copy	Change <code>Line1_Dev</code> while <code>Line1</code> runs on the production server
Customer delivery	Export the project on the engineering PC, import it on the site server
Training	A <code>Training</code> project with simulated tags

5.2 Where projects are stored

Project	Folder
Default	The backend folder itself: <code>C:\OptiZeus\backend\</code> (or the running server copy, Chapter 2)
Any other project	<code>C:\OptiZeus\backend\projects\<name>\</code>

The file `backend\active-project.json` names the active project:

```
{
  "name": "Line1",
  "directory": "projects\\Line1"
}
```

If the named folder does not exist when the server starts, the server logs a warning and falls back to the Default project.

5.2.1 Contents of a project folder

File or folder	Content
<code>scada.db</code> (+ <code>scada.db-wal</code> , <code>scada.db-shm</code>)	The project database (5.3)
<code>alarm-settings.json</code>	Global alarm settings, alarm groups and areas

File or folder	Content
layout-settings.json	Default dashboard layout and header
super-objects.json , super-object-instances.json	Super object definitions and instances
udts.json	UDT definitions file (when present; UDTs are also stored in the database)
roles.json	Roles and their permissions and security letters
security-settings.json	Password policy, lockout, session, directory settings
notification-settings.json	Notification channels
ai-settings.json	AI assistant settings
electron-config.json	Desktop application settings (Chapter 2)
areas.json , zones.json , named-queries.json , historian-tiers.json , oidc-config.json	Present when the corresponding functions are used
historian-archives\	Binary historian archive written by the historian worker
backups\	Project backups (Chapter 26)
uploads\	Uploaded images and logos
store-forward\	Store-and-forward buffer (Chapter 27)
runtime\	State written by the running server (5.2.2)

5.2.2 The runtime folder

runtime\ holds files that the server writes while it is used, not configuration that an engineer edits:

File	Content
client-settings.json	Registered workstations (one entry per IP address), their names and touch-screen setting
notification-history.json	Sent notifications
alarm-notification-log.json	Alarm notification log
other logs and results	Diagnostic files

Deleting the contents of runtime\ resets this accumulated state (for example the workstation list) without touching the configuration. It is excluded from releases and from project export.

5.2.3 Files outside the project

Some files belong to the installation, not to a project, and are the same whatever project is active:

File	Content
backend\active-project.json	Active project
backend\scada.db (SVG library tables)	The SVG symbol library is always kept in the Default project's database and shared by all projects
backend\.jwt-secret , .hmac-secret	Installation secrets
backend\ssl-*.pem	HTTPS certificate
backend\service-config.json , redundancy-config.json , server-deployment-config.json	Service, redundancy and multi-server settings
license.dat and the other licence files	Licence (one level above backend)

5.3 What lives in the project database

scada.db is a SQLite database in WAL mode. Do not open it with other tools while the server runs, and never copy only scada.db without its -wal file — recent changes are in the WAL file until SQLite merges them.

Group	Main tables
Communication	plc_connections , opcua_servers , ethernetip_servers , opc_ae_servers , data_sources
Tags	tags_config , udt_definitions , udt_instances , tags
Screens	synoptic_screens , screen_folders , super_objects , super_object_instances , svg_library (Default project only), charts
Alarms	alarm_config , alarms , alarm_translations , sequence_of_events
Events and audit	event_log , audit_trail , electronic_signatures
History	historian_data , tag_data , datalogs
Production	recipes , procedures , procedure_steps , procedure_executions , procedure_step_log , downtime_events , equipment_status
Reports	report_templates , visual_report_layouts , scheduled_reports , schedules
Users and security	users , user_sessions , password_history , user_layouts , station_layouts
Notifications	notification_settings , notification_users , notification_groups , gsm_modems , sms_logs , sms_acl , voice_callout_config , webhooks
Integration	opcua_server_settings , opcua_server_tags , remote_sites , shared_dashboards
Operator tools	operator_notes , anomaly_detection_config , anomaly_history , ip_cameras
Languages	languages , translations
System	system_settings , project_version , pending_changes , deploy_history

Because users are stored in the project database, **each project has its own user accounts**. A new project gets the default accounts again (Chapter 4).

5.4 Managing projects

All project functions are in **Config** → **Projects (Project Management)** and require the administrator role.

Area	Content
Active project banner	Active Project: name, project version (v1.0.12), Build # , and Last deployed: date by user
Version Control	Bump Minor , Bump Major (5.4.7)
Create New Project	Name field and Create Project
Import Project	Name field and Select ZIP & Import
 Merge Import Into Active Project	5.4.6
Projects	List of all projects with Refresh

Each row of the **Projects** list shows the name ((**Active**) for the running one), the folder (**Default installation directory** or `projects/<name>/`), **Database exists** or **No database yet (will be created on switch)**, **Created:**, **Last update:**, **Version:**, and the buttons **Export**, **Switch** (not on the active project) and **Delete** (not on Default or the active project).

5.4.1 Creating a project

1. In **Create New Project**, type the name. Use letters, numbers, spaces, hyphens and underscores; `Default` is reserved.
2. Click **Create Project**.

OptiZeus creates `projects\<name>\` with `backups\`, `runtime\` and the default configuration files. The database is created the first time the project is started.

5.4.2 Switching the active project

WARNING: Switching restarts the server. Every client is disconnected, PLC communication, alarm supervision, datalogging and historian recording stop for the time of the restart (typically 10–60 seconds), and the new project may connect to different PLCs. Switch only in a planned stop.

1. Click **Switch** on the project's row.
2. Confirm "Switch to project "<name>"? The server will restart and all clients will be disconnected briefly."
3. Connected clients receive the message "Server restarting for project switch...". The server stops its datalogs, disconnects every PLC, closes the database and exits.
4. When it runs as the Windows service, the service manager restarts it within seconds. Otherwise the server starts a new copy of itself.
5. The browser reloads after 10 seconds. Log in again — with an account of the new project.

The switch is recorded in the event log of the old project.

Note: If the server runs in a console window (not as a service) and cannot start the new copy, it keeps running with the old project and logs "restart Node manually to apply the project switch fully". Restart it by hand.

5.4.3 Exporting a project

Export downloads `project-<name>-<date>.zip` containing:

- `project-meta.json`;
- `scada.db`, `scada.db-wal` and `scada.db-shm` (a checkpoint is attempted first; the export does not stop the server);
- the standard configuration files (`alarm-settings.json`, `layout-settings.json`, `super-objects.json`, `super-object-instances.json`, `roles.json`, `security-settings.json`, `notification-settings.json`, `ai-settings.json`, `electron-config.json`) and `udts.json`;
- `historian-archives\` and `uploads\`.

Not included: `runtime\`, `backups\`, `store-forward\`, `areas.json`, `zones.json`, `named-queries.json`, `historian-tiers.json`, `oidc-config.json`, the shared SVG library and the licence. Copy those files separately if the target needs them.

Important: The export contains the complete database, including users with their password hashes, notification settings and PLC connection data. Treat the file as confidential.

5.4.4 Importing a project

1. In **Import Project**, type the name the project will have on this server.
2. Click **Select ZIP & Import** and select an exported ZIP (up to 500 MB).
3. The message reports the number of files and the counts of tags, recipes, UDTs and screens found in the imported database.
4. Switch to the imported project (5.4.2) to run it.

An import always creates a new project; it never overwrites an existing one ("Project "<name>" already exists. Delete it first or choose a different name.").

5.4.5 Copying a project

There is no separate copy command. To copy a project, **Export** it and **Import** the ZIP under a new name. This is also the way to move a project to another server.

5.4.6 Merge import

 **Merge Import Into Active Project** copies content from an exported project ZIP into the **active** project instead of creating a new one — for example to add a finished line section built on the engineering PC to the running site project.

Setting	Options
Entities	Tags, Alarm Configs, UDT Definitions, UDT Instances, PLC Connections, Synoptic Screens, Super Objects, Alarm Groups + Areas
Conflict policy	Skip existing (safe — only adds new) — items whose name already exists are left alone; Overwrite existing (source wins on every conflict)

To merge:

1. Tick the entities to merge and choose the conflict policy.

2. Click  **Preview (Dry Run)** and select the ZIP. The report table (**Entity**, **Source rows**, **Would import**, **Skipped (conflicts)**, **Errors**) shows what would happen; nothing is written.
3. Click  **Select ZIP & Merge**, select the same ZIP and confirm. The table now shows **Imported** counts.

Entities are matched by name and merged in the order PLC connections, UDTs, UDT instances, tags, alarms, screens, super objects.

WARNING: A merge writes directly into the running project — it does not go through staging and cannot be undone or rolled back. New PLC connections start communicating and new alarms become active immediately. Make a backup (Chapter 26) and run the dry run first.

5.4.7 Project version

Every project carries a version `MAJOR.MINOR.PATCH` and a build number, shown in the active-project banner and in the project list.

Event	Effect
Each deploy (5.5.4)	PATCH + 1, build + 1, "last deployed" date and user updated
Bump Minor	MINOR + 1, PATCH = 0 (e.g. 1.0.5 → 1.1.0)
Bump Major	MAJOR + 1, MINOR = PATCH = 0 (e.g. 1.2.5 → 2.0.0)

The system tags `_System.ProjectVersion` and `_System.DeployCount` carry the version and build number, so they can be shown on a screen or written to reports and batch records.

5.4.8 Deleting a project

WARNING: Delete removes the project folder with its database, historian archive and backups permanently. It cannot be undone. Export the project first if you may need it again.

1. Switch to another project if the one to delete is active (the active project and Default cannot be deleted).
2. Click **Delete** on the row and confirm "This will permanently delete the database and all configuration files for this project. This cannot be undone!".

5.5 Staging and deployment

5.5.1 Principle

With staging, configuration changes are not applied to the running system when you save them. They are collected as **pending changes** and applied together when an administrator **deploys** them. This allows an engineer to prepare a set of related changes (for example new tags, their alarms and the screen that shows them), check them, and put them into operation at one chosen moment — and to roll the set back if necessary.

Staging is on by default (system setting `staging_enabled` = `true`). Version 2.6.1.26 has no switch for it in the user interface.

5.5.2 What is staged

Staged (appears in Pending Changes)	Applied immediately (not staged)
Tags (create, update, delete)	PLC connections
Alarm configurations	Users and roles
Synoptic screens and popups	Recipes
Super objects and super object instances	Datalogs and data sources
UDT definitions and UDT instances	Notification settings
Charts	Display, layout and security settings
Merge import (5.4.6) and bulk imports that the importer marks as direct	

While a change is pending, the configuration pages show the item with a marker such as **PENDING CREATE**, **PENDING UPDATE**, **UPDATE PENDING** or **DELETE PENDING**, so you can see what the running system does not have yet.

Note: The licence tag limit is checked only when a tag is created directly. Tags created through staging are not counted against the limit until they exist (Appendix C).

5.5.3 Where pending changes are shown

Place	Who sees it	Content
Pending Deploy indicator in the header	Administrators	Count of pending changes; pulses when 5 or more. Click to open the staging panel.
Floating Deploy button	Administrators, in Config	Count; click to open the staging panel
Deployment tab in Config	Administrators	Blinks orange with a red count badge while changes are pending
Viewer Builder → Deploy Changes	Administrators	The same list, reachable from the screen builder

The staging panel (**Staging & Deployment**) has two tabs:

- **Pending Changes (n)** — changes grouped by type (Tag, Screen, Alarm Config ...), each with an operation badge (+ create, ~ update, - delete), the entity name, the operation, "by <user> at <time>" and a × discard button; below them **Deployment description (optional)**, **Discard All** and **Deploy n Change(s)**.
- **Deploy History** — past deployments with status, count, user, time and description, and **Rollback**.

The **Deployment** → **Staging** tab in **Config** shows the same information as **Deploy Configuration Changes**, with **Deploy Description (Optional)**, **Pending Changes (n)**, **Deploy History**, **Discard All** and **Deploy n Changes**.

For large imports, each list shows only the first 200 changes ("Showing first 200 of n pending changes"); deploying or discarding still acts on all of them.

5.5.4 Deploying

To deploy the pending changes:

1. Open the staging panel or **Config → Deployment**.
2. Review the list. Discard any change you do not want with **×** / **Discard** (confirm "Are you sure you want to discard this change?").
3. Type a description, for example `Line 1: added Tank_101 level alarms`.
4. Click **Deploy n Change(s)** and confirm "Deploy n pending change(s)? This will apply all changes to the system."
5. The changes are applied in the order they were made. More than 20 changes are applied in a single database transaction.
6. The tag table in memory is reloaded, the project version is increased (5.4.7), the deployment is recorded in the event log, and every connected client is notified.

The result is **success**, **partial** (some changes failed — the message "Deployed n changes with m errors" lists them) or **failed**.

WARNING: Deploying changes the running system at once: new tags start polling, changed addresses are read from the new location, new alarms are evaluated and deleted tags disappear from every screen. Deploy at a moment agreed with the operators, and check the affected screens immediately afterwards.

5.5.5 Discarding and rolling back

Action	Effect
× / Discard on one change	The change is dropped; the live configuration is unchanged
Discard All	All pending changes are dropped (confirm "...This cannot be undone.")
Rollback on a deployment in Deploy History	Reverses every change of that deployment in reverse order: created items are deleted, updated items get their previous values back, deleted items are restored. The deployment is then marked rolled_back .

Rollback is offered only for deployments with status **success**. A **partial** deployment cannot be rolled back from the user interface; correct it with new changes. A deployment can be rolled back only once.

Important: A rollback applies immediately to the running system; it does not create pending changes and does not change the project version.

5.5.6 What operators see after a deploy

Every open client shows **New Configuration Deployed** — "A new version of the system configuration has been deployed. (n changes) Would you like to refresh to get the latest version?"

- **Refresh Now** reloads the page with the new configuration (screens, tags, alarms).
- **Later** keeps the current page. Live values continue, but new or changed screens appear only after a reload.

Operators should refresh when the current operation allows it.

5.6 Server deployment (multi-server)

Enterprise installations can pass a configuration from a test server to a production server and keep a backup server, a dedicated historian server and a batch server in step. The functions are in **Config → Admin → Multi-**

Server and require the Enterprise edition or the **multi-server** licence feature; otherwise the page shows **Enterprise License Required**.

5.6.1 Topologies and server roles

Deployment Topology	Servers	Roles available
Standalone	1	Standalone
Production + Backup	2	Production Server, Backup Server
Test + Prod + Backup	3	Test Server (Development), Production Server, Backup Server
Full (5-Server)	5	Test, Production, Backup, Historian Server, Batch Control Server

Under **Server Configuration**, click **Edit**, choose **This Server Role**: and fill in a card for each partner server (**Test Server**, **Production Server**, **Backup Server**, **Historian Server**, **Batch Control Server**): enable switch, **Name**:, **URL**:, and **API Key**: (**Generate** creates a key). The receiving server must have the same key configured for the sending server's role. Click **Save**. **Server Status** shows the health of each server (**Refresh**).

5.6.2 Deploying to production

On a Test or Standalone server with a production server enabled:

1. Under **Deployment Actions**, click **Deploy to Production**.
2. In **Select Items to Deploy**:, tick **Screens**, **Tags**, **Alarms**, **PLC Connections**, **Recipes**, **Super Objects** and/or **Users (metadata only)**. Each shows the number of items.
3. Read the warning "This will update the production server configuration." (and "Backup server will also be updated automatically." when configured).
4. Click **Deploy Now**.

The package is sent to the production server's `/api/deployment/receive` endpoint with the API key and applied there. **Deployment History** lists each operation (**Date/Time**, **Type**, **Deployed By**, **Targets**, **Status**). **Sync Backup Server**, **Sync Historian Server** and **Sync Batch Server** push the current configuration to those servers.

Important: In version 2.6.1.26 the receiving server applies tags, alarm configurations and PLC connections from the package. Screens, recipes, super objects and users in the package are not applied correctly. Transfer those by exporting the project and using **Merge Import** (5.4.6) on the target, and verify the target after every deployment.

WARNING: Deploying to production overwrites existing tags, alarms and PLC connections with the same identity on the production server, outside its staging, and takes effect immediately.

5.6.3 Multi-Site Dashboard

Config → **Deployment** also contains the **Multi-Site Dashboard**, which connects to other OptiZeus servers for an overview (online state, tag count, active alarms). **+ Add Site** asks for a site name and URL; **Refresh Sites** loads the status; each site card has **Open** and **Remove**. Multi-site operation is described in Chapter 27.

5.7 Worked example: taking a change into operation

Goal: add the level alarms and screen of Tank_101 (Chapter 3) to the running project **Line1** with minimum risk.

1. **Backup.** Create a project backup (Chapter 26) and export **Line1** (5.4.3).
2. **Prepare.** In **Line1**, create the tags, alarms and screen. They stay pending; the plant keeps running on the old configuration. The header shows **Pending Deploy** with the count.
3. **Review.** Open **Config** → **Deployment**. Check that only the intended items are listed; discard anything else.
4. **Agree.** Inform the shift supervisor. Choose a moment without critical operations.
5. **Deploy.** Enter the description **Tank_101 level alarms and transfer screen** and click **Deploy**.
6. **Verify.** Open the new screen, check that values are live (no **xxxx**), and check the alarm limits in **Config** → **Alarms**. Ask the operators to click **Refresh Now**.
7. **If something is wrong.** Click **Rollback** on the deployment in **Deploy History**, or correct the item and deploy again.
8. **Record.** The deployment, its user and description are in the deploy history and the event log; the project version shows the new patch number.

6. PLC and device connections

This chapter describes how OptiZeus communicates with PLCs, controllers and field devices: the **PLC Connections** page, the connection dialog, every protocol driver that is implemented in version 2.6.1.26, the address syntax of each driver, how often values are read, how writes behave, and how to diagnose a connection. It is written for the system integrator and automation engineer who configures a project. Supervisors will find the status and diagnostics section (6.15) useful when a plant area shows bad quality.

Tags that use a connection are described in Chapter 7. Alarms raised when a connection is lost are described in Chapter 17. The OPC UA *server* (OptiZeus exposing its own tags to other systems) is a separate function and is described in Chapter 8.

6.1 Concepts

Term	Meaning
Connection	One configured link to one controller or device (one row in the PLC Connections list). A connection has a name, a brand, a model, a protocol and protocol-specific parameters.
Driver	The communication code for one protocol, running inside the OptiZeus backend service.
Scan	One pass of the backend acquisition loop. It reads every tag whose connection is enabled and then evaluates calculated tags and alarms.
Quality	Every tag carries a quality: <code>good</code> , <code>bad</code> or <code>unknown</code> (not yet read). Screens show a bad-quality value as a placeholder (for example <code>XXXX</code>) instead of a stale number.
Server ID	Internal identifier of a connection, <code>plc_<id></code> , where <code><id></code> is the database row number. You see it in logs and on the OPC UA status panel.

Tags refer to a connection **by its name** (the tag's **PLC Connection** field stores the connection name). If you rename a connection, the tags that refer to the old name lose their binding. Rename a connection only before you create its tags, or re-assign the tags afterwards (Chapter 7, bulk CSV import).

6.2 Supported protocols

The table lists what the backend really implements. Use it before you promise a protocol to a customer.

Protocol (as shown in the dialog)	Internal key	Read	Write	Status in 2.6.1.26
OPC UA	<code>opcua</code>	Yes (subscription + batched reads)	Yes	Full driver. Recommended for any controller that has an OPC UA server.
Siemens S7 (ISO-on-TCP)	<code>s7</code>	Yes (block-optimised)	Yes	Full driver (native S7 communication via the snap7 library).
Modbus TCP	<code>modbus</code>	Yes (block-optimised)	Yes	Full driver, 16-bit registers and coils only (see 6.7.2).

Protocol (as shown in the dialog)	Internal key	Read	Write	Status in 2.6.1.26
EtherNet/IP (CIP)	ethernetip	Yes (worker thread, multi-connection)	Yes, with a restriction (see 6.8.5)	Allen-Bradley Logix-family symbolic tags (ControlLogix, CompactLogix, Micro800 and other CIP devices that expose symbolic tags).
PCCC over Ethernet/IP	pccc	Yes	Yes	Allen-Bradley MicroLogix / SLC 500 data-table addresses (N7:0 , F8:1 , B3:0/5 ...).
MQTT	mqtt	Yes (subscribe)	Yes (publish)	Generic MQTT broker client, plain or JSON payloads.
BACnet/IP	bacnet	Yes	Yes	ReadProperty / WriteProperty on standard object types.
GE SRTP (SNP-X)	srtip	Yes	Yes	GE / Emerson PACSystems and Series 90 memory areas (%R , %AI , %Q ...).
MELSEC Communication	melsec	No	No	Listed for Mitsubishi models. Only Test Connection (TCP port check) is implemented. No data is exchanged.
Omron FINS	fins	No	No	Listed for Omron models. Only the TCP port check is implemented.
Beckhoff ADS	ads	No	No	Listed for Beckhoff models. Only the TCP port check is implemented.

Important: MELSEC, FINS and ADS appear in the protocol list but have no data driver in this version. A connection with one of these protocols can be saved, tested (the test only opens a TCP socket) and even shows as "connected" after you click **Connect**, but its tags are never read and stay at bad quality. For Mitsubishi, Omron and Beckhoff controllers use the controller's OPC UA server or Modbus TCP instead (both are offered in the same model list where the controller supports them).

Note: **Modbus RTU** (serial) is not available. The dialog has no serial-port fields and the backend opens Modbus connections over TCP only. To reach serial Modbus devices, use a Modbus TCP-to-RTU gateway and configure a Modbus TCP connection to the gateway, one connection per slave (Unit ID).

Zeus PLC controllers. OptiZeus has no dedicated Zeus PLC driver. Connect a Zeus PLC through one of its standard interfaces: select brand **Generic / Other**, model **OPC UA Server**, and enter the controller's OPC UA endpoint; or, where the controller's Ethernet/IP tag server is enabled, model **EtherNet/IP Device**. Which interfaces a given Zeus PLC firmware exposes is described in the Zeus PLC documentation.

MQTT Sparkplug B is a separate, *outbound* function: OptiZeus publishes its tags to a broker as a Sparkplug B edge node. It is not a PLC connection. See Section 6.16 and Chapter 28.

6.3 Licence limits and permissions

The number of connections is limited by the licence tier (Appendix C):

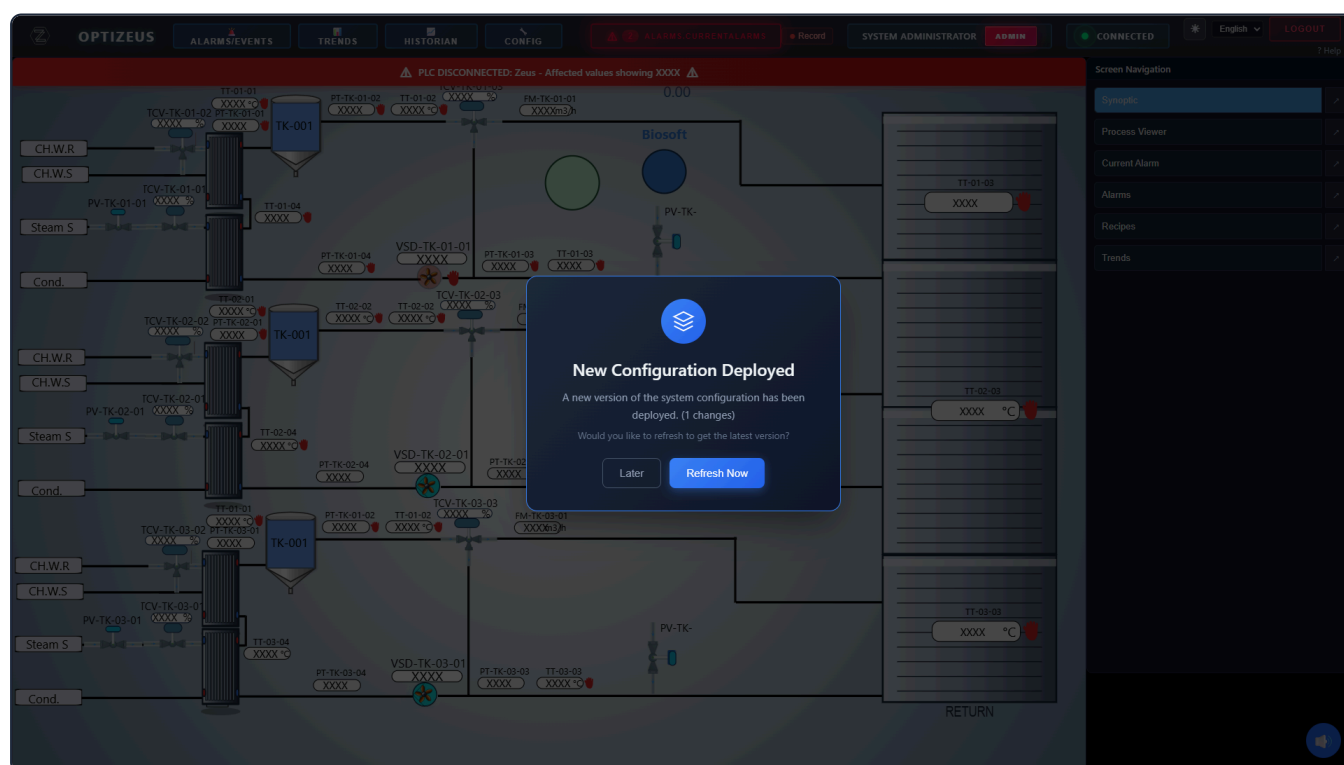
Tier	Maximum PLC connections
Starter	1
Basic	3
Professional / Professional Max	8
Enterprise	20

When the limit is reached, **Add Connection** fails with "PLC connection limit reached (n/max). Please upgrade your license to add more PLC connections." Every row in the list counts, including disabled connections.

Operation	Roles allowed by the server
View the connection list and status	Any authenticated user
Add, edit, delete a connection	admin
Test Connection, Refresh Tags	admin
Connect, Disconnect	operator , admin
Browse a controller (tag browser)	Any authenticated user

Note — changed in 2.6.1.24: Earlier versions allowed **Test Connection, Connect, Disconnect** and **Refresh Tags** to any logged-in user. A viewer is now refused (*Insufficient privileges*). Operators may still connect and disconnect a controller; hide the Configuration view from non-administrators (Chapter 14 and Chapter 24) if operators must not do so.

6.4 The PLC Connections page



PLC connections page

To open it, click **Config** in the header and select the **PLC Connections** tab. The page contains:

- **+ Add PLC Connection** — opens the connection dialog (Section 6.5).
- A search box, *Search connections by name, IP, protocol...*, that filters the list as you type.
- The connection list, grouped by brand. Click a brand header to collapse or expand it. While you search, all groups are expanded.

Each connection row shows:

Element	Meaning
Status dot	Green = connected, red = disconnected or failed, grey/amber = connecting or unknown. A disabled connection always shows as disconnected.
Name	The connection name. A yellow PENDING CREATE / UPDATE / DELETE badge means the change is staged and not yet deployed (Chapter 5).
 protocol,  model,  address	The protocol name, the model, and <code>ip:port</code> (for OPC UA, the endpoint URL). S7 rows also show <i>Rack r/Slot s</i> ; Modbus rows show <i>Unit n</i> .
Scan rate	The stored scan rate value (see 6.11.1 — this value is informational in 2.6.1.26).
 Connect / Disconnect	Opens or closes the driver connection immediately.
Refresh Tags	EtherNet/IP rows only. Clears the "bad tag" cache of the EtherNet/IP worker and re-scans all tags (6.8.3).
 Edit	Opens the dialog for this connection.
	Deletes the connection.

WARNING: Deleting a connection also deletes **every tag bound to it** and every alarm configuration of those tags. The confirmation box says so; the result box reports how many tags and alarm configurations were removed. There is no undo other than restoring a backup (Chapter 26). Take a backup before you delete a connection on a running plant.

When you create a connection, OptiZeus automatically creates a **communication alarm configuration** for it, so that loss of the connection raises an alarm (Chapter 17).

6.5 The connection dialog

To add a connection:

1. Click **+ Add PLC Connection**.
2. Enter a **Connection Name**. Use a short, stable, unique name (for example `PLC_Line1`). Tags reference this name.
3. Select the **PLC Brand**, then the **Model**. The model selects a default protocol and port.
4. Check the **Protocol**. Only protocols offered for that model are listed.
5. Fill in the address fields (**Host / IP Address** and **Port**, or **Endpoint URL** for OPC UA) and the protocol-specific fields described in 6.5.2.
6. Click **Test Connection** and read the result line under the form.
7. Leave **Enabled** checked and click **Add Connection**.

To change a connection, click **Edit**, change the fields and click **Update Connection**. **Cancel** closes the dialog without saving.

When you edit a connection that uses username/password authentication (OPC UA or MQTT), the **Password** field opens empty. Leave it empty to **keep** the stored password; type a new password only to change it. (A masked value `*****` is also treated as "keep".) To remove a stored password through the REST API, send `"clearPassword": true` with `PUT /api/plc-connections/<id>`.

Note — changed in 2.6.1.26: Up to 2.6.1.25 an empty **Password** field was saved as entered, so every edit of an OPC UA or MQTT connection cleared the stored password unless it was typed again, and the next connection attempt failed authentication.

6.5.1 Common fields

Field	Meaning	Default / range
Connection Name *	Unique name. Referenced by tags.	—
PLC Brand *	Siemens, Allen-Bradley (Rockwell), Schneider Electric, Mitsubishi Electric, Omron, Beckhoff, ABB, Delta Electronics, Keyence, Unitronics, WAGO, Phoenix Contact, GE Intelligent Platforms, Generic / Other.	—
Model *	Model family; defines the protocols offered and the default port (table 6.5.3).	—
Protocol *	Driver to use (Section 6.2). Changing the protocol resets Port to the protocol's default.	First protocol of the model
Host / IP Address *	Controller or broker address. Hidden for OPC UA.	—
Port *	TCP (or UDP for BACnet) port. Hidden for OPC UA.	Protocol default
Enabled	When cleared, the connection is not polled, its tags are skipped and it raises no disconnection alarm.	Checked

The following values are stored with every connection but are **not shown** in the dialog; new connections always get the defaults: timeout 5000 ms, retry count 3, scan rate 1000 ms.

6.5.2 Protocol-specific fields

Protocol	Field	Meaning	Default / range
OPC UA	Endpoint URL *	Full endpoint, must start with <code>opc.tcp://</code> , for example <code>opc.tcp://192.168.1.10:4840</code> .	—
OPC UA	Enable Username/Password Authentication	Unchecked = anonymous session.	Unchecked
OPC UA	Username, Password	Credentials for the session.	—
OPC UA	Subscribe to remote alarms (OPC UA Alarms & Conditions)	Forwards alarms raised on the remote OPC UA server into OptiZeus (External Alarm Bridge, Chapter 17).	Unchecked

Protocol	Field	Meaning	Default / range
OPC UA	Severity Min, Severity Max	Ignore remote events outside this OPC UA severity range.	0 / 1000
OPC UA	Default Area, Default Group	Area and group given to forwarded alarms.	Empty
Modbus TCP	Unit ID (Slave Address)	Modbus unit identifier.	1 (1–247)
Siemens S7	Rack, Slot	Position of the CPU. S7-1200/1500: rack 0, slot 0 or 1 (per hardware configuration); S7-300: rack 0, slot 2.	0 / 0 (rack 0–7, slot 0–31)
EtherNet/IP	Slot	Slot of the Logix CPU in the chassis (0 for CompactLogix and Micro800).	0 (0–30)
EtherNet/IP	Parallel Connections	Number of simultaneous CIP connections the worker opens to this PLC: 1 (Safe - slower), 2 (Default), 3 (Fast), 4 (Fastest - requires strong PLC), 6 (Maximum) .	2
MQTT	Client ID	MQTT client identifier. Leave empty for <code>scada_<id></code> . Must be unique on the broker.	Empty
MQTT	QoS Level	QoS used for subscriptions and publishes: 0 - At most once, 1 - At least once, 2 - Exactly once .	0
MQTT	Use TLS (port 8883)	Connects with <code>mqtts://</code> . Checking it sets Port to 8883.	Unchecked
MQTT	Enable Username/Password Authentication, Username, Password	Broker credentials.	—
BACnet/IP	Device Instance	BACnet device object instance of the target device.	0 (0–4194303)

Note: The OPC UA dialog has no security-mode or certificate field. In 2.6.1.26 OptiZeus always opens OPC UA client sessions with **security mode None** and security policy None, with either anonymous or username/password identity. Use OPC UA client connections on a protected control network (Chapter 25), or place the controller behind a firewall that only admits the OptiZeus server.

6.5.3 Brands, models and protocols

Brand	Model	Protocols offered (first = default)	Default port
Siemens	S7-1500, S7-1200	S7, OPC UA, Modbus TCP	102
Siemens	S7-300/400	S7, Modbus TCP	102
Siemens	LOGO!	Modbus TCP	502
Allen-Bradley (Rockwell)	ControlLogix, CompactLogix	EtherNet/IP, OPC UA	44818
Allen-Bradley (Rockwell)	Micro800	EtherNet/IP, Modbus TCP	44818

Brand	Model	Protocols offered (first = default)	Default port
Allen-Bradley (Rockwell)	MicroLogix	PCCC over EtherNet/IP, EtherNet/IP, Modbus TCP	44818
Schneider Electric	Modicon M340	Modbus TCP, OPC UA, BACnet/IP	502
Schneider Electric	Modicon M580	Modbus TCP, OPC UA, EtherNet/IP, BACnet/IP	502
Schneider Electric	Premium, Quantum	Modbus TCP	502
Mitsubishi Electric	iQ-R	MELSEC*, OPC UA	5007
Mitsubishi Electric	iQ-F, FX Series	MELSEC*, Modbus TCP	5007
Mitsubishi Electric	Q Series	MELSEC*	5007
Omron	NJ, NX	FINS*, EtherNet/IP, OPC UA	9600
Omron	CJ, CP	FINS*, Modbus TCP	9600
Beckhoff	CX Series, Embedded PC	ADS*, OPC UA, Modbus TCP	48898
ABB	AC500, PM5xx	Modbus TCP, OPC UA, BACnet/IP	502
Delta Electronics	AS Series	Modbus TCP, EtherNet/IP	502
Delta Electronics	AH Series, DVP Series	Modbus TCP	502
Keyence	KV-8000	EtherNet/IP, Modbus TCP, OPC UA	44818
Keyence	KV-7000	EtherNet/IP, Modbus TCP	44818
Unitronics	UniStream	Modbus TCP, OPC UA	502
Unitronics	Vision Series	Modbus TCP	502
WAGO	PFC200	Modbus TCP, OPC UA	502
WAGO	PFC100	Modbus TCP	502
Phoenix Contact	PLCnext	OPC UA, Modbus TCP	4840
Phoenix Contact	ILC Series	Modbus TCP	502
GE Intelligent Platforms	PACSystems RX3i, RX7i	GE SRTP, Modbus TCP, OPC UA	18245

Brand	Model	Protocols offered (first = default)	Default port
GE Intelligent Platforms	Series 90-30, 90-70, VersaMax	GE SRTP, Modbus TCP	18245
Generic / Other	Modbus TCP Device / OPC UA Server / EtherNet/IP Device / MQTT Broker / BACnet/IP Device / GE SRTP Device	the single named protocol	502 / 4840 / 44818 / 1883 / 47808 / 18245

* No data driver in 2.6.1.26 (see 6.2). Select one of the other protocols for that model.

Tip: The brand and model are used for grouping, for defaults and, for Siemens S7, for tuning (the model decides the S7 connection type and the read block size, see 6.6). Choose the model that matches the hardware even if you use a generic protocol.

6.5.4 Test Connection

Test Connection checks the parameters without saving. What it really tests:

Protocol	Test performed	Success message (example)
OPC UA	Opens a secure channel to the endpoint (no session). Rejects endpoints that do not start with <code>opc.tcp://</code> .	<i>Successfully connected to OPC UA server at <code>opc.tcp://...</code></i>
Modbus TCP	Opens TCP, then tries to read holding register 0. A failed register read still reports success, with a hint.	<i>...and verified register read or TCP connection ... succeeded. Register read was not possible - verify slave ID and register configuration.</i>
Siemens S7	Full S7 connect with rack and slot.	<i>Successfully connected to S7 PLC at ... (Rack: r, Slot: s)</i>
EtherNet/IP	CIP connect to the slot.	<i>Successfully connected to EtherNet/IP controller at ...</i>
PCCC	Registers an EtherNet/IP session and tries to read <code>N7:0</code> .	<i>Successfully connected to MicroLogix at ...:44818 — N7:0 = 12</i>
MQTT	Connects to the broker with the given credentials (TLS certificate not verified during the test).	<i>Successfully connected to MQTT broker at <code>mqtt://...</code></i>
BACnet/IP	Reads the <code>objectName</code> property of the device object.	<i>Successfully connected to BACnet device at ...</i>
GE SRTP	TCP connect, SRTP initialisation, test read of the first <code>%R</code> register.	<i>...verified register read (%R0)</i>
MELSEC, FINS, ADS, others	Opens a TCP socket only.	<i>Successfully connected to ...</i>

The test timeout is 5 seconds.

6.6 Siemens S7

6.6.1 Controller preparation

OptiZeus uses absolute addressing through the S7 communication protocol (port 102). On S7-1200 and S7-1500 CPUs:

- enable **Permit access with PUT/GET communication from remote partner** in the CPU protection settings;
- for every data block that OptiZeus reads or writes, clear **Optimized block access** so that absolute offsets (`DBX` , `DBW` , `DBD`) exist.

If you cannot change the PLC program, use the CPU's OPC UA server (S7-1500, S7-1200 firmware 4.4 and later) and an OPC UA connection instead.

For models whose name contains `1200` , `1500` or `LOGO` , OptiZeus uses the S7 "basic" connection type automatically. S7-300/400 use the default PG connection type.

6.6.2 Address syntax

Addresses are not case-sensitive. Spaces are ignored. `E / A` (German mnemonics) are accepted for inputs and outputs.

Address	Area	Read as	Example
<code>DB<n>.DBX<byte>.<bit></code>	Data block bit	Bit (0/1)	<code>DB10.DBX4.0</code>
<code>DB<n>.DBB<byte></code>	Data block byte	Unsigned 8-bit	<code>DB10.DBB2</code>
<code>DB<n>.DBW<byte></code>	Data block word	Signed 16-bit INT	<code>DB10.DBW6</code>
<code>DB<n>.DBD<byte></code>	Data block double word	32-bit REAL	<code>DB10.DBD8</code>
<code>M<byte>.<bit></code> / <code>MB</code> / <code>MW</code> / <code>MD<byte></code>	Memory (flags)	Bit / byte / INT / REAL	<code>M20.1</code> , <code>MW100</code> , <code>MD200</code>
<code>I<byte>.<bit></code> / <code>IB</code> / <code>IW</code> / <code>ID<byte></code>	Process inputs	Bit / byte / INT / REAL	<code>I0.3</code> , <code>IW64</code>
<code>Q<byte>.<bit></code> / <code>QB</code> / <code>QW</code> / <code>QD<byte></code>	Process outputs	Bit / byte / INT / REAL	<code>Q4.0</code> , <code>QW80</code>

All multi-byte values are big-endian (S7 native order).

Important: A double-word address (`DBD` , `MD` , `ID` , `QD`) is always interpreted as a 32-bit **REAL**. The tag dialog has no field for DINT or DWORD, so a DINT counter stored in `DB10.DBD12` is displayed as a meaningless floating-point number. Store integer values the HMI must read as INT (16-bit) words, or copy them into a REAL in the PLC. Timers, counters (`T` , `C`), strings and peripheral areas are not supported.

To read a single bit of a word, you can also use a tag of type **Bit** — but for S7 the simplest method is the `DBX / M / I / Q` bit address.

6.6.3 Optimised block reads

OptiZeus does not read S7 tags one at a time. In every scan it:

1. groups all tags of one connection by memory area and DB number;
2. computes the smallest byte range that covers all tags of the group (lowest start byte to highest end byte) and reads that range in one request;

3. splits ranges that are larger than one PDU into several requests: 222 bytes for S7-1200 and LOGO!, 440 for S7-300/400, 880 for S7-1500, 960 for other models;
4. waits 15 ms between requests on S7-1200/LOGO! so the small CPUs are not overloaded;
5. extracts every tag value from the returned buffer.

If a block read fails (typically because the DB is shorter than the highest configured address), the driver retries that group tag-by-tag so that valid tags still update. One read request times out after 5 seconds.

Tip: Keep the tags of one DB close together. A single tag at byte 2000 of a DB whose other tags are at bytes 0–40 forces OptiZeus to read the whole 2 KB range every scan.

6.6.4 Errors and reconnection

A failed block read marks the tags of that block bad. The connection is closed only after **5 consecutive** failed reads, so a single timeout does not cause a reconnect storm. A closed connection is re-opened automatically at most once every 30 seconds (6.15.1).

6.7 Modbus TCP

6.7.1 Address syntax

Address form	Register type	Function used for reading	Offset used on the wire
HR<n>	Holding register	FC03	n (0-based)
IR<n>	Input register	FC04	n (0-based)
C<n>	Coil	FC01	n (0-based)
DI<n>	Discrete input	FC02	n (0-based)
4xxxx (5 digits)	Holding register	FC03	xxxx - 1 (40001 → 0)
3xxxx (5 digits)	Input register	FC04	30001 → 0
0xxxx (5 digits)	Coil	FC01	00001 → 0
1xxxx (5 digits)	Discrete input	FC02	10001 → 0
<n> (any other number)	Holding register	FC03	n (0-based)

Examples: 40001 and HR0 address the same register. C12 is coil 13 in one-based device documentation.

6.7.2 Data representation

Each Modbus analog tag reads **one 16-bit register**, returned as an unsigned value 0–65535. A register that holds a negative number (for example –5) is shown as 65531.

- **Bit tags** (tag type **Bit**, Chapter 7) extract one bit from the register. With **Word Size** DINT two consecutive registers are read, the first as the high word.
- **Coils and discrete inputs** give 0 or 1.
- **Linear scaling** (Chapter 7) converts the raw register value into engineering units, for example raw 0–27648 → 0–100 %.

Important: 32-bit integers, 32-bit floating-point values (two registers), byte-swapped layouts and signed 16-bit interpretation are **not supported** by the Modbus driver in 2.6.1.26. If the device only provides floats, configure the device (or a gateway) to publish scaled 16-bit integers, or use OPC UA.

6.7.3 Block reads

Tags of one connection are grouped by register type, sorted by address and merged into blocks: a new block starts when the gap to the previous tag exceeds 10 registers or the block would exceed 120 registers. Each block is one Modbus request. If a block fails, only its tags turn bad. The connection counts as connected in a scan as long as at least one block succeeded.

6.7.4 Writes

Tag address	Write performed
HR<n> , 4xxxx , plain number	FC06 Write Single Register with the (reverse-scaled, rounded) value
C<n> , 0xxxxx	FC05 Write Single Coil (non-zero = ON)
Tag type Bit on a holding register	Read-modify-write: the register is read, one bit set or cleared, the register written back
IR<n> , DI<n> , I<n> , 3xxxx , 1xxxx	Refused — input registers and discrete inputs are read-only in Modbus. Nothing is sent to the device.
Any other address form	Refused — not a coil or holding register. Nothing is sent to the device.

A refused write is handled like any write that does not reach the controller (6.11.3): the tag is not changed, the screen shows **Write Failed!** (*Not written: the PLC connection ... is not connected or rejected the write.*), the event log records `tag_write_failed`, and the server console shows the exact reason, for example *Modbus address IR5 is an input (read-only) and cannot be written*. Do not place writable controls (setpoint entry, buttons) on tags that address inputs.

Note — changed in 2.6.1.26: Up to 2.6.1.25 input addresses were not rejected: a write to a tag addressed IR5 or DI3 was sent to **holding register 0**, and a write to 30001 to holding register 30001, which could change an unrelated value in the device. After upgrading, check that no writable control is bound to an input address — such writes now fail.

The read-modify-write of a Bit tag is not atomic: if the PLC changes another bit of the same register between the read and the write, that change is overwritten.

6.8 EtherNet/IP (Logix symbolic tags)

6.8.1 How the driver works

EtherNet/IP tags are read by a dedicated **EIP worker thread**, separate from the main acquisition loop, so a slow Logix controller never delays other connections. For each PLC the worker:

1. opens the number of CIP connections set in **Parallel Connections** (the first must succeed; the others are optional);

2. on first connection, reads every configured tag **individually** to find out which addresses exist ("scan"); addresses that fail with a non-connection error are put in a **bad-tag cache** and skipped from then on;
3. then reads the valid tags in a continuous loop with CIP Multiple Service requests, splitting the tag list evenly across the parallel connections, pausing 50 ms between cycles;
4. if a batch fails for a non-connection reason, falls back to individual reads for that part.

If two consecutive cycles time out while more than one connection is open, the worker halves the number of connections (for example 4 → 2) and re-scans. When the PLC stops answering, all its tags are set to bad quality and the worker retries every 5 seconds.

6.8.2 Address syntax

Use the Logix tag name exactly as in the controller program.

Address	Meaning
Tank_Level	Controller-scoped tag
Program:MainProgram.Pump_Speed	Program-scoped tag
Motor_1.Running	Member of a structure / UDT
Recipe[3]	Array element
Matrix[1,2]	Element of a two-dimensional array

BOOL values are shown as 0/1. SINT/INT/DINT/REAL/LREAL are shown as numbers.

6.8.3 Refresh Tags and the bad-tag cache

If you correct a tag address, or the controller program was downloaded with new tags, click **Refresh Tags** on the connection row. The worker clears its bad-tag cache and scans all tags again. Adding a new EtherNet/IP tag through the tag dialog updates the worker automatically after about 2 seconds.

Note: In 2.6.1.26 the **Refresh Tags** button sends the worker a tag list in a format the worker does not expect. After pressing it, tags of that PLC may stop updating until the tag list is re-sent (edit and save any tag of that PLC, or restart the OptiZeus service). Until this is corrected, prefer correcting the tag and saving it, which refreshes the worker correctly.

6.8.4 Browsing controller tags

In the tag dialog (Chapter 7) select the EtherNet/IP connection and click **Browse...** next to **PLC Address**. OptiZeus reads the controller's symbol table and shows the controller name and every **controller-scoped** tag with its data type (BOOL, SINT, INT, DINT, LINT, USINT, UINT, UDINT, REAL, LREAL, STRING, STRUCT) and an array flag. Type in *Filter tags...* to narrow the list, then click a tag to copy its name into the address field. Program-scoped tags and structure members are not listed; type them manually.

6.8.5 Writes

Writes use a separate connection of the main backend process, not the worker. That connection exists after you click ► **Connect** on the connection row (or after the backend connected it before the worker started). If this connection is not open, a write from a screen fails with *"Not written: the PLC connection "<name>" is not connected or rejected the write."* (Section 6.11.3) — click ► **Connect** on the connection and write again.

Important: Verify every EtherNet/IP write path during commissioning (write a test value and read it back in the controller programming software). Since 2.6.1.26, writes started by automation actions, recipe downloads, alarm pipelines and weekly schedules (Chapters 9, 17, 20) use the same write path as an operator write and reach EtherNet/IP connections as well; up to 2.6.1.25 automation writes were not delivered to EtherNet/IP. The batch engine's own PLI writes still use a separate routine that does not reach EtherNet/IP (Chapter 20, Section 20.9.2).

6.9 PCCC over EtherNet/IP (MicroLogix, SLC 500)

Choose **PCCC over EtherNet/IP** for MicroLogix 1100/1400 and SLC 5/05 controllers, which use data-table addresses instead of symbolic tags.

6.9.1 Address syntax

```
<file type><file number>:<element>[/<bit>][.<sub-element>]
```

File type	Meaning	Default file	Example
O	Outputs	0	O:0/2
I	Inputs	1	I:0/5
S	Status	2	S:1/15
B	Binary (bits)	3	B3:0/5
T	Timer	4	T4:0.ACC , T4:0.DN
C	Counter	5	C5:2.PRE , C5:2.DN
R	Control	6	R6:0.POS
N	Integer	7	N7:10
F	Float	8	F8:3
ST	String	10	ST10:0
L	Long integer	11	L11:0

Timer sub-elements: PRE , ACC , EN , TT , DN . Counter: PRE , ACC , CU , CD , DN , OV , UN . Control: LEN , POS , EN , EU , DN , EM , ER . If the file number is omitted, the default file number is used.

6.9.2 Behaviour

PCCC tags are read by the main acquisition loop, one request per tag, over **one shared TCP connection per controller** (MicroLogix controllers accept only a few TCP connections). The connection is kept open between scans and closed 30 seconds after its last use. A read error caused by a wrong address marks only that tag bad; three connection-level errors in one scan skip the remaining tags of that PLC for the rest of the scan. Because reads are sequential, keep PCCC tag counts moderate (a few hundred per controller).

6.10 OPC UA client

6.10.1 Connection and session

- One OPC UA client and session are opened per endpoint. If two connections point to the **same endpoint URL**, the second reuses the first one's session instead of opening another (some controllers limit the number of sessions).
- Session timeout 10 minutes, keep-alive enabled, automatic reconnection by the OPC UA stack.
- A **dedicated write session** is opened next to the read session, so operator writes never wait behind the heavy subscription traffic. If the server refuses a second session, writes share the read session.
- Automatic connection attempts made by the scan are limited to 10 seconds each.

Typical connection errors are translated into readable messages, for example *"Connection refused - OPC UA server may not be running on ..."*, *"Security mode rejected - Try changing the security settings or enable anonymous access on the OPC UA server."*, *"Authentication failed - Check username and password..."*.

6.10.2 How values are acquired

Mechanism	Detail
Subscription	One subscription per connection, publishing interval 250 ms, sampling interval 100 ms, queue size 10, keep-alive after 12 publishing intervals. Every tag becomes a monitored item; value changes are pushed into a local cache and the scan reads from that cache.
Subscription watchdog	Checks every second. If a subscription has delivered no data change for 5 s (after a 30 s grace period following its creation), it is deleted and re-created with all its monitored items. It is re-created at most once every 15 s.
Fallback poll	For servers that accept a subscription but stop pushing changes, a background poll actively reads the monitored nodes of any server that has been silent for more than 350 ms. Servers are polled one after the other, 100 nodes per read, with a 5 s pause after a failed read.
Batched reads	When no subscription exists, tags are read with multi-node Read requests of 100 nodes. The first read probes whether the server supports multi-node reads; if not, the connection falls back to one-node reads.
Secondary connections	A connection that shares an endpoint with another connection has no subscription; its tags are re-read at most once per second.

Three-strike rule. A read error whose message indicates a broken session (closed session, connection reset, timeout ...) increments an error counter. Only after **3 consecutive** such errors is the session torn down and re-opened. A successful read resets the counter. This prevents a single transient error from costing a multi-second reconnect.

6.10.3 Node ID syntax

Enter the NodeId in standard OPC UA notation:

Form	Example
String identifier	ns=2;s=Line1.Tank101.Level1
Numeric identifier	ns=3;i=1042

Form	Example
Controllers using the IEC 61131-3 `	`ns=4;s= var PLC1.Application.GVL.Tank_Level` notation
Array element	append the index to the base address in the tag, or write to Tag[2] / Tag[1,3] (see 6.11.4)

For `[var]` NodeIds, OptiZeus removes the redundant path segments `DeviceSet.`, `.Resources.` and `.GlobalVars.` before reading or writing, so both the long and the short form work.

6.10.4 Browsing

To pick a node, open the tag dialog, select the OPC UA connection and click **Browse...** next to **PLC Address** (the connection must be connected). The browser starts at the **Objects** folder. Click a folder to open it, use the path bar to go back, type in the search box to filter the current level, and click a variable to take its NodeId. Variables that are structures show a ► button that expands their members. When a readable path is known, the tag dialog shows the path in blue with the raw NodeId below; the × button clears the path so that you can edit the raw address.

Servers that use opaque NodeIds (`ns=3;b=...`) cannot be typed by hand. For these, the tag's readable **browse path** (`opc_browse_path`, for example `Symbol Set\Line1\Tank101\Level1`) is resolved to the live NodeId when you import tags by CSV (Chapter 7).

6.10.5 Writes

The write routine:

1. uses the data type already known from the subscription cache; otherwise reads the node once to learn its data type (and assumes Float if that read fails);
2. converts the value: Boolean (`1`, `"1"`, `true`, `"true"`, `"TRUE"` → `true`), integer types rounded, Float/Double parsed;
3. writes through the dedicated write session;
4. on success reads the node back once and puts the confirmed value into the cache, so the screen shows the new value immediately even if the server's next data change arrives late.

In addition, before the write is sent, the new value is broadcast to all clients at once ("optimistic" update) so the operator sees the change without waiting for the controller.

Note — changed in 2.6.1.24: If the controller rejects the value (for example out of range), the write status is not "Good" and the write counts as **failed**: the server broadcasts the old value again at once, so the optimistic value disappears from every screen, and the write is answered as described in 6.11.3 (*Write Failed!*).

6.10.6 OPC UA status panel

Config → **Admin** → **Performance** contains the **OPC UA Status** panel. It refreshes every 5 s (**Auto-refresh (5 s)**, **Refresh now**) and shows:

- totals: connected servers, total monitored items, publishing interval, last poll cycle, read chunk size;
- one card per server: server ID, endpoint, **Subscription · n items** or **No subscription, Multi-node Read ✓ / X** / **Probe pending, Reconnecting...**, Monitored Items, Pending First Push, Cached Values, Oldest Cache Age,

Publish Interval, Subscription Age, Last Activity;

- **Reset capability** — forgets whether the server supports multi-node reads, so the next read probes again (use it after a controller firmware update).

6.11 Reading and writing: common behaviour

6.11.1 Scan rate

All drivers except EtherNet/IP are served by one acquisition loop that runs every **1000 ms** by default; values are broadcast to clients every 1000 ms. EtherNet/IP runs continuously in its worker; MQTT values arrive when the broker delivers them; OPC UA values are served from the subscription cache.

The loop rate is set per installation, not per connection, in `backend\performance-config.json` :

```
{
  "scanIntervalMs": 500,
  "broadcastIntervalMs": 400
}
```

Values below 100 ms are ignored. The file is read when the service starts; restart the OptiZeus service after changing it (Chapter 26).

Important: Lower the scan interval only on OPC UA-based installations, where a faster scan costs almost nothing. S7, Modbus, PCCC, BACnet and SRTP are polled inside the loop, so halving the interval doubles the load on those controllers and on the server. If a scan takes longer than the interval, the next scan is skipped rather than queued.

6.11.2 Manual-write protection

After a successful write from OptiZeus the tag remembers the time of the write (`lastManualWrite`). For the next **5 seconds**:

- values arriving from the OPC UA subscription cache do not overwrite the written value (the cache may still hold the old value for a few publishing intervals);
- simulation does not overwrite the value.

This prevents the value on screen from jumping back to the old value and then forward again. S7, Modbus, EtherNet/IP and PCCC reads are not held back; for these protocols the value read in the next scan is the value in the controller.

6.11.3 What happens when a write cannot reach the controller

For a tag that is bound to a controller — the tag has a connection and a **PLC Address**, and its data source is not **Memory** or **Simulation** — the backend tries the connection's driver. If the connection is not connected, the driver reports an error, the OPC UA server answers with a bad status, or there is no write driver for the protocol, the write **fails**:

- the value in OptiZeus is **not** changed (an optimistic OPC UA value is rolled back to the old value, 6.10.5);
- the client receives HTTP 502 with the message *"Not written: the PLC connection "<name>" is not connected or rejected the write."*, and the screen shows **Write Failed!** with that message;

- the event log records `tag_write_failed` (category `control`, severity warning) with the tag, the value, the connection, the address, and the reason and confirming user if they were given.

Tags without a controller behind them still update locally: a **Memory** tag answers "*Memory tag updated*" (the value is also saved in the database), a simulated tag answers "*Simulated tag updated (no PLC behind this tag)*". These writes are logged as `tag_write` with the note `(local)`.

Note — changed in 2.6.1.24: Up to 2.6.1.23 a write to a PLC-bound tag that did not reach the controller was stored locally and answered "*Value updated locally (no PLC connection)*", so it looked successful on the screen. From 2.6.1.24 it is reported as a failure as described above. Still verify every write path end-to-end with the controller during commissioning (start/stop commands, setpoints): a write accepted by the driver tells you the controller took the value, not that the plant reacted to it.

6.11.4 Writing array elements

A write to `TagName[i]`, `TagName[i][j]` or `TagName[i][j][k]` writes one element of an array tag. For OPC UA the index is appended to the NodeId as `[i]`, `[i,j]` or `[i,j,k]` (comma without spaces), using the indices exactly as typed — so they must match the controller's own index base. Locally, the element in the array value is updated using the tag's **Array Base Index** (Chapter 7).

6.11.5 Write path summary

Step	Behaviour
Permission	Role <code>operator</code> or <code>admin</code> ; licence (or trial) must allow tag writes; security areas that enforce writes are checked (Chapter 25).
Clamping	If Clamp output to engineering range is on, the value is limited to the engineering range (or the wider of engineering and display range).
Reverse scaling	With linear scaling enabled, the engineering value is converted back to the raw range; rounded if the raw limits are integers.
Driver write	As described per protocol.
Result	Success only when the driver confirms the write (6.11.3). A PLC-bound tag whose write does not reach the controller is not changed and the client receives HTTP 502.
Audit	Event <code>tag_write</code> with old value, new value, user, optional reason and confirming user; <code>tag_write_failed</code> when the write did not reach the controller.

Since 2.6.1.26 the server's own writers — weekly schedules, automation actions, alarm-pipeline **Set Tag** steps and recipe downloads — run through this same path, so the protocol coverage, scaling, failure handling and audit above apply to them too. Security areas are not applied to them, and they are recorded with the user `system`, `scheduler` or (recipe download) the operator who started the download.

6.12 MQTT (client connection)

6.12.1 Addressing

The tag's **PLC Address** is the topic to subscribe to, optionally followed by `::` and a dotted JSON path:

Address	Payload received	Tag value
plant/line1/tank101/level	62.4	62.4
plant/line1/pump201/state	true	1
plant/line1/tank101::level	{"level":62.4,"temp":18.2}	62.4
plant/line1/tank101::data.sensors.temp	{"data":{"sensors":{"temp":18.2}}}	18.2
plant/line1/batch/id	B-2026-0412	B-2026-0412 (string)

Payloads are parsed as JSON when possible; otherwise as a number, otherwise kept as text. Booleans become 1/0. Wildcards (+ , #) are not supported in tag addresses.

6.12.2 Behaviour

- The connection subscribes to all topics of its tags when it connects. **A tag added later receives values only after the connection is re-established** — click **Disconnect** and **Connect** after adding MQTT tags.
- Values are event-driven: the scan reads the last received value. A tag has no value (and keeps its previous value) until the first message on its topic arrives. Use retained messages on the publisher side so values are available immediately after a reconnect.
- Keep-alive 60 s; the client reconnects every 5 s after a loss.
- Writing to an MQTT tag publishes the value to the topic (without the `::path` part) with the connection's QoS. Objects are sent as JSON, everything else as text.

Note: During **Test Connection** with **Use TLS**, the broker certificate is not verified; during normal operation it is verified against the Windows/Node.js trusted roots. A broker with a self-signed certificate therefore passes the test but can fail at run time.

6.13 BACnet/IP

6.13.1 Addressing

`<objectType>:<instance>[:<property>]` — the property defaults to `presentValue`.

Object type (long / short)	Number
analogInput / ai	0
analogOutput / ao	1
analogValue / av	2
binaryInput / bi	3
binaryOutput / bo	4
binaryValue / bv	5
multiStateInput / msi	13
multiStateOutput / mso	14

Object type (long / short)	Number
multiStateValue / msv	19
device	8

Properties by name: `presentValue`, `objectName`, `description`, `statusFlags`, `outOfService`, `units`, `reliability`, `eventState`, `minPresValue`, `maxPresValue`, `covIncrement`, `objectType`, `objectIdentifier`; or a numeric property identifier.

Examples: `ai:3` (temperature sensor), `bo:1:presentValue` (fan command), `msv:0` (operating mode), `av:12:85`.

6.13.2 Behaviour

Each tag is read with one `ReadProperty` request per scan (no `ReadPropertyMultiple`, no COV subscription). APDU timeout 6 s. Writes use `WriteProperty` with **priority 16**: binary objects as `ENUMERATED` (0/1), multi-state objects as `UNSIGNED`, all others as `REAL`. A value written at priority 16 is overridden by any higher-priority command in the BACnet device.

6.14 GE SRTP

Addresses: `%<area><index>[.<bit>]` (the % is optional).

Area	Meaning	Unit
R	Registers	Word
AI , AQ	Analog inputs / outputs	Word
I , Q	Discrete inputs / outputs	Bit
M	Internal coils	Bit
T , G , S , SA , SB , SC	Temporary, Genius global, system status	Byte

Word areas are read as signed 16-bit INT (little-endian). With tag type **Float**, two words are read as a 32-bit REAL. `%R100.5` (or tag type **Bit**) extracts bit 5 of the word. Tags are read one request at a time; a read error closes the connection, which is reopened by the next connection attempt (6.15.1). Writes support INT, REAL (2 words), bits in words (read-modify-write) and discrete bits. Writes started by Automation Actions (Chapter 9) write 16-bit INT values only.

6.15 Status, alarms and diagnostics

6.15.1 Connected state

A connection's state is derived from real reads, not only from the TCP socket:

Rule	Value
Disconnection alarm raised	after the first failed reading
Disconnection alarm cleared	after 3 consecutive successful readings (prevents flapping)

Rule	Value
Automatic reconnection attempts	at most once every 30 s per connection
OPC UA session considered stale	no activity for 60 s
PLC watchdog	every 15 s, connections without tags are checked by a TCP connect to <code>ip:port</code> (3 s timeout) so their communication alarm still works

The communication alarm has the ID `plc_disconnect_<connection>`; its behaviour (priority, acknowledgement, notification) is configured in **Config** → **Alarms** (Chapter 17). Disabled connections raise no alarm.

6.15.2 Where to look when values are bad

Symptom	Check
All tags of a connection bad	Status dot; Test Connection ; network/firewall; the communication alarm; System Logs (Chapter 29) for lines starting <code>[OPC UA]</code> , <code>[S7]</code> , <code>[Modbus]</code> , <code>[EIP Worker]</code> , <code>[PCCC]</code> , <code>[MQTT]</code> , <code>[BACnet]</code> , <code>[SRTP]</code> .
Some tags bad, others good	Address syntax; S7: DB length and optimized access; EtherNet/IP: bad-tag cache (6.8.3); OPC UA: NodeId exists (browse it).
Values update slowly (OPC UA)	OPC UA Status panel: <i>No subscription, Multi-node Read X</i> , high <i>Oldest Cache Age</i> .
Values update slowly (S7/Modbus)	Too many scattered addresses (6.6.3, 6.7.3); scan interval (6.11.1).
EtherNet/IP log shows "Auto-reduced connections"	The PLC could not serve the parallel connections; lower Parallel Connections .
Write shows <i>Write Failed! / Not written: the PLC connection ... is not connected or rejected the write.</i>	Connection status (▶ Connect), address and write permission in the controller; 6.11.3.

The event log records `plc_connection_created`, every write (`tag_write`) and every write that did not reach the controller (`tag_write_failed`). The external alarm bridge and OPC UA subscription details are also written to the system log.

6.15.3 Redundancy and failover

OptiZeus has no automatic failover between two PLC connections (for example a redundant controller pair with two IP addresses). For redundant controllers, connect to an address that follows the active CPU (a redundant OPC UA server or a virtual IP provided by the controller system). Server redundancy of OptiZeus itself and data-source failover for the datalog are described in Chapter 27 and Chapter 19.

6.16 MQTT Sparkplug B publishing

Config → **Admin** →  **Sparkplug B** turns OptiZeus into a Sparkplug B **edge node** that publishes its tag values to a broker. It is an outbound integration (Chapter 28 describes it in full). Summary of the fields:

Field	Meaning
Enable Sparkplug B Publishing	Master switch.
Broker URL *	For example <code>mqtt://broker:1883</code> .
Username, Password	Broker credentials.
Group ID *	Sparkplug group (site or factory).
Edge Node ID *	Name of this OptiZeus node.
Device ID (optional)	When set, publishes DBIRTH/DDATA/DDEATH under this device instead of NBIRTH/NDATA/NDEATH.
Host ID (optional)	Subscribes to <code>STATE/<hostId></code> and republishes the birth certificate when the host comes online.
Client ID (optional)	Auto-generated if blank.
Publish only changed tags (recommended)	Send NDATA for changed tags only.
Use JSON payload (for debugging — non-standard)	Plain JSON instead of Protocol Buffers.

Topics follow `spBv1.0/<group>/<message type>/<edge node>[/<device>]`. The node answers the **Node Control/Rebirth** command. Tag writes from a Sparkplug host (NCMD/DCMD metric writes) are **not** executed in 2.6.1.26.

6.17 Worked example: a Modbus TCP skid and an S7-1200

Plant: Line 1 has a water-treatment skid with a Modbus TCP controller (IP 192.168.10.20, unit 1) and a filling machine with an S7-1200 (IP 192.168.10.30).

Skid register map:

Register	Content	Raw range
40001	Tank_101 level	0–10000 = 0–100.00 %
40002, bit 0	Pump_P201 running feedback	—
40002, bit 1	Pump_P201 fault	—
40010	Pump_P201 speed setpoint	0–1500 rpm
00001 (coil)	Pump_P201 start command	—

S7-1200 data block DB20 (non-optimised): `DBD0` = Filler_Rate (REAL, bottles/min), `DBX4.0` = Filler_Running.

1. **Config** → **PLC Connections** → **+ Add PLC Connection**. Name `Skid_Line1`, brand **Generic / Other**, model **Modbus TCP Device**, protocol **Modbus TCP**, host `192.168.10.20`, port `502`, **Unit ID** `1`. **Test Connection** → ...verified register read. **Add Connection**.
2. **+ Add PLC Connection**. Name `Filler_Line1`, brand **Siemens**, model **S7-1200**, protocol **Siemens S7 (ISO-on-TCP)**, host `192.168.10.30`, port `102`, **Rack** `0`, **Slot** `1` (check the CPU's hardware configuration). **Test Connection**, then **Add Connection**.

3. Create the tags (Chapter 7):

Tag	PLC Connection	PLC Address	Type	Notes
Tank_101_Level	Skid_Line1	40001	Analog	Linear scaling raw 0–10000 → 0–100, unit %
Pump_P201_Run	Skid_Line1	40002	Bit	Word Size INT, Bit Number 0
Pump_P201_Fault	Skid_Line1	40002	Bit	Bit Number 1
Pump_P201_SpeedSP	Skid_Line1	40010	Analog	unit rpm, range 0–1500
Pump_P201_Start	Skid_Line1	00001	Bool	Coil, writable
Filler_Rate	Filler_Line1	DB20.DBD0	Float	unit count (bottles/min)
Filler_Running	Filler_Line1	DB20.DBX4.0	Bool	

4. Open the Tag Monitor on the Tags tab (Chapter 7) and confirm all values are good.

5. Write test: in the Tag Monitor click the value of Pump_P201_SpeedSP, type 750, press Enter, and confirm in the skid's own panel that the setpoint changed.

WARNING: Do step 5 only with the process owner's permission, with the pump in a safe state, and never on the start command of running equipment.

The skid's tags 40001 and 40002 are read in one Modbus request (one block), and 40010 joins that block because the gap is under 10 registers.

7. Tags, UDTs and instances

This chapter describes tags — the named data points that connect screens, alarms, trends, reports and recipes to the process — and User Defined Types (UDTs), the templates from which OptiZeus generates a complete set of tags for one piece of equipment in one step. It covers the tag dialog field by field, the data sources (PLC, simulation, memory, calculated), scaling, folders, the Tag Browser, the Tag Monitor, bulk CSV import and export, and the naming rules. It is written for the engineer who builds the tag database. Supervisors use the Tag Monitor (7.9) to check live values.

Connections are described in Chapter 6, alarm configuration in Chapter 17 and historian recording in Chapter 19.

7.1 Concepts

Term	Meaning
Tag	A named value held in the OptiZeus server: a process value, a status, a setpoint, a counter or a text. Each tag has a type, a data source, a value, a quality and an update time.
Data source	Where the value comes from: PLC / External Device , Simulation , Memory or Calculated .
Folder	A path such as <code>Line1/Tanks</code> that organises tags in the Tag Browser. The folder is part of the stored tag name (7.3).
UDT	<i>User Defined Type</i> — a template with a list of members (for example a <i>Pump</i> with <code>Run</code> , <code>Fault</code> , <code>Speed</code>).
UDT instance	One piece of equipment created from a UDT, for example <code>Pump_P201</code> . Creating it generates one tag per member: <code>Pump_P201.Run</code> , <code>Pump_P201.Fault</code> , <code>Pump_P201.Speed</code> .
Quality	<code>good</code> , <code>bad</code> (read failed, connection lost, formula error) or <code>unknown</code> (not yet read).

All tags live in memory in the backend and are saved in the project database. A change to a tag's configuration takes effect in the next scan. If staging is enabled (Chapter 5), tag creates, updates and deletes are **staged** and take effect only after **Deploy**; the list shows them with a **PENDING** badge.

7.2 Permissions and licence limits

Operation	Roles allowed by the server
View tags and live values	Any authenticated user
Create, edit, delete a single tag; enable or disable historian recording of a tag	<code>admin</code>
Move tags between folders	<code>admin</code> , <code>operator</code>
Bulk import (CSV), delete all tags, delete folders	<code>admin</code>
Create, edit, delete UDTs and UDT instances	<code>admin</code>
Write a value to a tag	<code>operator</code> , <code>admin</code> (and the licence must allow writes)

Note — changed in 2.6.1.24: Earlier versions accepted single-tag create, edit and delete from any logged-in user. From 2.6.1.24 `POST /api/tags`, `PUT /api/tags/:name`, `DELETE /api/tags/:name` and `PUT /api/tags/:name/historian` require the `admin` role; other users receive *Insufficient privileges*.

The total number of tags is limited by the licence tier: Starter 100, Basic 1,000, Professional 5,000, Professional Max 10,000, Enterprise 20,000 (Appendix C). UDT member tags count like any other tag. When the limit is reached, "*Tag limit reached (current/max). Please upgrade your license to add more tags.*" is shown.

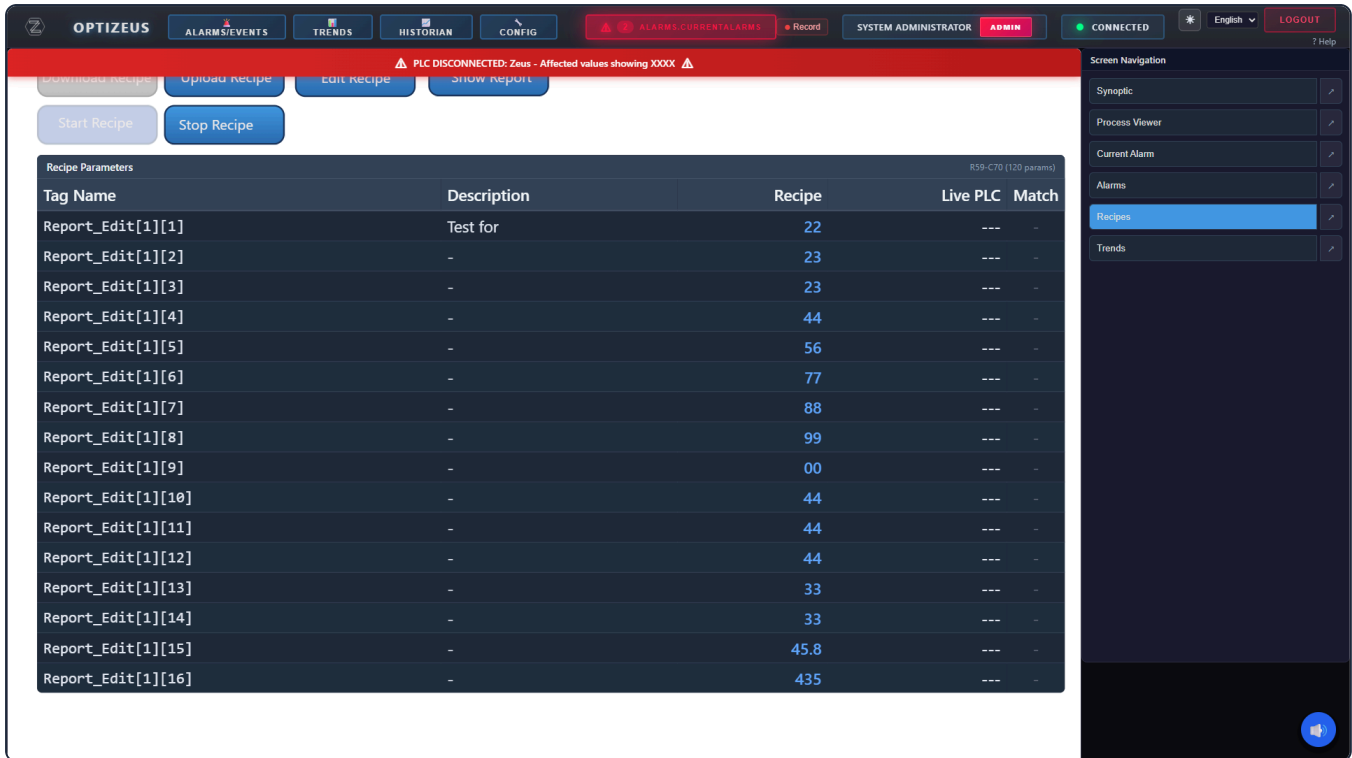
7.3 Naming rules

Rule	Detail
Characters	The Tag Name field accepts letters, digits and underscore only (<code>[a-zA-Z0-9_]+</code>).
Case	Names are case-sensitive: <code>Tank_101_Level1</code> and <code>tank_101_level1</code> are two tags.
Folder prefix	A tag created in folder <code>Line1/Tanks</code> with name <code>Level1</code> is stored as <code>Line1/Tanks/Level1</code> . Two folders may therefore each contain a tag <code>Level1</code> . Bindings, alarms, historian and reports use the full stored name.
UDT member tags	<code><instance>.<member></code> , for example <code>Pump_P201.Speed</code> , or <code>Line1/Pump_P201.Speed</code> if the instance is in a folder. The dot is reserved for UDT members.
UDT and instance names	May not contain a dot.
Renaming	A tag cannot be renamed in the dialog (<i>Tag name cannot be changed when editing</i>). Changing the Folder moves the tag in the Tag Browser without renaming it, so existing bindings keep working. To rename, duplicate under the new name, re-bind, then delete the old tag.
System tags	Tags in folder <code>_System</code> (for example <code>_System.ProjectVersion</code> , <code>_System.DeployCount</code>) are maintained by the server. Do not edit them.

Recommended convention: `<Area>_<Equipment>_<Signal>` for standalone tags (`L1_TK101_Level1`) and UDT instances named after the plant tag number (`Pump_P201`). Keep names stable — renaming later means re-binding screens, alarms and reports.

7.4 The Tags tab

To open it, click **Config** and select the **Tags** tab.



Tag configuration

Buttons at the top:

Button	Action
Export Tags to CSV	Downloads all tags as <code>tags_export_<date>.csv</code> (7.10).
Import Tags from CSV	Adds new tags and updates existing tags from a CSV file.
Replace All Tags from CSV	Deletes all tags of the project, then imports the file.
Add New Tag	Opens the tag form.
Create UDT Instance	Switches to the UDT / Super Tags tab with the instance form open (7.8).

Below the form area is the **Tag Browser** (7.9), and on the right edge of the page a tab opens the **Tag Monitor** (7.9.3).

7.5 The tag form

To create a tag:

1. Click **Add New Tag**.
2. Fill in the fields described below, section by section.
3. Click **Add Tag**. (When editing, the button is **Update Tag**; **Cancel** / **Cancel Edit** closes the form.)

To edit a tag, select it in the Tag Browser and choose edit; the form title becomes *Edit Tag: <name>*.

7.5.1 Identity and type

Field	Meaning	Default / range
Tag Name *	Unique name (7.3). Placeholder <i>temperature3</i> . Read-only when editing.	—
Folder	Folder path, / -separated, for example <i>TK20</i> or <i>Line1/Tanks</i> . Existing folders are suggested. The form shows the resulting <i>Stored name</i> .	Empty (root)
Tag Type *	Analog, Bool, Bit, Float, String, Time, Array [1D], Array [2D], Array [3D].	Analog
Label	Display label used on screens, faceplates and reports instead of the name.	Empty
Comment	Free description.	Empty

Tag types

Type	Use	Value
Analog	Numeric process value (level, flow, integer counters). Supports scaling.	Number
Float	Floating-point value. Supports scaling. For GE SRTP, reads a 32-bit REAL.	Number
Bool	Discrete signal.	0 / 1
Bit	One bit extracted from an integer register (Modbus, OPC UA, SRTP).	0 / 1
String	Text. No unit and no range.	Text
Time	Time-of-day value. No unit required. Initial value <i>00:00:00</i> .	Text / number from the source
Array [1D] / [2D] / [3D]	Array value read as a whole (typically from OPC UA); used by recipe and array tables.	Array

7.5.2 Bit extraction (type Bit)

Field	Meaning	Default / range
Word Size *	INT (16-bit) or DINT (32-bit) . For Modbus DINT, two consecutive registers are read (address and address + 1, first register = high word).	INT
Bit Number *	Bit to extract, 0–15 (INT) or 0–31 (DINT).	0

Writing to a Bit tag performs a read-modify-write of the whole register (Modbus, SRTP). Use a Bit tag for status words; use a separate Bool tag with a bit address where the protocol offers one (S7 DBX, PCCC B3:0/5, Logix BOOL).

7.5.3 Array configuration

Field	Meaning	Default
Array Element Type *	Bool, Int (16-bit), Dint (32-bit), Float, String, Time.	Float

Field	Meaning	Default
Array Base Index *	0-based (starts at 0) or 1-based (starts at 1) - PLC. Must match how the controller numbers the array.	0
Dimension 1 Size * , Dimension 2 Size * , Dimension 3 Size *	Number of elements per dimension (2 and 3 only for 2D/3D).	10

The form shows the total, for example *This will create a 2D array with $5 \times 10 = 50$ elements of type Float*. Elements are addressed on screens and in writes as `Recipe_Time[3]`, `Recipe_Time[1][2]` (Chapter 6, 6.11.4). The array is stored flat, row after row.

7.5.4 Historian

Field	Meaning	Default / range
Enable Recording	Record this tag in the historian.	Off
Recording Method	Interval — one sample per interval; On Change (Deadband) — a sample whenever the value moves by more than the deadband; Both .	Interval
Interval (ms)	Sampling interval.	1000 (min 100)
Deadband	Minimum change for <i>On Change</i> , in engineering units.	0

The historian must also be enabled globally (Chapter 19). The deadband affects recording only; it does not filter the value shown on screens.

7.5.5 Data source

Option	Meaning
PLC / External Device	<i>Tag value will be read from a PLC or external device via configured protocol.</i> Requires PLC Connection and PLC Address (7.5.6).
Simulation	<i>Tag value will be generated internally using simulation patterns (for testing/demo).</i> See 7.6.
Memory	<i>Tag value is stored in memory and persisted to database (survives restarts).</i> An internal variable: operator entries, flags, counters maintained by scripts. Selecting Memory clears any PLC binding.
Calculated	<i>Tag value computed from a formula using other tag values.</i> See 7.7.

A tag with no PLC binding and no explicit data source is treated as a simulation tag.

7.5.6 PLC binding

Field	Meaning
PLC Connection	The connection name (Chapter 6). -- <i>Select PLC</i> -- leaves the tag unbound.
PLC Address	Address in the syntax of the connection's protocol (Chapter 6, sections 6.6–6.14). The placeholder adapts: <code>ns=2;s=TagName</code> (OPC UA), <code>DB1.DBD0</code> (S7), <code>40001</code> (Modbus), <code>TagName</code> (others).

Field	Meaning
Browse...	OPC UA and EtherNet/IP connections only: opens the controller browser (6.10.4, 6.8.4). For OPC UA the selected node's readable path is shown in blue above the raw Nodetd; ✕ clears the path to edit the raw address.

7.5.7 Status attribute

Field	Meaning
Enable Status Attribute	Status attribute reads from PLC to indicate if the tag/device is active, in fault, etc.
Read Status from PLC	Reads the status from a second address.
Status address	Placeholder Status address (e.g., ns=2;s=Tag.Status). Browse... is available for OPC UA.

The status value is read in the same batch as the value and is available to animations and faceplates (Chapter 12). In 2.6.1.26 the status address is read only for OPC UA connections.

7.5.8 Retentive

Retentive (*retain last value on server restart*) is stored with the tag. In 2.6.1.26 the server does not save or restore the last value of PLC, simulation or calculated tags on the basis of this flag: PLC tags are re-read after a restart anyway, and **Memory** tags always keep their last written value regardless of the flag. Treat the flag as documentation.

7.5.9 Unit and range

Shown for Analog, Float, String, Time and array types (not for Bool and Bit).

Field	Meaning	Default
Unit	Engineering unit from a categorised list: Temperature, Pressure, Flow, Level, Volume, Mass/Weight, Speed/Velocity, Electrical, Time, Other. Required for Analog and Float; disabled for String.	—
Custom Unit (if not in list)	Free text unit, overrides the list.	—
Min Value, Max Value	Display range: used by bar graphs, gauges, trends and as the default simulation range. Not shown for String.	0 / 100

7.5.10 Linear scaling

Only for **Analog** and **Float**. Check **Linear Scaling (Raw → Engineering)** to convert the raw PLC value into engineering units.

Field	Meaning	Default
Raw Range (PLC) Min / Max	Raw value range from the controller.	0 / 32767
Engineering Range Min / Max	Corresponding engineering range.	0 / 100
Clamp output to engineering range	Limits the scaled value (and written values) to the engineering range.	On

The live preview shows the conversion. The formula is:

$$\text{eng} = (\text{raw} - \text{rawMin}) / (\text{rawMax} - \text{rawMin}) \times (\text{engMax} - \text{engMin}) + \text{engMin}$$

Scaling is applied immediately after every read, so clients never see unscaled values. When an operator writes an engineering value, OptiZeus converts it back ($\text{raw} = (\text{eng} - \text{engMin}) / (\text{engMax} - \text{engMin}) \times (\text{rawMax} - \text{rawMin}) + \text{rawMin}$), and rounds it when both raw limits are integers.

Example: a 4–20 mA level transmitter delivered by an S7 analog card as 0–27648: raw 0–27648, engineering 0–5 m. A raw value of 13824 is shown as 2.5 m.

7.5.11 What the tag form does not contain

Expected field	Where it is in OptiZeus
Alarm limits (HH/H/L/LL), bool alarms, alarm delays	Config → Alarms (Chapter 17). The tag's own threshold fields are kept for CSV import, UDT members and the <i>sync alarms from tags</i> function in the Alarms tab; they do not raise alarms by themselves.
Read-only / read-write flag	Every tag can be written by users with the <code>operator</code> or <code>admin</code> role. Restrict writes with screen object security and security areas (Chapter 12, Chapter 25).
Per-tag scan rate	The scan rate is global (Chapter 6, 6.11.1).
Acquisition deadband	None; the historian deadband (7.5.4) applies to recording only.
Data type for S7 DINT / Modbus FLOAT32	Not available (Chapter 6, 6.6.2 and 6.7.2).

7.6 Simulation tags

Field	Meaning	Default
Simulation Type	Numeric tags: Random (within range) , Sine Wave , Ramp (sawtooth) , Constant Value , Random Walk (noise) . Bool tags: Random Toggle , Constant Value , Toggle (alternating) . String tags: Constant Value , Random from List , Current Timestamp .	Random
Update Interval (ms)	Stored; see the note.	1000
Simulation Min, Simulation Max	Range of the generated value.	0 / 100
Constant Value	Bool: True (1) / False (0) ; String: text.	—
Random Values (comma-separated)	String <i>Random from List</i> .	—

Server behaviour in 2.6.1.26 (per scan, that is about once per second):

Type	Generated value
Sine Wave	$\text{mid} + (\text{range}/2) \times \sin(t / 5 \text{ s})$ — one period about 31 s
Ramp	rises from min to max every 10 s, then restarts
Constant	keeps the current value (set it by writing to the tag)

Type	Generated value
Random, Random Walk	random walk with a step of ± 0.5 % of the range, limited to min–max
Bool Toggle	inverts every scan
Bool Random	0 or 1 at random every scan

Note: The **Update Interval (ms)** is not used — simulation tags change once per scan. String simulation types are not implemented by the server; use Memory tags for text test values. A write to a simulation tag is kept for 5 seconds, then the simulation continues from the written value.

7.7 Calculated tags

A calculated tag is evaluated on the server in every scan after all PLC values have been read. Enter the expression in **Formula** (placeholder *e.g.*, `Tag1 + Tag2`, `(TempF - 32) / 1.8`, `Math.max(Flow1, Flow2)`).

7.7.1 Formula syntax

The formula is a JavaScript arithmetic expression:

Element	Syntax	Example
Tag reference (simple name)	the name itself	<code>Flow_FT101 + Flow_FT102</code>
Tag reference (name with /, ., - or spaces, i.e. folder tags and UDT members)	<code>{full name}</code>	<code>{Line1/Tanks/Level} * 2 ,</code> <code>{Pump_P201.Speed} / 1500 *</code> <code>100</code>
Operators	<code>+ - * / % ,</code> parentheses, comparisons <code>> < >=</code> <code><= == != ,</code> logic <code>`&&</code>	<code>! , conditional</code> <code>cond ? a : b`</code> <code>Level > 90 ? 1 : 0</code>
Functions	the JavaScript <code>Math</code> object	<code>Math.max(P1, P2, P3) ,</code> <code>Math.round(x) , Math.abs(x) ,</code> <code>Math.sqrt(x) , Math.min(...)</code>

Rules:

- Referenced tag values are converted to numbers; a non-numeric or missing value counts as **0**. `{Name}` references to a tag that does not exist also give 0.
- The result must be a number. It is rounded to 4 decimals. A result that is not a number (text, `NaN`) leaves the previous value unchanged.
- A syntax or runtime error sets the tag's quality to **bad**; the error is logged at most once per minute (`[Calculated] Tag "... " formula error: ...`).
- Calculated tags are evaluated one after the other in the order in which they are stored; a calculated tag that uses another calculated tag may see that tag's value from the previous scan.
- Boolean results (`true / false`) are not numbers — write `(Level > 90) ? 1 : 0`.

7.7.2 Examples

```
// Average of three temperatures
(TT_101 + TT_102 + TT_103) / 3

// Fahrenheit to Celsius
(TT_201_F - 32) / 1.8

// Total flow of two lines, UDT members
{Line1/FT_101.Flow} + {Line2/FT_201.Flow}

// Tank volume in m³ from level in m (cylinder, radius 1.2 m)
Math.PI * 1.2 * 1.2 * Tank_101_Level

// Pump P201 "available" = running feedback without fault (1/0)
(Pump_P201_Run == 1 && Pump_P201_Fault == 0) ? 1 : 0
```

WARNING: The formula is executed as JavaScript inside the OptiZeus service, without a sandbox. A formula can therefore do anything the service account can do. Only trusted engineers may create calculated tags, and the Configuration view must be closed to other users (see 7.2). Screen expressions (Chapter 9) use a different, safe evaluator.

Calculated tags cannot be written; a write only changes the value until the next scan.

7.8 UDTs and UDT instances

The **UDT / Super Tags** tab holds UDT definitions and their instances.

7.8.1 Creating a UDT

1. Open **Config** → **UDT / Super Tags**.
2. In **Create New UDT** enter **UDT Name** * (for example `Pump` ; no dots), a **Category** and a **Description**.
3. Under **UDT Members**, add each member with the fields below.
4. Save the UDT.

Member field	Meaning
Member Name *	Name of the member (<code>Run</code> , <code>Fault</code> , <code>Speed</code>). Becomes the part after the dot in the tag name.
Data Type	Float (Analog), Integer, Boolean (Digital), String, Time. Boolean members become digital tags, String members string tags, all others Analog tags.
Address Offset	How the member's PLC address is built from the instance's base address (7.8.3), for example <code>.Speed</code> or <code>[0]</code> . A full address (containing <code>ns=</code> or starting with <code>/</code>) is used as-is.
Unit, Min Value, Max Value	Defaults for the generated tag.
Description	Becomes the tag comment.
Enable Alarm for this Member	Stores alarm thresholds on the generated tags: High-High, High, Low, Low-Low (analog), Alarm when ON (true) / Alarm when OFF (false) with messages (boolean). Create the alarm configurations from these values in the Alarms tab (Chapter 17).

When you edit a UDT, **Sync changes to existing instances** applies changed member defaults (min, max, unit, label) and new members to all instances; *Instance-specific overrides will be preserved*.

UDT definitions can be **exported** to and **imported** from a JSON file (`{"exportDate": "1.0", "udts": [...]}`); on import, UDTs whose name already exists are skipped unless overwrite is requested. A UDT can be **duplicated**.

7.8.2 Creating a UDT instance

1. Click  **Create UDT Instance** on the Tags tab, or open the instance form on the **UDT / Super Tags** tab (*Create Tag Instance from UDT*).
2. Fill in the fields:

Field	Meaning
Select UDT *	The template.
Instance Name *	For example Pump_P201 . <i>Tags will be created as: InstanceName.MemberName</i> . No dots.
Folder	Folder of the instance; the instance's tags are stored as <code><folder>/<instance>.<member></code> .
Label, Unit	Instance label and unit (inherited by members in exports).
PLC Server *	The connection used by all members.
Tag Folder	Folder in which the member tags are shown (<i>Use / for subfolders</i>).
Base Address	The common part of the member addresses (7.8.3).
Member overrides	Per member: address offset, min, max, unit.

3. Save. OptiZeus creates one tag per member and places them under the instance in the Tag Browser.

7.8.3 How member addresses are built

Base Address	Member Address Offset	Resulting PLC address	
ns=2;s=Line1.P201	.Speed	ns=2;s=Line1.P201.Speed	
ns=2;s=Line1.P201	Speed	ns=2;s=Line1.P201.Speed (a dot is inserted)	
ns=2;s=Line1.P201	(empty)	ns=2;s=Line1.P201.<MemberName>	
Pump_P201 (Logix)	.Running	Pump_P201.Running	
(empty)	`ns=4;s=	var	PLC1.GVL.P201_Speed` used as-is
(empty)	40010	40010	

The concatenation suits symbolic protocols (OPC UA, EtherNet/IP, MQTT topics). For register-based protocols (Modbus, S7) leave **Base Address** empty and give every member its full address as the offset or override.

7.8.4 Managing instances

In the Tag Browser, each instance row has buttons to **edit** the instance (name, unit, addresses, folder), **duplicate** it under a new name with its member tags, **change its UDT type** (members are re-mapped; member tags that no longer exist in the new type are removed) and **delete** it with all its member tags. Duplicates and deletes are staged for deployment when staging is on. The server also offers regeneration of an instance's tags from the current UDT definition and a bulk address update; these are used after UDT changes.

7.9 Browsing and monitoring tags

7.9.1 Tag Browser (Tags tab)

The **Tag Browser** shows folders, standalone tags and UDT instances as a tree.

- **Expand All / Collapse All**; drag the edge to resize (double-click to reset).
- Search box *Search tags by name, label, or comment...*
- Folder rows: create a sub-folder, rename, delete. Deleting a folder asks *move* (move all tags and instances to root — the default and safe choice), *delete* (delete everything in the folder, confirmed a second time) or *cancel*.
- Tag rows: edit, duplicate (*Enter name for the duplicated tag*; the copy gets the label suffix (*Copy*)), delete, move to another folder.

7.9.2 Tag pickers in other editors

Wherever a tag must be chosen (screen objects, scripts, reports), OptiZeus offers a picker that lists standalone tags and UDT instances with expandable members, with a search box; some pickers allow multiple selection. In the Viewer Builder, the tag tree can be dragged onto the canvas: dropping a tag on an object binds it, dropping it on empty canvas creates a value display (Chapter 10).

7.9.3 Tag Monitor

The Tag Monitor is a live list that slides in from the right edge of the Tags tab (**Tag Monitor (n tags)**).

Element	Behaviour
<i>Filter tags...</i>	Case-insensitive text filter on the name.
Columns	Tag Name, Value, Unit. Bad-quality rows are highlighted. UDT members are grouped under their instance.
List size	At most 300 rows are shown: <i>Showing first 300 of n tags. Type in the filter box above to find a specific tag.</i>
Edit a value	Click the value, type the new value, press Enter to write or Esc to cancel. Numbers, <code>true</code> and <code>false</code> are recognised.

WARNING: A value entered in the Tag Monitor is written to the controller immediately, without a confirmation dialog (logged with the reason *Tag Monitor edit*). Use it for commissioning only and never on a tag that starts or stops equipment while the process is running.

7.10 CSV import and export

7.10.1 Export

 **Export Tags to CSV** writes one row per tag with these columns:

```
name, folder, label, comment, tagType, unit, min, max, server_name, plc_address,
opc_browse_path, udt_instance, udt_type,
threshold_high_high, threshold_high, threshold_low, threshold_low_low,
alarm_high_high_enabled, alarm_high_enabled, alarm_low_enabled, alarm_low_low_enabled,
alarm_on_enabled, alarm_off_enabled, alarm_on_severity, alarm_off_severity
```

- UDT member tags are exported with the bare `instance.member` name; the folder is in the **folder** column; **udt_instance** and **udt_type** identify the instance and its UDT. Empty unit, label and comment of member tags are filled from the instance.
- Values containing commas, quotes or line breaks are quoted.

7.10.2 Import

The importer reads the first line as the header and accepts any subset of the tag fields as columns (the export columns above, plus fields such as `dataSource`, `formula`, `scalingEnabled`, `rawMin`, `rawMax`, `engMin`, `engMax`, `historianEnabled`, `retentive`, `bitWordSize`, `bitNumber`). The parser is quote-aware. `true` / `false` become booleans, numeric texts become numbers, except for name-like columns (name, label, folder, `plc_address`, `server_name`, `unit`, ...) which always stay text.

Mode	Behaviour
 Import Tags from CSV	Inserts new tags and updates existing tags with the same (folder-qualified) name.
 Replace All Tags from CSV	After a confirmation (<i>This will permanently delete ALL n existing tag(s)...</i>), deletes every tag, then imports.

Additional import behaviour:

- **OPC UA path resolution:** if a row has an `opc_browse_path` (for example `[Controller] Symbol Set\Line1\TK101\Level`) and its connection is connected, OptiZeus browses the server and replaces `plc_address` with the live NodeId. This allows you to copy OPC tags in a spreadsheet by editing readable paths. Unresolved paths keep the CSV address and are counted in the result message.
- **UDT instances:** rows with a `udt_type` create the corresponding UDT instances automatically.
- The whole file is processed in one transaction and checked against the licence tag limit first; if the limit would be exceeded, nothing is imported.
- The result reports inserted/updated rows, skipped rows and OPC path results.

Important: Import writes directly to the project (admin only) and updates existing tags. Export first, keep the export as a backup, and review the file before a **Replace All**.

7.10.3 Example import file

```
name,folder,label,tagType,unit,min,max,server_name,plc_address,scalingEnabled,rawMin,rawMax,engMin,engMa
x
Tank_101_Level,Line1/Tanks,Tank 101 level,Analog,%,0,100,Skid_Line1,40001,true,0,10000,0,100
Pump_P201_Run,Line1/Pumps,P201 running,Bit,,0,1,Skid_Line1,40002,false,,,,
Pump_P201_SpeedSP,Line1/Pumps,P201 speed setpoint,Analog,rpm,0,1500,Skid_Line1,40010,false,,,,
Filler_Rate,Line1/Filler,Filling rate,Float,count,0,300,Filler_Line1,DB20.DB20,false,,,,
```

For the Bit tag, add columns `bitWordSize ( int )` and `bitNumber ( 0 )`, or set them in the form after import.

7.11 REST access

Tags can also be managed through the REST API (Chapter 28). The most used endpoints:

```
GET /api/tags # all tags with configuration and live value
GET /api/tags/search?q=Tank # search
POST /api/tags # create one tag (JSON body = form fields) (admin)
PUT /api/tags/Line1%2FTanks%2FTank_101_Level (admin)
POST /api/tags/bulk # { "tags": [ {...}, {...} ] } (admin)
POST /api/tags/Tank_101_SP/write # { "value": 75, "reason": "Shift change" } (operator, admin)
GET /api/udt # UDT definitions
POST /api/udt-instances # create an instance
```

URL-encode folder separators (/ → %2F) in tag names.

A write to a PLC-bound tag succeeds only when the controller accepted the value; otherwise the answer is HTTP 502 with *"Not written: the PLC connection "<name>" is not connected or rejected the write."* and the tag keeps its old value (Chapter 6, 6.11.3). Check the HTTP status of every write in an integration.

7.12 Worked example: pump UDT with 3 instances

Goal: three identical pumps (P201, P202, P203) in an OPC UA controller whose variables are `ns=2;s=Line1.P201.Run`, `.Fault`, `.Speed`, `.SpeedSP`.

1. **Config** → **UDT / Super Tags** → **Create New UDT**. Name `Pump`, category `Motors`. Members:

Member	Data Type	Address Offset	Unit	Min	Max
Run	Boolean (Digital)	.Run			
Fault	Boolean (Digital)	.Fault			
Speed	Float (Analog)	.Speed	rpm	0	1500
SpeedSP	Float (Analog)	.SpeedSP	rpm	0	1500

2. Save the UDT.
3.  **Create UDT Instance:** UDT `Pump`, Instance Name `Pump_P201`, Folder `Line1/Pumps`, PLC Server the OPC UA connection, Base Address `ns=2;s=Line1.P201`. Save. OptiZeus creates `Line1/Pumps/Pump_P201.Run`, `...Fault`, `...Speed`, `...SpeedSP` with addresses `ns=2;s=Line1.P201.Run` etc.

4. Repeat for Pump_P202 (base ns=2;s=Line1.P202) and Pump_P203 . (Faster: **duplicate** Pump_P201 in the Tag Browser and edit the base address.)
5. Open the Tag Monitor, filter Pump_P2 , and check that all twelve tags are good.
6. In **Config** → **Alarms**, create the Fault alarms for the three pumps (Chapter 17).
7. Build one pump faceplate as a super object bound to the UDT (Chapter 13) and place it three times.

8. OPC UA server

OptiZeus can act as an **OPC UA server** and publish selected tags to other systems — an MES, a plant historian, a reporting or analytics tool, or another supervisory system. This chapter describes the **OPC UA Server** page, its settings, the choice of exposed tags, the resulting address space, security and certificates, and the limits of the function in version 2.6.1.26. It is written for the system integrator who connects OptiZeus to higher-level systems.

This chapter is about OptiZeus as a *server*. Reading PLCs through OPC UA (OptiZeus as a *client*) is described in Chapter 6, Section 6.10. Other outbound integrations (REST API, webhooks, MQTT, Sparkplug B, ERP) are described in Chapter 28.

8.1 Overview

Property	Value in 2.6.1.26
Transport	OPC UA binary over TCP (<code>opc.tcp://</code>)
Default endpoint	<code>opc.tcp://<server host name>:4840/UA/OptiZeus</code>
Access	Read-only . Clients can browse, read and subscribe; writes are rejected.
Content	Only the tags you add to the Exposed Tags list.
Address space	<code>Objects/OptiZeus/Tags/<tag></code> (8.4)
Security modes	None; Sign or Sign and Encrypt with policy Basic256Sha256
User authentication	Anonymous and/or one user name with password
Start	Manual (Start Server), or automatically about 15 s after the OptiZeus service starts when Auto-start server when OptiZeus starts is ticked (8.3.1)
Licence	Sold as the OPC UA Server add-on (Appendix C)

8.2 Permissions

Operation	Roles allowed by the server
Open the page, read settings and status, list exposed tags	Any authenticated user
Save settings, start, stop, add or remove exposed tags	<code>admin</code>

The settings are returned to any logged-in user, but the OPC UA password is never sent: it is returned masked as `*****`. When the settings are saved with the mask unchanged, the stored password is kept; type a new password to replace it. (Before 2.6.1.24 the password was returned in clear text.) Still use a dedicated password for the OPC UA server that is not used anywhere else.

8.3 The OPC UA Server page

Open **Config** → **OPC UA Server**. The page header shows the server state, **Running** or **Stopped** (refreshed every 5 seconds), and two sub-tabs: **Server Settings** and **Exposed Tags (n)**.

8.3.1 Server Settings

Server Configuration

Field	Meaning	Default / range
Port	TCP port on which the server listens. <i>Default OPC UA port is 4840.</i>	4840 (1–65535)
Endpoint Path	Resource path appended to the URL.	/UA/OptiZeus
Security Mode	None (No Security), Sign, Sign and Encrypt.	None

Authentication

Field	Meaning	Default
Allow Anonymous Access	Clients may connect without a user identity. When cleared, anonymous clients are refused.	Checked
Username	Shown when anonymous access is cleared. The single user name clients must use.	—
Password	Password for that user. Shown as ***** when a password is stored; leave the mask to keep it.	—

Startup Options

Field	Meaning	Default
Auto-start server when OptiZeus starts	When set, the OPC UA server starts automatically about 15 seconds after the OptiZeus service starts (after the tags have loaded), with the exposed tags of the list. The event <code>opcua_server_autostart</code> is logged. Works since 2.6.1.24.	Cleared

When the server is running, an **Endpoint URL** box shows the address to give to clients: *Use this URL in your OPC UA client to connect.*

Buttons:

Button	Action
Save Settings	Saves the settings. A running server keeps its old settings until it is stopped and started again.
Start Server	Saves the settings, then starts the server with the current list of exposed tags. On success the message shows the endpoint.
Stop Server	Stops the server; connected clients are disconnected.

8.3.2 Exposed Tags

The tab shows two lists.

Area	Content and controls
Exposed Tags (n)	Tags that are published. Each row shows the tag name and, if different, its browse name. × removes the tag. Remove All removes every tag (confirmation <i>Remove all exposed tags?</i>).
Available Tags (n)	All other tags, with label and tag type. <i>Search tags...</i> filters by name or label. + adds a tag; Add All adds every tag in the filtered list.

When you add a tag, its **label** becomes its OPC UA browse name (or the tag name if it has no label).

Note: Changes to the exposed-tag list take effect only when the server is started. While it is running the page shows: *Restart the server for tag changes to take effect.* Click **Stop Server**, then **Start Server**.

8.4 Address space

After **Start Server**, clients see:

```

Root
├── Objects
│   ├── Server (standard OPC UA server object)
│   └── OptiZeus (folder)
│       ├── Tags (folder)
│       │   ├── Tank 101 level (Variable, Double) NodeId ns=1;s=Line1/Tanks/Tank_101_Level
│       │   ├── P201 running (Variable, Boolean) NodeId ns=1;s=Pump_P201_Run
│       │   └── ...

```

Item	Rule
Folder	All exposed tags are in one flat folder, <code>OptiZeus/Tags</code> . OptiZeus folders and UDT instances are not reproduced as OPC UA folders or objects.
Browse name	The tag's label at the time it was added (editable through the API as <code>browse_name</code>), otherwise the tag name.
NodeId	String NodeId <code>s=<full tag name></code> in the server's own namespace (namespace index 1 on a standard server). A custom <code>node_id</code> can be stored per exposed tag through the REST API.
Access level	CurrentRead only.
Value	Read live from the OptiZeus tag at every client read or subscription sample.

Data type mapping

OptiZeus tag type	OPC UA data type	Conversion
Analog	Double	numeric value, 0 if not numeric
Float	Float	numeric value
Bool, Digital (UDT boolean members)	Boolean	0 → false, anything else → true
String	String	text

OptiZeus tag type	OPC UA data type	Conversion
Bit, Time, arrays, all other types	Double	numeric value; a non-numeric value (text, array) is published as 0

Note: The published value carries no OPC UA status and no source timestamp from the PLC: a tag with bad quality in OptiZeus is still published with status Good and its last value. Clients that need the quality should also read a status tag, or use the REST API (Chapter 28), which returns the quality.

A tag that does not exist when the server starts (for example deleted since it was exposed) is skipped; the other tags are published normally.

8.5 Security and certificates

Setting	Endpoints offered to clients
Security Mode None	None / None
Security Mode Sign	Sign / Basic256Sha256, plus None / None while Allow Anonymous Access is checked
Security Mode Sign and Encrypt	SignAndEncrypt / Basic256Sha256, plus None / None while Allow Anonymous Access is checked

For a secured server, set **Security Mode** to **Sign and Encrypt**, clear **Allow Anonymous Access**, and enter **Username** and **Password**.

With **Allow Anonymous Access** cleared, the server refuses anonymous identity tokens: a client must log in with the **Username** and **Password**. Enter both before you clear the option, otherwise no client can log in. Also protect port 4840 with the Windows firewall so that only the known client computers can reach it (Chapter 25).

Note — fixed in 2.6.1.24: In 2.6.1.23 anonymous clients were accepted even when **Allow Anonymous Access** was cleared. Stop and start the OPC UA server after the upgrade (or restart the service with auto-start on) so that the setting takes effect.

Certificates. The server creates a self-signed application certificate automatically the first time it starts, using the OPC UA stack's default certificate store under the profile of the account that runs the OptiZeus service. There is no certificate management page in OptiZeus:

- Clients that connect with Sign or Sign and Encrypt must **trust the OptiZeus server certificate** (most clients show it on the first connection and ask you to accept it).
- Client certificates presented to OptiZeus are accepted by the stack's default trust handling; there is no OptiZeus page to approve or reject individual client certificates.
- The application URI is `urn:OptiZeus:Server`; the product name is *OptiZeus OPC UA Server*.

8.6 Procedure: publish tags to an MES

Goal: an MES must read the Line 1 production values every 5 seconds.

1. In **Config** → **OPC UA Server** → **Server Settings** set **Port** 4840 , **Endpoint Path** /UA/OptiZeus , **Security Mode** **Sign and Encrypt**, clear **Allow Anonymous Access**, and enter a **Username** (mes_reader) and a strong **Password**. Click **Save Settings**.
2. Open **Exposed Tags**. Type `Line1` in *Search tags...* and click + for each tag the MES needs (for example `Tank_101_Level` , `Pump_P201_Run` , `Filler_Rate` , `Line1_GoodCount`). Give the tags meaningful **labels** first (Chapter 7) — they become the OPC UA browse names.
3. Return to **Server Settings** and click **Start Server**. Note the endpoint in the message, for example `opc.tcp://SCADA01:4840/UA/OptiZeus` .
4. On the OptiZeus server, open TCP port 4840 inbound in Windows Defender Firewall for the MES server's IP address only.
5. On the MES side, create an OPC UA connection to the endpoint with security **Basic256Sha256 – Sign & Encrypt**, user `mes_reader` , trust the OptiZeus certificate, browse to **Objects** → **OptiZeus** → **Tags** and subscribe to the variables with a 5000 ms sampling interval.
6. Verify: change a value in OptiZeus (or wait for a process change) and check it in the MES client.
7. Tick **Auto-start server when OptiZeus starts**, and click **Save Settings**, so that the server is running again after every OptiZeus service restart. After a restart, check that the page shows **Running**; if it does not, click **Start Server**.

8.7 Limits and known issues in 2.6.1.26

Item	Behaviour
Auto-start	Works since 2.6.1.24 (8.3.1). The server starts about 15 s after the service; a client that connects earlier is refused. If the start fails (for example a port conflict) the reason is written to the console log ([OPC UA Server] auto-start failed).
Writes	Not supported; all variables are read-only.
Structure	One flat folder; no UDT objects, no engineering-unit or range properties (EURange, EngineeringUnits), no alarms and conditions, no historical access.
Quality	Not transported (8.4).
Users	One user name/password pair. The password is stored in the project database in clear text (it is masked in the settings returned to clients).
Security	Only the Basic256Sha256 policy is offered for Sign modes.
Tag list changes	Require Stop Server / Start Server .
Licence	The server is sold as an add-on. Version 2.6.1.26 does not block the function when the add-on is absent; use it only if the site's licence includes it.
Port conflicts	If another OPC UA server on the same computer already uses the port, Start Server fails with the stack's error message. Choose another port (for example 4841).
Redundancy	Each OptiZeus server publishes its own values; there is no redundant OPC UA server pair (Chapter 27).

8.8 REST endpoints

```
GET    /api/opcua-server/settings      # password returned as *****
PUT    /api/opcua-server/settings      # admin
GET    /api/opcua-server/status        # { running, endpoint, exposedTagsCount, port }
POST   /api/opcua-server/start         # admin
POST   /api/opcua-server/stop          # admin
GET    /api/opcua-server/exposed-tags  # { tag_name, browse_name, node_id, enabled }
POST   /api/opcua-server/exposed-tags  # { tag_name, browse_name, node_id, enabled }
POST   /api/opcua-server/exposed-tags/bulk # { tags: [ ... ] }
PUT    /api/opcua-server/exposed-tags/:id # { node_id, browse_name, enabled }
DELETE /api/opcua-server/exposed-tags/:tagName
DELETE /api/opcua-server/exposed-tags  # remove all
```

Example — publish a tag with a custom browse name and NodeId:

```
{
  "tag_name": "Line1/Tanks/Tank_101_Level",
  "browse_name": "TK101_Level",
  "node_id": "s=Line1.TK101.Level",
  "enabled": true
}
```

Exposed-tag rows with `enabled = false` are ignored when the server starts.

9. Scripts, expressions and automation actions

OptiZeus offers four ways to add logic without changing the PLC program: **event scripts** (Python programs run by the server on tag changes, alarms, schedules or on demand), **expressions** (formulas starting with `=` that compute a screen property in the client), **automation actions** (tag writes or scripts run at service start, service stop, on an interval or when a screen opens or closes) and the **weekly scheduler** screen object. This chapter describes each one, its exact syntax and functions as implemented in version 2.6.1.26, and its limits. It is written for the engineer who configures a project.

Server-side **calculated tags** (formula tags evaluated every scan) are described in Chapter 7, Section 7.7. Scheduled reports are described in Chapter 21, batch recipes in Chapter 20.

WARNING: Logic in the SCADA is not a substitute for the controller's safety and interlock logic. Scripts and automation actions run in a Windows service, depend on the network and can be delayed or skipped. Never implement protective functions (trips, interlocks, permissives) in OptiZeus; implement them in the PLC or the safety system.

9.1 Choosing the right tool

Need	Use
Show a colour, text or visibility that depends on tag values	Expression (9.3) or an animation (Chapter 12)
A derived numeric value that must be logged, alarmed or trended	Calculated tag (Chapter 7)
Write fixed values when the service starts or stops, or every N minutes	Automation action (9.4)
Run logic on a tag change or alarm; send an SMS; open a popup on every client; run a SQL named query	Event script (9.2)
Switch a tag on and off by a weekly timetable	Weekly scheduler (9.5) — run by the server every 30 s (since 2.6.1.26); no screen needs to be open

9.2 Event scripts

9.2.1 Requirements and execution model

- Scripts are written in **Python 3**. Python must be installed on the OptiZeus server and callable as `python` (Windows) or `python3` (Linux) by the account that runs the OptiZeus service. OptiZeus does not install Python.
- For every run the server writes the script, preceded by a generated header (9.2.4), to a temporary `.py` file and starts a **new Python process**. The file is deleted afterwards.
- Timeout:** 30 seconds. A script still running after 30 s is killed and the run is recorded as an error.
- No overlap:** while a script is running, further triggers of the same script are ignored (not queued).
- The script's standard output (up to 8,000 characters) is stored as the last output; errors are written to the event log as `script_error`.

WARNING: Scripts are not sandboxed. They run with the full rights of the OptiZeus service account and can read and write files, open network connections and start programs. Only administrators can create and edit scripts; review every script as you would review PLC code, and keep the server's service account at the minimum rights it needs (Chapter 25).

9.2.2 The Scripts page

Open **Config** → **Admin** →  **Scripts**. The page (**Event Scripts**) has a list on the left and an editor on the right.

List (**Event Scripts (n), + New**): each entry shows an icon for the event type, the name, an **ON/OFF** switch (click to enable or disable), the event type and the number of runs. The coloured left border shows the last result: green = success, red = error, grey = never run or disabled.

Editor:

Field / button	Meaning	Default
Name *	Unique script name.	—
Event Type *	What starts the script (table 9.2.3). Changing it clears the trigger settings.	Tag Change
Enabled	Active / Disabled. Disabled scripts never run automatically.	Active
Description	Free text.	—
Trigger settings	Depend on the event type (9.2.3).	—
Python Code	The script body.	Example code
Save	Creates or updates the script and (re)starts its schedule.	—
▶ Test Run	Runs the saved script once as a <i>manual</i> event and shows the result:  <i>Success (n ms)</i> with the output, or  <i>Error</i> with the error text. Save first.	—
Delete	Deletes the script after confirmation.	—
Run History	<i>Last run: <time> — <status> • Total runs: n</i> and the last error.	—

9.2.3 Event types and trigger settings

Event type (UI)	Key	Runs	Trigger settings
 Tag Change	tag_change	When the value of any watched tag changes.	Watched Tags (pick one or many) — tags or UDT members.
 Alarm Raised	alarm_raise	When an alarm becomes active.	Tag filter (optional), Severity filter (Any, Critical, High, Medium, Low), Type filter (for example HH, communication).
 Alarm Cleared	alarm_clear	When an alarm returns to normal.	Tag filter (optional).

Event type (UI)	Key	Runs	Trigger settings
 Gateway Startup	gateway_startup	Once when the OptiZeus service starts.	—
 Gateway Shutdown	gateway_shutdown	When the service stops normally.	—
 Scheduled	scheduled	On a schedule.	Schedule Type: Interval (every N seconds) with Interval (seconds) (min 5, default 60); Daily at a specific time with Time of day (local) ; One-shot at a specific date + time with Date + Time (local) .
 Manual	manual	Only from ► Test Run , the API, or an automation action.	—

Notes on triggers:

- **Tag Change** is evaluated each time the server broadcasts values to clients (about once per second). A tag's first value after start-up does not count as a change. If several watched tags change in the same cycle, the script runs once, for the first of them. Changes that come and go within one cycle are not seen.
- **Alarm Raised** severity filter: OptiZeus alarm severities are `critical`, `warning` and `info` (Chapter 17). Only **Critical** matches; the **High**, **Medium** and **Low** options never match in 2.6.1.26. Use **Any** and test `alarm['severity']` in the script instead.
- **Daily** runs at the given local time every day; **One-shot** is skipped if the time is already past when the script is saved or the service starts, and runs again only if you edit and save the script.
- Schedules are held in memory; after a service restart they are re-created from the saved scripts.

9.2.4 Variables and helper functions

The header generated before your code defines:

Name	Type	Content
<code>event</code>	str	'tag_change', 'alarm_raise', 'alarm_clear', 'gateway_startup', 'gateway_shutdown', 'scheduled', 'manual' or 'automation'
<code>tagName</code>	str	Tag that changed (Tag Change), otherwise ''
<code>oldValue</code> , <code>newValue</code>	any	Previous and new value (Tag Change), otherwise <code>None</code>
<code>alarm</code>	dict	The alarm (Alarm events): keys such as <code>tag</code> , <code>type</code> , <code>severity</code> , <code>message</code> , <code>value</code> ; otherwise <code>{}</code>
<code>tags</code>	dict	Snapshot of all tags at the moment of the run: <code>tags['Name']</code> is a dict with <code>value</code> , <code>quality</code> , <code>unit</code> , <code>label</code> , ...

Function	Description
<code>get_tag(name)</code>	Value of a tag from the snapshot, or <code>None</code> .

Function	Description
<code>log(*args)</code>	Prints <code>[SCRIPT] ...</code> to the script output (shown by Test Run, stored as last output).
<code>run_named_query(name_or_id, params=None)</code>	Runs a saved Named Query (Chapter 28) and returns <code>{'rows': [...], 'rowCount': n}</code> . Raises <code>RuntimeError</code> on failure.
<code>show_popup(screen_name)</code>	Opens the popup screen with that name on every connected client (a client that already shows it brings it to the front).
<code>send_sms(number, message)</code>	Sends an SMS through the configured GSM modem (Chapter 17 notifications). <code>number</code> in international format, for example <code>+15551234567</code> . Raises <code>RuntimeError</code> on failure.

The modules `json`, `urllib.request`, `urllib.error`, `os` and `sys` are already imported; any other installed Python module can be imported.

Important: In 2.6.1.26 scripts **cannot write tags**: there is no `set_tag()` helper, and the `tags` dictionary is a copy — changing it has no effect. To write a value on an event, use an automation action (9.4) for fixed values, or let the PLC react to a value that an operator or an automation action writes. The snapshot is taken when the run starts; for large projects (tens of thousands of tags) every run serialises all tags, so keep high-frequency tag-change scripts to a minimum.

9.2.5 Examples

Log a tank level alarm and send an SMS to the shift supervisor

Event type **Alarm Raised**, tag filter `Tank_101_Level`, severity **Any**:

```
sev = alarm.get('severity')
if sev in ('critical', 'warning'):
    lvl = get_tag('Tank_101_Level')
    msg = f"Line 1: {alarm.get('type')} alarm on Tank 101, level {lvl:.1f} %"
    log(msg)
    send_sms('+15551234567', msg)
```

Open a popup when pump P201 trips

Event type **Tag Change**, watched tag `Pump_P201_Fault`:

```
if newValue == 1 and oldValue == 0:
    show_popup('P201_Fault_Help')
```

Hourly production summary from the database

Event type **Scheduled, Interval (every N seconds)**, 3600:

```
res = run_named_query('Line1_HourlyCount', {'line': 'Line 1'})
for row in res.get('rows', []):
    log(row)
```


9.2.6 Legacy "run script" object actions

Screen objects can also start a Python file or an operating-system command from a click action (**Python Script Path**, Chapter 12). That older mechanism runs a file on the server with a 30-second timeout, is restricted to administrators and strips the characters `;` `&` `|` `$` and backticks from the command. Prefer event scripts of type **Manual** for new projects.

9.3 Expressions

An expression is a formula that starts with `=` and is typed into a screen property instead of a fixed value. It is evaluated in each client (browser or desktop app) whenever the screen re-renders with new tag values.

9.3.1 The expression input

Some property fields in the Viewer Builder have an expression input with three parts:

Control	Meaning
Aa / fx button	Toggles between a literal value (Aa) and an expression (fx). Switching to fx adds the leading <code>=</code> , switching back removes it.
Input field	In fx mode, the placeholder is <code>=Tag_Name > 50 ? 'red' : 'green'</code> ; a red border means a syntax error.
?	Shows the syntax help.

Below the field a line shows either the live result ($\rightarrow$ **value**, evaluated against current tag values) or the error (**message**).

Fields with the expression input: **Text Content** (text and label objects), **Label**, **Tooltip (hover text)**, **Popup Title**, **URL**, **Confirmation Message**. In addition, **any** text property of a screen object whose stored value starts with `=` is evaluated as an expression at run time (for example a colour entered as `=Level > 90 ? '#dc2626' : '#16a34a'`).

Note: Because of this rule, a literal text that must begin with an equals sign cannot be displayed as-is; prefix it with a space or use `"=..."`.

9.3.2 Syntax

Element	Syntax	Notes
Tag value	<code>Tag_Name</code>	Identifier: starts with a letter or <code>_</code> , may contain letters, digits, <code>_</code> and <code>.</code> (UDT members: <code>Pump_P201.Speed</code>).
Tag with other characters	<code>[Line1/Tanks/Level]</code> , <code>[Value Display]</code>	Square brackets for names with <code>/</code> , spaces or <code>-</code> . Array elements (<code>Recipe[3]</code>) cannot be referenced.
Numbers	<code>12</code> , <code>1.5</code> , <code>.5</code>	No exponent notation, no hexadecimal.
Text	<code>'red'</code> , <code>"Tank: "</code>	Backslash escapes the next character.
Constants	<code>true</code> , <code>false</code> , <code>null</code>	
Arithmetic	<code>+</code> <code>-</code> <code>*</code> <code>/</code> <code>%</code>	<code>+</code> concatenates if either side is text. Division by zero gives <code>NaN</code> .

Element	Syntax	Notes
Comparison	<code>> < >= <= == !=</code>	<code>== / !=</code> compare loosely (<code>1 == '1'</code> is true).
Logic	<code>`&&`</code>	<code>!` `a`</code> <code>b`</code> returns the first truthy operand, not only true/false.
Conditional	<code>cond ? a : b</code>	
Grouping	<code>(...)</code>	
Unary minus	<code>-x</code>	

Precedence, from lowest: `?:`, `||`, `&&`, `== !=`, `< <= > >=`, `+ -`, `* / %`, unary `! -`.

An unknown tag name evaluates to `null` (no error). A tag with bad quality still returns its last value; test quality with an animation instead (Chapter 12).

9.3.3 Functions

Group	Functions
Math	<code>min(a, b, ...)</code> , <code>max(a, b, ...)</code> (non-numbers ignored), <code>abs(x)</code> , <code>round(x, digits=0)</code> , <code>floor(x)</code> , <code>ceil(x)</code> , <code>sqrt(x)</code> , <code>pow(x, y)</code>
Transforms	<code>scale(v, inMin, inMax, outMin, outMax)</code> , <code>clamp(v, lo, hi)</code> , <code>lerp(a, b, t)</code> , <code>map(v, k1, v1, k2, v2, ..., default)</code> (returns the value paired with the first key equal to <code>v</code> , else the default, else <code>null</code>)
Text	<code>lower(s)</code> , <code>upper(s)</code> , <code>len(x)</code> , <code>trim(s)</code> , <code>startsWith(s, p)</code> , <code>endsWith(s, p)</code> , <code>contains(s, p)</code> , <code>replace(s, a, b)</code> (all occurrences), <code>left(s, n)</code> , <code>right(s, n)</code> , <code>substring(s, start, end)</code>
Logic	<code>if(cond, a, b)</code>
Type	<code>toNumber(x)</code> (0 if not numeric), <code>toString(x)</code> , <code>toBool(x)</code> (<code>'0'</code> and <code>'false'</code> are false), <code>isNaN(x)</code> , <code>isNull(x)</code>
Format	<code>format(n, digits=1)</code> (fixed decimals, <code>''</code> if not numeric), <code>concat(a, b, ...)</code>
Time (client clock)	<code>now()</code> (milliseconds since 1970), <code>today()</code> and <code>date()</code> (YYYY-MM-DD, UTC), <code>time()</code> (HH:MM:SS, local)

An unknown function name is an error. Function names are case-sensitive.

9.3.4 Examples

```
=Tank_101_Level > 90 ? 'red' : 'green'
="Tank 101: " + format(Tank_101_Level, 1) + " %"
=Pump_P201_Run && !Pump_P201_Fault
=round(min(TT_101, TT_102, TT_103), 1)
=[Line1/Tanks/Level] > 50 ? 'High' : 'Normal'
=scale(PT_301, 0, 27648, 0, 10)
=clamp(Pump_P201.Speed, 0, 1500)
=map(Line1_State, 0, 'Stopped', 1, 'Running', 2, 'Fault', 'Unknown')
=upper(trim(Batch_ID))
=if(Filler_Running, 'Filling ' + format(Filler_Rate, 0) + ' bpm', 'Idle')
```

9.3.5 Safety and limits

The expression evaluator is a dedicated parser: it has no access to JavaScript, the browser, the file system or the network, and cannot change tag values. Parsed expressions are cached (up to 500). Expressions are evaluated per client and per render, so a screen with hundreds of complex expressions costs CPU on every client; prefer a calculated tag (Chapter 7) when the same value is used on many screens. Expressions are not evaluated on the server — they cannot drive alarms, the historian or reports.

9.4 Automation actions

9.4.1 The Automation page

Open **Config** → **Automation**. The page (**Automation Actions**) lets you *Run tag writes or a saved script when the OptiZeus service starts (Windows boot), stops (Windows shutdown), or on a repeating interval*, and when screens open or close. Administrators only.

Editor (**New action** / **Edit action**):

Field	Meaning	Default
Name	Name of the action (required).	—
Trigger	Service: On startup (service start), On shutdown (service stop), Interval (every N min). Window (synoptic screen): Screen opened, Screen closed, While screen open (every N min).	On startup
Screen	Screen for window triggers (required).	—
Every (minutes)	Interval for the two interval triggers (≥ 1).	30
Action	Write tag value(s) or Run a script .	Write tag value(s)
Enabled		Checked
Tags to write	One row per write: tag name (with suggestions) and Value ; + Add tag adds a row, X removes one. Writes are executed in the listed order.	—
Script	Script to run (from 9.2); it runs with <code>event = 'automation'</code> .	—

Buttons: **Add action** / **Update action**, **Cancel**. The table lists **Name**, **Trigger**, **Action**, **Details**, **Last run**, **Enabled** (checkbox) and the buttons **Edit**, **Run now**, **Delete**. **Run now** executes the action immediately and shows the per-

tag result.

9.4.2 Trigger behaviour

Trigger	Behaviour
On startup	Once when the OptiZeus service starts serving.
On shutdown	Once during a normal service stop, before PLC connections are closed. Not executed on a crash or power loss.
Interval	Checked once per minute; runs when $now - last\ run \geq interval$. The first run happens within a minute after the service starts (there is no last run yet).
Screen opened / closed	Sent by the client that opens or closes the screen. If ten clients open the screen, the action runs ten times.
While screen open	Checked while at least one client shows the screen; runs when the interval has elapsed since the last run (shared by all clients).

Every run is logged as an `automation_action` event with the number of successful writes; creating, editing and deleting actions is recorded in the audit trail.

9.4.3 How automation writes are executed

Changed in 2.6.1.26: automation writes run through the same server code as an operator write (`POST /api/tags/<name>/write` , Chapter 16). Recipe downloads (Chapter 20), the **Set Tag** step of alarm pipelines (Chapter 17) and the weekly scheduler (9.5) use the same path.

Aspect	Behaviour
Supported protocols	Every driver that an operator write supports (Chapter 6), including EtherNet/IP (Logix5000 and MicroLogix/PCCC).
Memory, Simulation and Calculated tags	Handled as for an operator write (a Memory tag keeps the written value).
Scaling and limits	Reverse scaling and the tag's clamping to its limits are applied as for an operator write.
Security areas	Not applied: the action was configured by an administrator. The write is recorded with user <code>system</code> .
Result reporting	A write that fails is reported as failed in Run now (with the error), and the <code>automation_action</code> event ($n/m\ ok$) is logged as a warning; the failure is also logged as <code>tag_write_failed</code> . Successful writes appear as <code>tag_write</code> events.

Note — changed in 2.6.1.26: Up to 2.6.1.25 automation writes used a separate, simplified write routine: Memory, Simulation and Calculated tags were not written, EtherNet/IP and PCCC connections were not written, and a failed write was reported as `ok`.

Important: Test every automation action with **Run now** and check the result in the controller before relying on it — especially **On shutdown** actions intended to leave the plant in a defined state, which run while the service is stopping, when nobody watches the result.

9.4.4 Example: set a safe state on shutdown

1. **Config** → **Automation**. Name `Line1 safe state on stop`.
2. **Trigger On shutdown (service stop); Action Write tag value(s)**.
3. Rows: `Line1_SCADA_Healthy = 0`; `Pump_P201_SpeedSP = 0`.
4. **Add action**, then **Run now** and confirm both values in the PLC.
5. Create a second action `Line1 SCADA healthy`, **On startup**, writing `Line1_SCADA_Healthy = 1`.

The PLC program can then use `Line1_SCADA_Healthy` as information (not as a safety permissive) to show that the supervisory system is running.

9.5 Weekly scheduler

The **weekly scheduler** is a screen object (Chapter 11) that shows a 7-day × 24-hour grid and switches a Boolean tag ON (1) during the marked hours and OFF (0) otherwise.

9.5.1 Configuration

In the Viewer Builder the object's configuration dialog **Configure Weekly Scheduler** has:

Field	Meaning	Default
Title	Title shown on the object.	Weekly Schedule
Boolean Tag Binding	The tag to drive (Bool/Bit/Digital tags are listed; if there are none, the first 50 tags). <i>The tag will be set to 1 (ON) during scheduled times, 0 (OFF) otherwise.</i>	—
Display Options	Use 24-hour format, Show current time indicator, Compact mode (6-hour blocks).	On, On, Off
Colors	Active, Inactive, Current.	green, dark grey, blue
Preview & Configure Schedule	The grid: click a cell to toggle one hour, drag to set several.	All off
Save Configuration / Cancel		

The resolution is one hour; days run Sunday to Saturday. The server evaluates the schedule on the **server's** clock (the local time of the OptiZeus server); the grid on a screen shows the client's clock.

9.5.2 Operation

On a running screen the grid is editable: an operator clicks or drags cells, and the new schedule is saved immediately (✓ *Schedule saved*). The schedule is stored both in the screen and in the server's schedule table, and the event log records `schedule_updated`. The legend shows *Inactive (Tag = 0)* and the current hour is framed.

Since 2.6.1.26 the **server** switches the tag. Every 30 seconds it evaluates the enabled rows of the schedule table on the server clock and writes 1 or 0 to the bound tag when the scheduled state changes (at the start or end of a marked hour). It also writes the current state once, about 20 seconds after the service starts. The write uses the same path as an operator write (Section 9.4.3) with user `scheduler` and reason *Weekly schedule <title>*, so each

switch appears as one `tag_write` event. If the write fails (for example the PLC is offline), the failure is logged and the write is retried every 5 minutes until it succeeds. No client needs to have the screen open.

While a screen containing the scheduler is open, the screen object also writes the bound tag when the scheduled state changes on the client's clock (through `POST /api/tags/<name>/write`, reason *Weekly schedule*, with the operator's role and security areas; Chapter 16, Section 16.4.5). The value is the same as the server's, so this does no harm; keep the clocks of the server and the operator PCs synchronised so that the two do not disagree around the hour change.

Note — changed in 2.6.1.26: Up to 2.6.1.25 the server did not run the schedules; the tag was switched only while a client had the screen open. After upgrading, every enabled schedule with a bound tag drives its tag — also schedules on screens that nobody opens. Review the schedules (`GET /api/schedules`) before installing the update, and disable those that should not switch anything. **Note — fixed in 2.6.1.25:** Up to 2.6.1.24 the object sent its tag write to a server address that does not exist (`/api/tags/write`), so the bound tag was never switched.

The current state of all schedules can be read from `GET /api/schedules/status/all` (field `isActive`). For a time switch that must keep working while the OptiZeus server is stopped, implement it in the PLC (for example a PLC time-switch block whose on/off times are written from OptiZeus setpoint tags).

9.6 REST endpoints

```
GET    /api/scripts
POST   /api/scripts                # admin: { name, description, event_type, trigger_config,
python_code, enabled }
PUT    /api/scripts/:id           # admin
DELETE /api/scripts/:id           # admin
POST   /api/scripts/:id/run       # admin: { context: {} }

GET    /api/automation-actions
POST   /api/automation-actions    # admin
PUT    /api/automation-actions/:id # admin
DELETE /api/automation-actions/:id # admin
POST   /api/automation-actions/:id/run # admin

GET    /api/schedules
GET    /api/schedules/status/all
```

Example — create a scheduled script:

```
{
  "name": "Line1_HourlySummary",
  "description": "Logs the hourly production count",
  "event_type": "scheduled",
  "trigger_config": { "scheduleType": "daily", "time": "06:00" },
  "python_code": "res = run_named_query('Line1_HourlyCount')\nlog(res.get('rowCount'))",
  "enabled": true
}
```

Example — create an automation action (the tag must be bound to a PLC connection, see 9.4.3):

```
{
  "name": "Reset shift counter",
  "trigger_type": "interval",
  "interval_minutes": 480,
  "action_type": "write_tag",
  "action_config": { "writes": [ { "tag": "Line1_ShiftCount_Reset", "value": 1 } ] },
  "enabled": true
}
```

Valid `trigger_type` values: `startup`, `shutdown`, `interval`, `window_open`, `window_close`, `window_interval` (window triggers require `target_screen`). Valid `action_type` values: `write_tag`, `script` (with `action_config.scriptId`).

OPTIZEUS SCADA · OPERATION MANUAL

Part III — Screens

10. The synoptic builder

This chapter describes the **Viewer Builder**, the graphical editor in which you draw synoptic (process) screens and popup screens: how to open it, the layout of the editor, the canvas and its tools, selection, grouping, alignment, layers, copy/paste, undo, screen properties, saving and publishing, the screen list, and popup screens. It is written for the engineers and integrators who build screens. Operators who only use finished screens should read Chapter 16. The objects you place on the canvas are described in Chapter 11, their animations and click actions in Chapter 12, and reusable super objects in Chapter 13.

Note: In the user interface the synoptic builder is called the **Viewer Builder**. The two names mean the same editor. A *synoptic screen* is a screen drawn in this editor; a *popup screen* is a smaller synoptic screen that opens as a floating window on top of another screen.

10.1 Opening the builder

The builder is part of the Configuration view. The **Config** button in the header is shown to administrators by default (see Chapter 14 for how header buttons are configured).

To open the list of synoptic screens:

1. Click **Config** in the header.
2. Click the **Viewer Builder** tab.
3. Make sure the **Synoptic Screens (Viewer Builder)** sub-tab is selected. The other two sub-tabs are **Super Objects** (Chapter 13) and **Deploy Changes** (Chapter 5).

To create a new screen, click **+ Create New Screen**. To create a popup screen, click **+ New Popup Screen** (see Section 10.9). To edit an existing screen, click **Edit** on its card.

Note: When you click **Edit**, OptiZeus first loads the complete screen from the server. If that load fails, the screen is not opened and you see "Could not load screen "<name>". It was not opened, to avoid saving over it with an empty screen." Check the server connection and try again. This protects you from saving an empty screen over a real one.

10.1.1 Permissions

Operation	Roles allowed by the server
View the screen list and open screens	Any authenticated user
Create, update, delete a screen; manage screen folders	admin , operator
Create, update, delete super objects and super object instances	admin
Delete an SVG library item	admin
Add, edit, import SVG library items	admin , operator

In practice the builder is reached through the Configuration view, which is intended for administrators.

10.2 The synoptic screen list

The **Synoptic Screens** page lists every screen of the active project.

Element	Function
Import Screen	Imports a screen from a <code>.json</code> file previously produced by Export (see Section 10.8.3). You are asked for a name; the name must be unique.
+ Create New Screen	Opens an empty builder for a normal screen.
+ New Popup Screen	Opens an empty builder for a popup screen (default canvas 600 × 400 px).
All / Screens / Popups	Filter buttons. Screens hides popup screens; Popups shows only popup screens.
Folder tree (left)	All Screens (the root) and your folders. The number beside each entry is the number of screens it holds.
Rename / Delete Folder	Shown under the tree when a folder is selected. Deleting a folder does not delete its screens: they are moved to the root (" <i>All screens inside will be moved to root.</i> ").
Screen card	Name, a POPUP badge for popup screens, a PENDING CREATE / UPDATE / DELETE badge when the change is staged, the description, Objects: (object count) and Last updated: .
Edit	Opens the screen in the builder.
Export	Downloads the screen as <code>screen-<name>-<date>.json</code> .
Duplicate	Asks for a new name (default <code><name>_Copy</code>) and creates a copy.
Delete	Deletes the screen after confirmation (" <i>Are you sure you want to delete this synoptic screen?</i> ").

To move a screen into a folder, drag its card onto a folder in the tree or onto a folder tile. Drag it onto **All Screens** to move it back to the root. Folders can be nested; the path is shown as a breadcrumb above the cards.

A new screen is created in the folder that is open when you click **+ Create New Screen**.

Important: Screen names must be unique and must not contain a dot (`.`). The server rejects such names with "*Screen name cannot contain dots (.) - use underscores (_) or hyphens (-) instead*". Use names such as `Line1_Overview` or `Tank-Farm`.

10.3 The builder window

The builder occupies the whole Configuration area. From top to bottom it contains:

```

+-----+
| Edit Screen | Screen Name * | Description | [POPUP W: H:] | [Tags] [Hide] [Export] [Import] [Clear]
| [Save] [X] |
+-----+
| [Canvas] [Navigation Button] | Grid: On/Off | Snap On/Off | 20px | Zoom: - 100% + [Hand] |
|                               | Import CAD | Paste SVG | SVG Library | Background: ... |
+-----+
| Object | Objects: n | Undo | Redo | [Record] | (selection tools) | Properties |
| Library | ----- | ----- | ----- | ----- | ----- |
| (categories | | | | | |
| and items) | | | | | |
|             | C A N V A S | | | |
|             | | | | | |
|             | | | | | |
+-----+

```

10.3.1 Top row

Element	Function
Title	New Screen, New Popup or Edit Screen.
Screen Name *	Required. The name shown in navigation and in the screen list.
Description	Optional. Shown on the screen card and on the navigation card (Chapter 14).
POPUP W: / H:	Only for popup screens: the popup canvas width and height in pixels (200-2000).
[Tags]	Opens or closes the Tag Monitor (Section 10.3.4).
[Hide] / [Show]	Hides or shows the Object Library and Properties panels to give the canvas more room.
[Export]	Downloads the screen being edited as a JSON file.
[Import]	Adds the objects of a screen JSON file to the current canvas (Section 10.8.3).
[Clear]	Removes every object from the canvas after the confirmation <i>"Are you sure you want to clear the entire canvas?"</i> .
[Save]	Saves the screen (Section 10.8).
X	Leaves the builder. If there are unsaved changes, the Unsaved Changes dialog appears with Continue Editing and options to save or discard.

WARNING: [Clear] is not recorded in the undo history. If you clear the canvas by mistake, leave the builder with X and discard the changes instead of saving; the stored screen is unchanged until you save.

10.3.2 Second row: Canvas and Navigation Button tabs

The **Canvas** tab holds the grid, zoom, pan, import and background controls described in Sections 10.4 and 10.7. The **Navigation Button** tab holds the look of the card and button that represent this screen in the navigation panel; it is described in Chapter 14.

10.3.3 Object Library, canvas and Properties panel

- The **Object Library** on the left holds the drawing tools (select, pan, zoom), the library categories and their objects (Chapter 11), plus the special categories **Super Objects**, **Instances (n)**, [Tags] and **SVG Library**.

- The canvas in the middle is where you build the screen. The toolbar above it shows the object count, undo/redo, screen recording and the selection tools.
- The **Properties** panel on the right shows the properties of the selected object in tabs: **Props**, **Anim**, **Action**, **Attr**, **Script**, **Popup** and, for super objects, **Data**. With nothing selected it shows "Select an object on the canvas to view and edit its properties".

10.3.4 Tag Monitor

▣ **Tags** opens the **Tag Monitor**, a floating list of the tags used on this screen with their live **Value** and **Unit**. UDT instance members are grouped by instance and can be collapsed. A filter box (**Filter tags...**) narrows the list. The footer shows "*n screen tags | m total*".

A value in the Tag Monitor can be clicked (**Click to edit**) and typed in; pressing Enter writes the new value to the tag.

WARNING: A value entered in the Tag Monitor is written to the live tag, and through it to the PLC, exactly as an operator write would be. Use it only when it is safe to change that value on the running plant.

10.4 The canvas

10.4.1 Canvas size

The canvas has no fixed size setting. For a normal screen it is at least 2000 × 1500 pixels and grows automatically so that it always extends 100 px beyond the right-most and bottom-most object. For a popup screen it is at least the popup size (**W** × **H**) and is drawn with a dashed purple border so that you can see the popup area.

At runtime the viewer sizes the screen to its content (at least 1200 × 800 px). When a screen is shown inside a layout panel it is stretched to fill the panel (Chapter 14). Design each screen for the resolution of the monitors it will be shown on; a screen drawn for 1920 × 1080 displays best in a panel with the same proportions.

10.4.2 Zoom and pan

Action	How
Zoom in / out	+ and – in the Canvas tab (or  + /  - in the Object Library). Each step is 25 %, from 25 % to 300 %. The current zoom is shown between the buttons.
Pan	Select the Pan Tool () and drag the canvas. Click the Select Tool () to return to normal editing. The scroll bars of the canvas also work.

Note: There is no mouse-wheel zoom and no keyboard shortcut for zoom in the builder. The mouse wheel scrolls the canvas.

10.4.3 Grid and snap

Control	Meaning	Default / values
Grid: On/Off	Shows a light grid over the canvas.	Off

Control	Meaning	Default / values
Snap On/Off	When on, the position and size of objects snap to the grid while you drag or resize them, and objects created by dropping a tag snap too.	Off
Grid size	Spacing of the grid lines and snap step.	20px; options 10px, 20px, 25px, 50px

The grid settings are editor settings only; they are not stored with the screen and return to their defaults (off, 20 px) the next time the builder opens.

10.4.4 Background

The **Background**: control in the Canvas tab sets the background of the screen.

Field	Meaning	Default
Type	Solid or Gradient .	Solid
Background Color (solid)	Background colour.	#ffffff
Gradient Start / Gradient End (gradient)	The two gradient colours.	#ffffff / #e2e8f0
Gradient Direction (gradient)	→ (left to right), ← (right to left), ↓ (top to bottom), ↑ (bottom to top), ◎ (radial, from the centre outward).	↓

There is no background-image setting for a screen. To use a picture as a background, place an **Image** object (Chapter 11) that covers the canvas and send it to the back (Section 10.6.4).

10.5 Adding objects

To add an object from the library:

1. In the Object Library, click a category (for example **Tanks**).
2. Either drag the object onto the canvas and release it where you want it, or click the **+** button on the object to add it to the canvas at a default position.
3. Resize the object by dragging one of its handles, or type **Width** and **Height** in the **Props** tab.

To add a tag display quickly:

1. Click  **Tags** in the Object Library. The tag browser opens with a **Search tags...** box.
2. Drag a tag onto an empty part of the canvas. A tag display object is created at that point, bound to the tag and labelled with the tag name. (A digital tag gets an 80 px wide display, an analog tag a 120 px wide one.)

To bind a tag to an object that is already on the canvas, drag the tag from the tag browser and drop it **on the object**. A menu **Bind tag to object**: asks what to create:

Choice	Result
Animation	Adds a new animation using the tag. A digital tag gets a Color Change (On/Off) animation (green when true, grey when false); any other tag gets a Value Display animation with 1 decimal.
Action	Sets the object's click action: Toggle Tag (Bool) for a digital tag, Write Tag Value for any other tag.

Choice	Result
Both	Animation + Action.
Value Only	Sets the object's linked tag (the live value shown by the object).
Everything	Value + Animation + Action.

If the object already has a different tag in the chosen role, you are asked to confirm the replacement. If you drop a tag onto a super object, a different menu, **Bind tag to super-object variable**, lets you choose which variable of the super object receives the tag (Chapter 13); **Advanced** ► falls back to the menu above.

Note: A **Color Change (On/Off)** animation created by dropping a tag may show only its "false" colour at runtime until you open the object's **Anim** tab once and save the screen. See Section 12.3.2 for the reason and the work-around.

You can also drag a **datalog** onto a chart or table object to bind the object to that datalog; if another datalog is already bound, you are asked to confirm the replacement.

10.6 Selecting and arranging objects

10.6.1 Selecting

Action	Result
Click an object	Selects it. If the object belongs to a group, the whole group is selected.
Ctrl +Click	Adds the object to, or removes it from, the multi-selection.
Drag on an empty part of the canvas	Draws a selection rectangle (marquee); objects inside it are selected.
Ctrl +A	Selects every object on the canvas.
Alt +Click	Where several objects overlap, cycles the selection through them from top to bottom (Alt + Shift +Click cycles in the other direction).

When two or more objects are selected, the **Width** and **Height** fields in the **Props** tab show "*applies to n selected*": a value typed there is applied to every selected object.

10.6.2 Moving and nudging

Drag a selected object to move it (all selected objects move together). The arrow keys move the selection by 1 pixel; **Shift**+arrow moves it by 10 pixels. Exact positions can be typed in **Position X** and **Position Y** in the **Props** tab.

10.6.3 Canvas toolbar

The toolbar above the canvas changes with the selection.

Button	Shown when	Function
Objects: n	Always	Number of objects on the canvas.

Button	Shown when	Function
 Undo /  Redo	Always	See Section 10.6.7.
 Record	Always	Records the builder canvas as a video (see Chapter 15). While recording:  pause,  resume,  Stop (stops and downloads the recording).
 Group (n)	2 or more objects selected	Groups the selected objects (Section 10.6.5).
Align:      	2 or more objects selected	Align Left, Align Center Horizontal, Align Right, Align Top, Align Center Vertical, Align Bottom.
Size:  W  H 	2 or more objects selected	Same Width, Same Height, Same Size.
Space:  	2 or more objects selected	Distribute Horizontally / Distribute Vertically — equal gaps between objects; needs 3 or more objects.
 Ungroup	Selected object is in a group	Ungroups (Section 10.6.5).
 Rotate	One object selected	Rotates the object by 90°.
 Front ,  Back ,  ↑,  ↓	One object selected	Layer order (Section 10.6.4).
 Duplicate	One object selected	Duplicates the selection (Ctrl+D).
 Find/Replace Tag	One object selected	Opens Find & Replace Tag (Section 10.6.8).
 Delete	One object selected	Deletes the object; if it belongs to a group, the whole group is deleted.

With nothing selected the toolbar shows the hint "*Ctrl+Click to select multiple objects*".

How alignment works. The *first* object of the selection is the reference. **Align Left** moves every other object so that its left edge equals the reference's left edge; **Same Width** gives every object the reference's width, and so on. If only one object is selected but it belongs to a group, the alignment and size tools act on the whole group. **Distribute** keeps the left-most and right-most (or top-most and bottom-most) objects in place and spaces the others so that the gaps between neighbours are equal; a message reports the resulting gap in pixels.

10.6.4 Layers (z-order)

Objects are drawn in list order: an object later in the list is drawn on top of earlier ones. You change the order with:

Button	Function
Bring to Front ( Front)	Moves the object to the top of the stack.
Send to Back ( Back)	Moves the object to the bottom of the stack.
Bring Forward (↑)	Moves the object up one position.
Send Backward (↓)	Moves the object down one position.

The same four buttons also appear under **Layer Order (Z-Index)** in the **Props** tab. There are no named layers; there is only this single stacking order.

Tip: Place a large background image or panel first and use **Send to Back**, so that it never covers the process objects. Use **Alt+Click** to reach an object that is hidden under another.

10.6.5 Groups

A group ties objects together so that they are selected, moved, copied, duplicated and deleted as one.

To group objects:

1. Select two or more objects (Ctrl+Click or marquee).
2. Click  **Group (n)**.

To ungroup, select any member and click  **Ungroup**.

Important: Ungrouping **removes the animations and click actions of all group members**. If any member has an animation or action, OptiZeus warns *"This group has n animations and m actions on its members. Ungrouping will REMOVE them. Continue?"*. Click **Cancel** if you want to keep them.

When the selected object belongs to a group, the **Anim** and **Action** tabs offer to copy the current animation ("Copy this animation to n other group members?") or the action settings ("Copy this object's action settings to all n group members? Their existing action will be overwritten.") to every other member of the group.

10.6.6 Copy, paste and duplicate

Shortcut	Function
Ctrl+C	Copies the selected objects. If any selected object belongs to a group, the whole group is copied. The clipboard is kept in the browser, so you can paste on another screen.
Ctrl+V	Pastes the copied objects 30 px right and 30 px down from the originals. Pasting again cascades by another 30 px. Pasted groups become new, independent groups.
Ctrl+D or  Duplicate	Duplicates the selection 20 px right and down. Whole groups are duplicated.
Delete or Backspace	Deletes the selected objects.

The shortcuts are ignored while the cursor is in a text field.

Note: The **Delete** key removes only the selected objects. The  **Delete** toolbar button removes the whole group of the selected object. Select all members (click one member) before pressing **Delete** if you want the key to remove the whole group.

To copy only the behaviour of an object (not the object itself), use the **Anim + Action** strip at the top of the **Props** tab:  **Copy (nA / mX)** copies all animations (A) and actions (X) of the selected object; select another object and click  **Paste** to append them; **X** clears this clipboard.

10.6.7 Undo and redo

Ctrl+Z undoes and **Ctrl+Y** (or **Ctrl+Shift+Z**) redoes. The history holds the last 50 changes to the objects of the canvas and is cleared when you close the builder. Selecting objects is not an undoable step. **Clear** is not recorded (see the warning in Section 10.3.1).

10.6.8 Find & Replace Tag

Find & Replace Tag renames tag references in bulk — for example, after duplicating the screen of Tank 101 to build Tank 102.

Field	Meaning
Find:	Text to find in tag references, for example <code>TT_TK_01</code> .
Replace with:	Replacement text, for example <code>TT_TK_02</code> .
Match case	Case-sensitive matching.
Also update Data Logs (selected tags + event/batch tag fields)	Also renames the references in the project's datalogs (on by default).

Click **Replace All**. The replacement acts on the selected objects; if nothing is selected you are asked whether to apply it to all objects on the screen. It covers the linked tag, action tags, animations, actions, super object bindings and UDT instance references. A message reports the number of references replaced and, if selected, the datalogs updated.

Important: The datalog part of Find & Replace is written to the server immediately, but the canvas changes are stored only when you **Save** the screen.

10.7 Screen properties

A screen has the following stored properties. Most are set in the top row and the Canvas and Navigation Button tabs of the builder.

Property	Where it is set	Meaning	Default
Name	Screen Name *	Unique screen name, no dots.	—
Description	Description	Free text.	empty
Folder	Screen list (drag and drop)	Folder path, e.g. <code>Utilities/Water</code> .	root, or the folder open when the screen was created
Screen type	+ Create New Screen / + New Popup Screen	<code>screen</code> or <code>popup</code> . Cannot be changed later in the builder.	<code>screen</code>
Popup width / height	POPUP W: / H:	Popup canvas size in pixels (200–2000).	600 × 400
Background	Background: in the Canvas tab	Solid colour or two-colour gradient (Section 10.4.4).	solid white

Property	Where it is set	Meaning	Default
Navigation card and button	Navigation Button tab	Text, colour or gradient, shape of the button and card colour used for this screen in the navigation panel (Chapter 14).	View , #4299e1 , Rounded, card #ffffff
Objects	Canvas	All objects with their properties, animations and actions.	—

There is no per-screen access level. Access to individual actions on a screen is controlled with security letters (Chapter 12 and Chapter 24).

10.8 Saving and publishing

10.8.1 Saving

To save a screen:

1. Enter a **Screen Name** *. (Without a name you see *"Please enter a screen name"*.)
2. Click  **Save**.

The builder closes and you return to the screen list. One of two messages appears:

- *"Screen saved successfully!"* — the change is live. Clients load the new version of the screen when they next open it.
- *"Screen update has been staged for deployment"* (or *"Screen creation has been staged ..."*) — the project has staging enabled. The card shows **PENDING UPDATE** (or **PENDING CREATE**) and operators continue to see the previous version until the change is deployed.

To publish staged screens, open the **Deploy Changes** sub-tab (its badge shows the number of pending changes), enter a description of the change and click **Deploy Changes**. Staging and deployment are described in Chapter 5.

Note: When a screen is saved, the placed super objects are stored in a compact form (the reference to the super object definition plus the per-instance bindings and overrides). The viewer rebuilds them from the current super object definition when the screen is loaded, so changes to a super object definition appear on every screen that uses it (Chapter 13).

10.8.2 Leaving without saving

Click X. If you have unsaved changes, the **Unsaved Changes** dialog (*"You have unsaved changes. Do you want to save before leaving?"*) lets you continue editing, save, or discard.

10.8.3 Exporting and importing screens

Where	Button	Behaviour
Screen list	Export	Downloads the stored screen as JSON (<code>_exportType: optizeus-screen</code>) with its objects, background and navigation button settings.
Screen list	Import Screen	Creates a new screen from a JSON file. You are asked for a unique name. The screen is created in the current folder (or the folder stored in the file).
Builder	↓ Export	Exports the screen currently being edited, including unsaved changes.

Where	Button	Behaviour
Builder	↑ Import	Adds the objects of the file to the current canvas (the objects get new ids). The background and navigation settings of the file are applied. The name and description are taken from the file only if they are empty. Nothing is stored until you save.

Important: Export, **Import Screen** and **Duplicate** do not carry the screen type or popup size. A duplicated or re-imported **popup** screen becomes a normal screen. To copy a popup screen, create a new popup with **+ New Popup Screen**, then use **↑ Import** inside the builder to bring in the objects of the exported file.

Note: A screen JSON file contains tag names, not tag definitions. Import the tags (Chapter 7) before importing screens into another project, or use **Find & Replace Tag** afterwards.

10.9 Popup screens

A popup screen is a synoptic screen of type `popup`. It is used for detail windows: a pump faceplate, a valve control window, a set-point entry panel.

10.9.1 Creating a popup screen

To create a popup screen for pump P201:

1. On the **Synoptic Screens** page click **+ New Popup Screen**. The builder title reads **New Popup** and the **POPUP** badge is shown with **W:** 600 and **H:** 400.
2. Enter **Screen Name *** `Popup_Pump_P201` and set **W:** 400, **H:** 300.
3. Draw the content inside the dashed border: for example a title text, a value display for `Pump_P201.Speed`, **Start** and **Stop** buttons with **Write Tag Value** actions (Chapter 12) and a **Close** button.
4. On the **Close** button, set **Click Action** to **Close Popup**.
5. Click  **Save**.

Popup screens are listed with the **POPUP** badge and can be shown alone with the **Popups** filter. They are not offered as normal screens in the screen navigation panel.

10.9.2 Opening a popup from a screen

On the main screen, select the object that should open the popup (for example the pump symbol) and, in the **Action** tab:

1. Set **Click Action** to **Open Popup**.
2. Choose the popup in **Popup Screen**.
3. Optionally set **Popup Size (Width, Height)** to open this popup at a different size from this button, and **Popup Title**.
4. Save the screen.

At runtime the popup opens as a floating window centred on the screen, with the popup's name in its title bar and a close button. Several popups can be open at the same time; each further popup is offset by 30 px. The window is made large enough to show all objects of the popup at their designed size (the larger of the declared popup size and the extent of its objects), limited to about 95 % of the browser window; a size set on the opening button takes precedence.

A **Close Popup** action on an object inside the popup closes that popup. The **Additional Actions** of a button can also open, close or navigate (Chapter 12), so one button can, for example, write a value and then close its popup.

Note: The **Popup Title** field applies when **Open Popup** opens a *normal* screen or a built-in popup; a popup screen's floating window always shows the popup screen's name.

10.9.3 Object popups (without a popup screen)

Every object also has a **Popup** tab with **Enable Popup on Click**. This opens a built-in popup — a tag faceplate, a custom faceplate, a trend, an information text or custom HTML — without drawing a separate popup screen. See Section 12.7.

Important: When **Enable Popup on Click** is ticked, clicking the object opens that popup and the object's **Click Action** is **not** executed. Use either the Popup tab or a click action on one object, not both.

10.10 Importing drawings: CAD and SVG

The Canvas tab has three import buttons.

Button	Function
Import CAD	Imports a <code>.dxf</code> or <code>.svg</code> file. For DXF, the Import DXF - Select Layers dialog lists the layers and entity types found; select what to import (Select All / Select None , All / None) and click Import Selected . Supported DXF entities: LINE, CIRCLE, ARC, LWPOLYLINE, POLYLINE, ELLIPSE, TEXT, MTEXT, SPLINE. The drawing is converted to SVG and placed as one image object.
Paste SVG	Opens Paste SVG Code : enter an optional Name and paste the SVG XML Code , then click Import SVG . The SVG is placed as an image object.
SVG Library	Opens the SVG library manager (Chapter 11).

If a DXF file contains no supported entities you see *"Could not parse DXF file. No supported entities found. Try exporting as SVG from your CAD software instead."* For drawings from CAD software, exporting to SVG usually gives the better result.

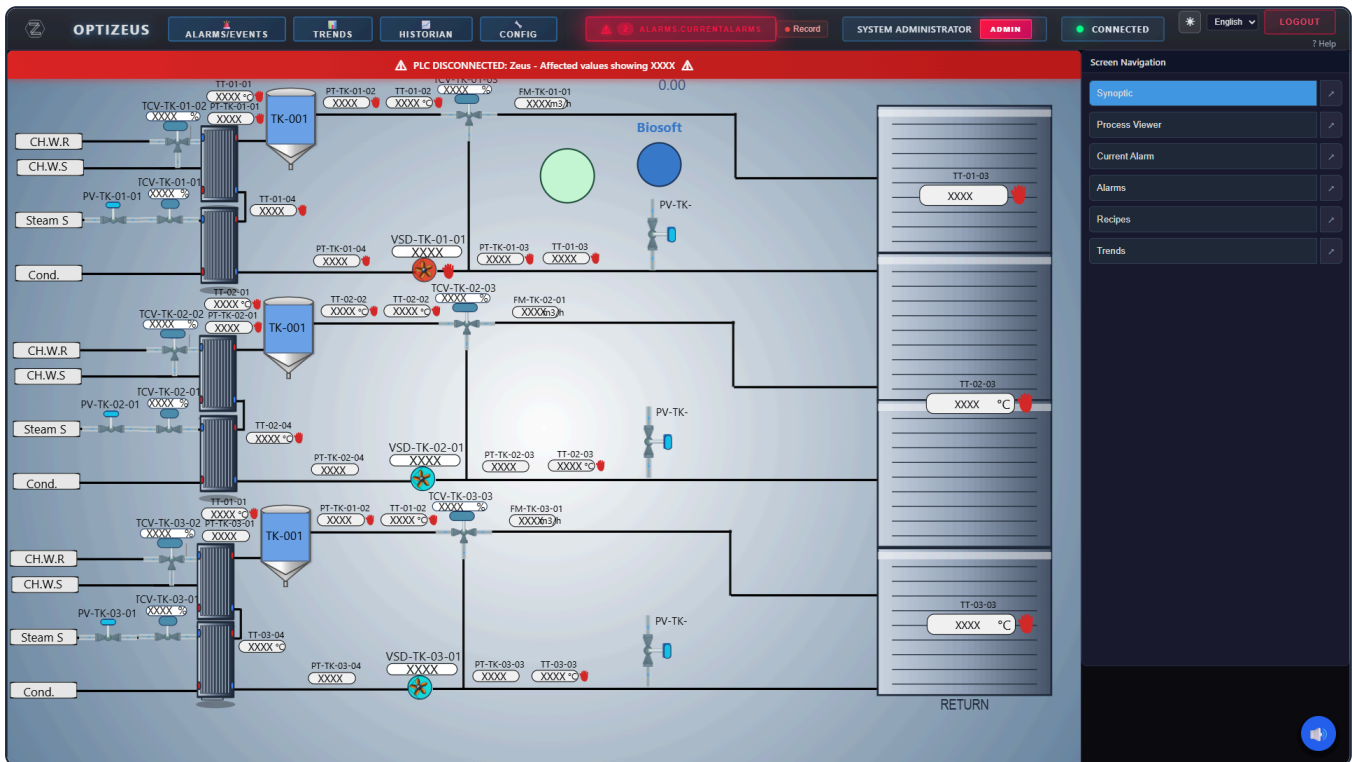
Tip: An imported drawing is a single image object. Use it as a static background (send it to the back) and place live objects — valves, pumps, value displays — on top of it.

10.11 Worked example: a tank overview screen

This example builds `Line1_TankFarm`, a screen with one tank, its level, an inlet valve and a transfer pump. It uses the tags `Tank_101.Level` (REAL, 0-100 %), `Valve_XV101.Open` (BOOL), `Pump_P201.Running` (BOOL) and `Pump_P201.Fault` (BOOL). The animations are configured in Chapter 12, Section 12.10.

1. Open **Config > Viewer Builder > Synoptic Screens (Viewer Builder)** and click **+ Create New Screen**.
2. Enter **Screen Name** * `Line1_TankFarm` and **Description** `Line 1 tank farm overview`.
3. In the Canvas tab set **Grid: On, Snap On**, grid size **10px**.
4. Set **Background:** to **Gradient**, **Gradient Start** `#f8fafc`, **Gradient End** `#e2e8f0`, direction ↓.

- From **Tanks**, drag a **Vertical Tank** to about X 300, Y 200; set **Width** 120 and **Height** 180 in the **Props** tab.
- From **Valves**, add a **Ball Valve** left of the tank; from **Piping**, add **Horizontal Pipe** segments to connect valve and tank.
- From **Pumps**, add a **Centrifugal Pump** right of the tank and connect it with pipes.
- From **Text**, add a **Title** (Tank Farm – Line 1) at the top and a **Value Display** under the tank.
- Select the three pipe segments on the inlet side and click **Align Center Vertical**, then **Group (3)**.
- Select the title and click **Front**.
- Add the animations and actions described in Section 12.10.
- Click **Save**. If staging is on, deploy the change in **Deploy Changes**.



Synoptic screen at runtime

10.12 Limits and good practice

Item	Value / behaviour
Undo history	50 steps per editing session
Zoom range	25 %-300 % in 25 % steps
Grid sizes	10, 20, 25, 50 px
Minimum canvas	2000 × 1500 px (screens), popup size (popups)
Popup size	200-2000 px each way
Screens per licence	The licence tier defines a maximum number of screens (Appendix C).

Good practice:

- Keep one process area per screen and use popups for detail. Very large screens with hundreds of animated objects load and render more slowly on weak client PCs.

- Use super objects (Chapter 13) for equipment that appears many times; a change to the definition then updates every screen.
- Name screens consistently (`Area_Function`) and use folders to organise them.
- After renaming tags, use **Find & Replace Tag** rather than editing objects one by one.
- Test a screen on a client with the target monitor resolution before deploying it.

11. Symbols, shapes and the SVG library

This chapter describes the objects you can place on a synoptic screen: the built-in Object Library with its categories, the basic shapes and text objects, images, the user SVG library, and the appearance properties of each kind of object. It is for engineers who build screens. The dynamic behaviour of objects (animations and click actions) is covered in Chapter 12; the data widgets of the **Components** category (charts, alarm tables, gauges, camera) are summarised here and described per widget in Chapter 14.

11.1 Overview of object sources

Source	Where in the builder	Content
Object Library	Category buttons in the left panel	52 built-in categories with 439 objects: P&ID equipment, piping, instruments, HMI controls, shapes, text, image, and data components.
Super Objects	Category button (shown when super objects exist)	Your reusable composite objects (Chapter 13).
Instances (n)	Category button (shown when instances exist)	Pre-bound super object instances (Chapter 13).
 Tags	Category button	Tag browser: drag a tag to create a tag display or to bind it (Chapter 10, Section 10.5).
SVG Library	Category button	SVG symbols stored on the server by your team (Section 11.5).
Import CAD / Paste SVG	Canvas tab	One-off imports of DXF/SVG drawings (Chapter 10, Section 10.10).

11.2 The Object Library

Click a category button to show its objects. Each object shows an icon and a name. Drag the object onto the canvas, or click its + button to add it. The hint at the bottom of the panel reads *"Drag objects onto the canvas to build your P&ID diagram"*.

11.2.1 Categories

The category button labels are derived from the internal category names (for example `heat_exchangers` is shown as **Heat Exchangers**).

Category	Objects	Contents
Tanks	14	Vertical Tank, Horizontal Tank, Sphere Tank, Cone Roof Tank, Floating Roof Tank, Pressure Vessel, ISO Tank, Conical Bottom Tank, Cone Roof (ISO), Floating Roof (ISO), Storage Sphere, Pressurized Vessel, Reactor (ISO), Mixer Vessel
Pumps	6	Centrifugal Pump, Positive Displacement, Reciprocating Pump, Gear Pump, Diaphragm Pump, Screw Pump
Compressors	4	Centrifugal Compressor, Rotary Compressor, Screw Compressor, Fan/Blower
Valves	9	Gate, Ball, Check, Globe, Butterfly, Needle, 3-Way, Motor Operated, Solenoid Valve

Category	Objects	Contents
Heat Exchangers	6	Shell & Tube HX, Plate Heat Exchanger, Double Pipe HX, Air Cooled HX, Heater, Condenser
Tags	7	Temperature 1/2, Pressure 1/2, Flow 1/2, Level 1 (see the note below)
Instruments	11	Temperature Sensor, Pressure Gauge, Flow Meter, Level Indicator, Magnetic Flow Meter, Coriolis Flow Meter, Rotameter, Turbine Flow Meter, Analyzer, Transmitter, Controller
Gradients	15	Gradient Rectangle, Square, Circle, Ellipse, Rounded Rectangle, Diamond, Triangle, Horizontal/Vertical Bar, Horizontal/Vertical Cylinder, Arrow Right/Left/Up/Down
Piping	25	Horizontal/Vertical Pipe, four elbows, four tees, Cross, reducers, flanges, four end caps, expansion joints, four flow arrows
Filters	5	Cartridge Filter, Bag Filter, Screen Filter, Strainer, Cyclone Separator
Motors	5	Electric Motor, Variable Speed Drive, Pneumatic Motor, Turbine, Engine
Mixers	4	Agitator, Static Mixer, Inline Mixer, Blender
Equipment	9	Cyclone, Scrubber, Dryer, Evaporator, Crystallizer, Centrifuge, Conveyor, Hopper, Silo
Electrical	5	Electrical Panel, Switch, Circuit Breaker, Transformer, Junction Box
Towers	6	Distillation Column, Absorption Tower, Stripping Tower, Packed Column, Tray Column, Cooling Tower
Labels	4	Text Label, Tag Number, Note, Title Block
Logic Gates	8	AND, OR, NOT, NAND, NOR, XOR, XNOR, Buffer
Flowchart	10	Start/End, Process, Decision, Input/Output, Document, Predefined Process, Database, Delay, Connector, Off-Page Connector
Math Operators	10	Add, Subtract, Multiply, Divide, Equals, Summation, Integral, Derivative, Square Root, Power
Comparisons	8	Equal To, Not Equal, Greater Than, Less Than, Greater or Equal, Less or Equal, Approximately, In Range
Control Flow	10	IF Condition, ELSE, Loop, WHILE Loop, FOR Loop, Break, Continue, Return, Function, Function Call
Data Structures	8	Array, List, Stack, Queue, Tree, Graph, Table, Object
Arrows	8	Arrow Right/Left/Up/Down, Double Arrow, Curved Arrow, Circular Arrow, Bold Arrow
Plc Ladder	8	Normally Open Contact, Normally Closed Contact, Coil, Timer, Counter, Move, Compare, Math Operation
Hmi Indicators	8	Green/Red/Yellow/Blue LED, Alarm Light, Warning Light, Pilot Light, Beacon
Hmi Buttons	14	Push Button, Emergency Stop, Start Button, Stop Button, Reset Button, Toggle Switch, Selector Switch, Key Switch, Flat Button, Rounded Button, Icon Button, Label Button, Close Button, Navigation Button
Hmi Displays	8	Digital Display, 7-Segment Display, Bar Graph, Trend Chart, Analog Meter, Speedometer, LCD Panel, Alarm Banner
Conveyors	6	Conveyor Belt Horizontal/Vertical, Roller Conveyor, Screw Conveyor, Bucket Elevator, Chain Conveyor
Storage	6	Storage Bin, Bunker, Chest Freezer, Storage Rack, Pallet, IBC Container

Category	Objects	Contents
Process Equipment	8	Mill, Grinder, Crusher, Extruder, Press, Feeder, Screener, Classifier
Actuators	6	Linear Actuator, Pneumatic Cylinder, Hydraulic Cylinder, Rotary Actuator, Damper, Louver
Robotics	5	Robotic Arm, Gripper, AGV Vehicle, Picker Robot, Palletizer
Utilities	7	Water Supply, Steam Supply, Compressed Air, Gas Supply, Power Supply, Drain, Vent
Safety	6	Safety Shower, Eye Wash, Fire Extinguisher, Safety Alarm, Emergency Exit, First Aid
Animation	8	Rotation Indicator, Pulse, Flow Left/Right/Up/Down, Vibration, Blink
Instrumentation	8	ISA-style Pressure/Flow/Level/Temperature Indicator-Transmitter, Analyzer Indicator/Controller, Flow/Pressure/Temperature Indicator/Controller
Opti Tanks	6	3D Tank, Cylindrical Tank, Hopper Tank, Cone Bottom Tank, Day Tank, Sump
Opti Valves	6	Angle Valve, Slide Valve, Pinch Valve, Rotary Valve, Air Operated Valve, Hydraulic Valve
Opti Gauges	6	Analog Gauge, Digital Gauge, Vertical Gauge, Horizontal Gauge, Thermometer, Manometer
Opti Switches	6	Limit, Proximity, Pressure, Temperature, Level, Flow Switch
Opti Motors Drives	6	Variable Frequency Drive, Soft Starter, 3-Phase Motor, Servo Motor, Stepper Motor, Gearbox
Opti Tags	8	Ultrasonic Sensor, Radar Sensor, Capacitance Sensor, Load Cell, Encoder, RTD Sensor, Thermocouple, Photoelectric Sensor
Opti Plc	7	PLC CPU, Input Module, Output Module, Analog Input, Analog Output, HMI Panel, SCADA Server
Opti Comm	6	Ethernet Switch, Router, Protocol Gateway, Wireless AP, Modem, Media Converter
Opti Power	6	UPS, Battery Bank, Power Supply, Inverter, Rectifier, Generator
Opti Hvac	8	Air Handling Unit, Fan Coil Unit, VAV Box, Chiller, Cooling Coil, Heating Coil, Damper Actuator, Humidifier
Opti Water	6	Water Pump, Sump Pump, Well, Water Treatment, Water Filter, Water Softener
Opti Material	6	Screw Feeder, Vibrating Feeder, Rotary Feeder, Belt Scale, Weigh Hopper, Baghouse Filter
Opti Pid	6	PID Controller, Cascade Control, Ratio Controller, Feedforward Control, Split Range, Override Control
Text	4	Text Label, Title, Note, Value Display
Shapes	29	Rectangle, Rounded Rectangle, Square, Circle, Ellipse, Triangle Up/Down/Left/Right, Diamond, Pentagon, Hexagon, Octagon, Star, 6-Point Star, Cross, Line, Vertical Line, Diagonal Line, Arrow Right/Left/Up/Down, Double Arrow H/V, Parallelogram, Trapezoid, Arc, Half Circle
Images	1	Image
Components	27	Data widgets: see Section 11.2.3

Note: The built-in **Tags** category (Temperature 1 ... Level 1) shares its internal name with the  **Tags** tag browser. Clicking either button shows the tag browser, so these seven preset objects cannot be reached from the library. Use a **Value Display** from **Text**, a **Transmitter** from **Instruments**, or drag a tag onto the canvas instead.

Note: The symbols are drawn in a clean 2D P&ID/HMI style. "3D Tank" is a shaded 2D drawing; there is no 3D rendering (see Chapter 15).

11.2.2 Default sizes

Each library object has a default size, for example Vertical Tank 80 × 120 px, Horizontal Tank 120 × 80 px, IP Camera 480 × 360 px, State Text 160 × 40 px. Resize after placing; SVG-based symbols scale without loss of quality.

11.2.3 The Components category

Components contains the data widgets. They fetch and display data by themselves and have their own settings in the **Props** tab.

Object	Internal type	Purpose
Real-Time Chart	chart	Trend of selected tags, real-time or from a datalog.
Historical Chart	historical_chart	Trend of datalog tags over a duration or date range.
Alarm Panel (Live)	alarms	Live alarm list.
Event Log (Live)	events	Live event list.
Historical Alarms	historical_alarms	Alarm history for a period, filterable by status.
Historical Events	historical_events	Event history, filterable by category and severity.
Weekly Scheduler	weekly_scheduler	Weekly time schedule that drives a boolean tag.
SPC Control Chart / SPC Histogram / Capability Analysis	spc_*	Statistical process control widgets (SPC add-on; Chapter 22).
OpAnalogGauge, OpLinearGauge, OpDigitalDisplay, OpStatusIndicator, OpBulletGraph, OpSparkline, OpTankLevel, OpMultiState	op_*	Operator gauges and indicators bound to one tag.
Server Topology	server_topology	Status of servers and deployment targets (Chapter 15).
Recipe Table	recipe_table	Recipe values next to live PLC values (Chapter 20).

Object	Internal type	Purpose
Array Table	<code>array_table</code>	Table view of a 1D/2D/3D array tag.
Process Viewer / Process Status Badge	<code>process_viewer</code> / <code>process_hold_indicator</code>	Stage-by-stage process view with hold points (Chapter 15).
IP Camera	<code>ip_camera</code>	Live camera image or camera web page (Chapter 15).
Current Alarms	<code>current_alarms_table</code>	Current alarm table with ACK button.
State Text	<code>state_text</code>	Text, colours and blink by integer state (Chapter 12).
Query Table	<code>named_query_table</code>	Result table of a named query.

The settings of these widgets are listed in Chapter 14, Section 14.6.

11.3 Common properties (Props tab)

Every object has the following properties in the **Props** tab.

Field	Meaning	Default / range
Label	Label text of the object. Accepts an expression (<code>=...</code> , Chapter 12).	empty
Tooltip (hover text)	Text shown when the mouse rests on the object at runtime. Accepts an expression.	empty
Position X / Position Y	Top-left corner in canvas pixels.	drop position
Width / Height	Size in pixels. With a multi-selection the value applies to every selected object.	library default
Layer Order (Z-Index)	Bring to Front, Bring Forward, Send Backward, Send to Back (Chapter 10, Section 10.6.4).	—
Rotation Angle (°)	Static rotation, 0-359 via number field or slider; quick buttons 0°, 45°, 90° ... 315°, and ↺ -15° / ↻ +15° .	0
Mirror / Flip	↔ Flip Horizontal and ↑ Flip Vertical toggles.	off

At the top of the tab the **Anim + Action** strip lets you copy and paste all animations and actions between objects (Chapter 10, Section 10.6.6).

11.4 Appearance properties by object kind

11.4.1 Equipment symbols (Object Appearance)

For P&ID and HMI symbols (tanks, pumps, valves, compressors, heat exchangers, filters, motors, mixers, equipment, electrical, towers, instruments, indicators, buttons, displays, conveyors, storage, process, actuators, robotics, utilities, safety, animation, all *Opti* categories, logic/flowchart/math/comparison/control/data/arrow/PLC symbols, labels and shapes) the **Props** tab contains **Object Appearance**:

Field	Meaning	Values
Fill Type	How the symbol body is filled.	Solid Color, Gradient, Transparent
Fill Color	Fill colour (solid).	colour
Gradient Start	First gradient colour.	colour
Use Middle Color / Gradient Middle	Optional third colour in the middle of the gradient.	off / colour
Gradient End	Last gradient colour.	colour
Gradient Direction	→ Left to Right, ← Right to Left, ↓ Top to Bottom, ↑ Bottom to Top, ↘ Top-Left to Bottom-Right, ↙ Top-Right to Bottom-Left, ↗ Bottom-Left to Top-Right, ↖ Bottom-Right to Top-Left, ◎ Center Outward (Radial)	—
Stroke Color	Outline colour.	colour
Stroke Width	Outline width in pixels.	number
Opacity	Transparency of the whole object.	0-1

Transparent leaves only the outline visible. It is useful for invisible click areas: draw a transparent rectangle over part of a drawing and give it a click action.

11.4.2 Shapes

The **Shapes** category (rectangle, circle, polygons, stars, lines, arrows, arc ...) uses the Object Appearance fields above plus:

Field	Meaning
Border Radius	Corner rounding in pixels (rectangles).

Lines are drawn as thin shapes: set the **Height** (horizontal line) or **Width** (vertical line) to the line thickness, or use **Stroke Width**.

11.4.3 Gradient objects

The **Gradients** category has its own **Gradient Colors** block: **Color 1**, **Color 2**, **Gradient Direction** (→, ←, ↓, ↑, ↖ To Top-Left, ↗ To Top-Right, ↙ To Bottom-Left, ↘ To Bottom-Right, ◎ Radial (Center Outward)), **Stroke Color**, **Stroke Width** and **Border Radius**. The cylinder and bar variants are convenient for tank bodies and bar graphs combined with a **Fill Level** animation.

11.4.4 Pipes

Pipe objects are drawn with three colours to give a cylindrical look:

Field	Meaning
Pipe Colors — Highlight, Main, Shadow	The light edge, body and dark edge of the pipe.
Flow Arrow Color	Colour of flow arrows on flow-arrow pipe pieces.
Stroke Color / Stroke Width	Outline.

Tip: To show flow, put a **Flow Right** (or Left/Up/Down) object from the **Animation** category on top of a pipe and give it a **Visibility (Show/Hide)** animation on the pump's running tag.

11.4.5 Text objects

The **Text** category objects (Text Label, Title, Note, Value Display) and **Labels** show a **Quick Format** block at the top of the Props tab and a full text block below:

Field	Meaning	Values / default
Text Content	The static text. Accepts an expression (= . . .).	—
Font Size	Size in pixels.	16
Font Weight	Light (300), Normal (400), Medium (500), Semi-Bold (600), Bold (700); the quick block also offers Black.	Normal
Font Family	Default, Arial, Segoe UI, Helvetica, Roboto, Inter, Poppins, Open Sans, Courier New, Consolas, Georgia, Times New Roman, Verdana, Tahoma, Impact	Default
Text Color / Bg Color	Text and background colours.	black / transparent
Style	Normal / Italic	Normal
Align / Text Align	Left, Center, Right	Left
Underline, Strikethrough	Text decoration.	off
Link Tag (Display Live Value)	A tag whose live value replaces the static text. -- None (Static Text) -- keeps the text static.	none
Value Format (e.g., 0.0, 0.00, 0.000)	Number of decimals for the linked value (the digits after the point in the pattern).	0.00
Enable Value Input	Makes the text an input field: clicking it opens a keyboard to write a new value to the linked tag. Ticking it sets the object's click action to Write Tag Value on the linked tag.	off

When the linked tag's PLC is disconnected or its quality is bad, the value is shown as **XXXX**. Internal (memory, simulation, calculated) tags are always shown.

WARNING: Enable Value Input turns a display into a control that writes to the PLC. Check the tag's engineering range (min/max, Chapter 7): values outside it are rejected with *"Value Out of Range!"*, but any value inside the range is written.

11.4.6 Images

The **Image** object (and SVG library symbols, which are placed as images) has:

Field	Meaning	Values
Fit Mode	How the picture fills the object.	Fill (stretch to fit), Contain (keep ratio, may leave gaps), Cover (keep ratio, may crop)
Image URL	URL of the picture, or an uploaded picture.	—
Export SVG Code	For SVG images: shows the SVG source with Copy to Clipboard .	—
Save to Library	Saves the image's SVG into the SVG library (Name * , Category , preview).	—

Note: Images are stored inside the screen. Large embedded pictures make the screen file large and slow to load. Prefer SVG over big bitmaps and reuse symbols from the SVG library.

11.5 The SVG library

The SVG library is a server-side collection of SVG symbols shared by all screens and all builder users of the project. Each item has a name, a category, the SVG code and a default size (200 × 200 if not given).

11.5.1 Using SVG library symbols

In the builder click the **SVG Library** category (the button shows the number of items). The symbols are grouped by category ("*category (n)*"). Drag a symbol onto the canvas or click **+**. The symbol is placed as an image object; its SVG code is kept in the screen.

The category panel also has buttons to paste SVG code into the library, import an SVG file into the library, and open the full manager.

11.5.2 The SVG Library manager

Open it with **SVG Library** in the Canvas tab. The window **SVG Library** contains:

Element	Function
+ Add SVG	Opens the editor to add a symbol by pasting code.
Import SVG File	Uploads one <code>.svg</code> file.
Export Library	Downloads the whole library as JSON.
Import Library	Imports a library JSON file (from Export Library); reports <i>"Imported n SVGs (m errors)"</i> .
Category list	All and each category.

Element	Function
Search SVGs...	Filters by name.
Item tile	Use (places the symbol on the canvas), Edit , Del (asks "Delete "<name>" from SVG library?").

The add/edit dialog:

Field	Meaning
Name	Symbol name, e.g. <code>Pressure Vessel</code> . Required ("Please enter a name").
Category	Existing category, or + New Category and a New Category Name .
Width / Height	Default size when placed.
SVG Code	The SVG source. Must contain <code><svg>...</svg></code> ("Invalid SVG code. Must contain <svg>...</svg> tags.").
Preview	Live preview.

Click **Save to Library** (or **Update** when editing).

To add a company symbol to the library:

1. In your drawing tool, export the symbol as SVG. Remove fixed `width / height` attributes and keep the `viewBox` so that it scales cleanly.
2. Open **SVG Library** and click **Import SVG File**, or + **Add SVG** and paste the code.
3. Enter **Name** `Dosing_Skid` , choose + **New Category** `Custom` , set **Width** 160 and **Height** 120.
4. Click **Save to Library**.
5. In the Object Library, click **SVG Library**, and drag **Dosing_Skid** onto the canvas.

OptiZeus adds a missing or incomplete `xmlns="http://www.w3.org/2000/svg"` attribute automatically when it displays an SVG.

Note: Adding and editing library items is allowed for the `admin` and `operator` roles; deleting items only for `admin` . Library changes take effect immediately; they are not staged.

Important: A symbol placed on a screen is a copy. Editing or deleting the library item later does not change screens where it is already placed. For symbols that must be updated centrally, build a super object (Chapter 13).

11.6 Symbols from other HMI/SCADA projects

Configuration > **Admin** contains an importer for projects exported from another HMI/SCADA product (Import from another HMI/SCADA project). It reads tag export files (CSV/text) and XML exports of graphics objects and faceplates. Converted graphics become ordinary screens and objects that you can edit in the builder. The importer is described with tag import in Chapter 7.

11.7 Worked example: a pump symbol with state colours

1. From **Pumps**, drag a **Centrifugal Pump** onto the screen; set **Width** 80, **Height** 80.

2. In **Object Appearance** set **Fill Type Gradient**, **Gradient Start** #e2e8f0 , **Gradient End** #94a3b8 , direction ↓, **Stroke Color** #334155 , **Stroke Width** 2.
3. From **Text**, add a **Text Label** under the pump with **Text Content** P-201 , **Font Size** 12, **Align** Center, **Font Weight** Bold.
4. Select the pump and the label and click  **Group (2)**.
5. Add the running/fault animations of Chapter 12, Section 12.10.3.

12. Animations and dynamic properties

This chapter describes how objects on a synoptic screen react to live data and to the operator: animations (colour, visibility, blink, rotation, fill level, value display and others), expression bindings on text properties, click actions including writes to the PLC, navigation and popups, object popups, event scripts and custom attributes. It is for engineers who build screens; supervisors who validate screens will find the exact runtime behaviour of every animation type here. The chapter ends with complete worked examples: a tank level, valve state colours and a pump with running and fault feedback.

Important: This chapter documents what the runtime viewer actually does. Some animation types can be configured in the builder but have no effect at runtime; they are marked **Not implemented** in the tables and summarised in Section 12.4.

12.1 Concepts

Term	Meaning
Animation	A rule that changes one visual property of an object from the value of one tag. An object can have any number of animations; they are evaluated together on every tag update.
Linked tag (value)	The tag whose value an object displays (Value Only binding, Link Tag (Display Live Value) on text).
Click action	What happens when the operator clicks the object (Action tab). One primary action plus optional Additional Actions .
Object popup	A built-in popup window opened by a click (Popup tab).
Expression	A text property that starts with <code>=</code> and is computed from tag values (Section 12.8).
Security letter	A letter A-Z that the user's role must have to execute the object's action (Chapter 24).

Tag values are pushed to the viewer continuously; animations follow the tag without a separate refresh setting.

A boolean value is **true** when it is `true`, `1`, `'1'`, `'true'`, `'on'` or `'ON'`; anything else is false. Analog animations use the numeric value of the tag; a value that is not a number leaves the animation's property unchanged.

12.2 Adding an animation

To add an animation to an object:

1. Select the object on the canvas.
2. Open the **Anim** tab of the Properties panel.
3. Click **+ Add Animation**. A new entry appears in the animation list.
4. In **Tag / Instance**, select the tag. Type in **Search tags...** to filter the list. Tags are grouped as **Regular Tags** and **UDT Instances**. For a UDT instance, choose the **Member** in the second list (**-- Select Member --**). For an array tag, enter the indices in the `[i]`, `[j]`, `[k]` fields.
5. In **Animation Type**, select the type (Section 12.3). Default settings for the type are filled in.

6. Adjust the settings in the coloured box below the type.

7. Save the screen.

The animation list shows each animation as *type – tag*. Click an entry to edit it;  copies that animation to the animation clipboard; **x** removes it. The line "*Type: ... | Tag: ...*" under the type selector shows the internal type name and the tag.

If the object is in a group, a banner explains that you can configure one animation and then click the **Apply to all n group members** button under it to copy it to every member.

Tip: Drag a tag from the  **Tags** browser onto an object and choose **Animation** to add a ready-made animation in one step (Chapter 10, Section 10.5).

12.3 Animation types

The **Animation Type** list is grouped as **Boolean Animations**, **Analog Animations**, **Multi-State Animations** and **Alarm Animations**. The internal type name is given in brackets; you see it in the *Type:* line and in exported screen files.

12.3.1 Visibility (Show/Hide) [bool_visibility]

Shows or hides the object.

Field	Meaning	Default
Mode:	Bool (true/false): the condition is the boolean value of the tag. Analog (value compare): the condition is <i>tag value</i> <operator> <i>value</i> .	Bool
Operator (analog mode)	> , < , >= , <= , == , != , between	>
value / high (analog mode)	Comparison value; for between the low and high limits (inclusive).	—
Show / Hide	Show when TRUE (or Show when CONDITION MET) / Hide when TRUE (or Hide when CONDITION MET).	Show when TRUE

If several visibility animations are present, the object is hidden as soon as one of them says hide. A hidden object is not drawn and cannot be clicked. If the tag does not exist, the animation is ignored and the object stays visible.

12.3.2 Color Change (On/Off) [bool_color]

Switches the object's fill colour between two colours.

Field	Meaning	Default
Condition Type	Boolean (TRUE/FALSE), Equals (=), Not Equals (!=), Greater Than (>), Less Than (<).	Boolean
Compare Value	Value for the comparison types (e.g. 14). Equals/Not Equals compare numerically when both sides are numbers, otherwise as case-insensitive text.	—
Color when FALSE	Colour when the condition is false.	#808080
Color when TRUE	Colour when the condition is true.	#22c55e

Field	Meaning	Default
Blink when condition is TRUE + speed	Also blink while true: Slow (2s), Normal (1s), Fast (0.5s), Very Fast (0.25s) .	off
Text when FALSE / Text when TRUE	Optional texts shown on text objects (e.g. OFF / ON).	—
Text Size (px)	Size of those texts.	object font size

On text objects both the text colour and the background take the dynamic colour.

Important — limitation. The condition of **Color Change (On/Off)** is evaluated against the tag of the **first** animation in the object's list, and only when that first animation is itself a Color Change. The condition type and compare value are stored once per object. Therefore: - Put the **Color Change (On/Off)** animation **first** in the list, and use only one per object. - If it is not first, the object always shows **Color when FALSE**. - A Color Change created by dropping a tag on an object (Chapter 10, Section 10.5) is not registered as the object's first animation until the animation list is edited. After dropping, open the **Anim** tab, click the animation and re-select its tag or type once, then save. Use **Color by State** (Section 12.3.20) when you need several colour animations on one object.

12.3.3 Blink when True [bool_blink]

While the tag is true, the object's fill takes **Blink Color 1** and the object blinks (its opacity pulses between 100 % and 30 %).

Field	Meaning	Default
Blink Speed	Slow (2s), Normal (1s), Fast (0.5s), Very Fast (0.25s) (one cycle).	Normal
Blink Color 1	Fill colour while blinking.	#ff0000
Blink Color 2	Stored, but not used by the runtime (the blink is an opacity pulse, not an alternation of two colours).	#ffff00

12.3.4 Rotate when True [bool_rotation]

Continuous rotation, for fans, agitators and pump impellers.

Field	Meaning	Default
Direction Mode	On/Off — rotate when TRUE, stop when FALSE; Toggle — TRUE rotates CW, FALSE rotates CCW (always rotating); Reverse — TRUE = configured direction, FALSE = opposite (always rotating).	On/Off
Direction	Clockwise / Counter-clockwise.	Clockwise
Rotation Speed	Slow (4s/rotation), Normal (2s/rotation), Fast (1s/rotation), Very Fast (0.5s/rotation).	Normal

12.3.5 Disable (Gray Out) [bool_disable]

Field	Meaning	Default
Disable Mode	Disable when TRUE / Disable when FALSE.	when TRUE
Opacity	Opacity of the disabled object (0.1-0.9).	0.4

A disabled object is shown in grey scale at reduced opacity, shows a "not allowed" cursor and **cannot be clicked** — its click action and popup are blocked. Use it to interlock controls, for example to disable a **Start** button while `Pump_P201.Interlock` is true.

12.3.6 Flip Horizontal when True [bool_flip]

"Object will be mirrored horizontally (scaleX: -1) when tag is TRUE." No settings. Useful for a flow arrow or conveyor that changes direction.

12.3.7 Pulse/Throb when True [bool_pulse]

While true, the object pulses in size.

Field	Meaning	Default
Pulse Color	Colour of the pulse effect.	#f59e0b
Scale	Maximum size factor, 1.05-1.5.	1.1

12.3.8 Horizontal Pulse (Mixer Blade) when True [bool_horizontal_pulse]

"Blades shrink horizontally toward center then expand back, simulating a spinning mixer blade viewed from the side."

Field	Values	Default
Pulse Speed	Slow (2s/cycle), Normal (1.2s/cycle), Fast (0.8s/cycle), Very Fast (0.5s/cycle)	Normal

12.3.9 Glow Effect when True [bool_glow]

While true, a coloured glow (shadow) surrounds the object.

Field	Meaning	Default
Glow Color	Glow colour.	#3b82f6
Size	Glow radius in pixels, 5-50.	20

12.3.10 Value Display — Number [analog_value]

Shows the tag value as formatted text inside the object (mainly for text and value display objects).

Field	Meaning	Default
Decimal Places	0 (Integer) to 4 decimals.	2
Font Size	Pixels. Overrides the object's font size.	14

Field	Meaning	Default
Font Weight	Light (300) ... Bold (700).	Normal
Prefix	Text before the value.	—
Suffix / Unit	Text after the value, e.g. % or bar .	—
Text Color	Colour of the value.	light grey (#e0e0e0); dark text is used automatically on white objects

The value is shown as `XXXX` in red when the tag does not exist, when the PLC connection is lost or when the tag quality is bad. Internal tags (memory, simulation, calculated) are always shown.

Tip: Remember to add the space in the suffix: " %", not "%", if you want 45.2 %.

12.3.11 Value Display — String / Text [`string_value`]

Shows the tag value as it is, without number formatting — for string tags such as batch numbers or recipe names. Fields: **Font Size**, **Font Weight**, **Text Color**, with a **Sample text** preview.

12.3.12 Color Gradient [`analog_color`]

The fill colour is interpolated between two colours according to the value.

Field	Meaning	Default
Min Value / Max Value	Value range. Values outside are clamped.	0 / 100
Min Color / Max Color	Colours at the ends of the range.	#22c55e / #ef4444

Example: a temperature from 20 °C (green) to 90 °C (red) gradually turns orange as it rises.

12.3.13 Rotation (0-360deg) [`analog_rotation`]

The object is rotated in proportion to the value — for a valve position indicator, damper blade or gauge needle.

Field	Meaning	Default
Min Value / Max Value	Value range.	0 / 100
Start Angle / End Angle	Angle in degrees at Min and at Max.	0 / 360

The object rotates about its centre. It replaces the static **Rotation Angle** set in the Props tab.

12.3.14 Fill Level [`analog_fill`]

Draws a coloured fill over the object whose height (or width) is proportional to the value — the classic tank level.

Field	Meaning	Default
Min Value (0%) / Max Value (100%)	Tag values for empty and full.	0 / 100
Fill Direction	Bottom to Top, Top to Bottom, Left to Right, Right to Left.	Bottom to Top

Field	Meaning	Default
Fill Color	Fill colour (drawn at 70 % opacity so that the symbol remains visible).	#06b6d4

The fill is a rectangle over the object's full bounding box. For round tanks, size the object so that the fill looks right, or place a rectangle shaped to the tank's inside and animate that rectangle.

12.3.15 Opacity [`analog_opacity`]

Field	Meaning	Default
Min Value / Max Value	Value range.	0 / 100
Min Opacity / Max Opacity	Opacity at Min and Max (0 = invisible, 1 = opaque).	0 / 1

12.3.16 Text by Value Range [`analog_text`]

Shows a text chosen by the range the value falls into.

Field	Meaning	Default
Min Value / Max Value	Overall range (informative).	0 / 100
Value Ranges	Rows of Min , Max , Text ; + Add Range adds a row. The first row whose range contains the value (inclusive) wins.	0-50 Low , 50-100 High

If no range matches, the object's normal text is shown.

12.3.17 Border Color Gradient [`analog_border_color`]

Like **Color Gradient**, but colours the object's border. Fields: **Min Value**, **Max Value**, **Min Color**, **Max Color**, border width (default 2 px).

12.3.18 Glow Intensity [`analog_glow`]

The glow radius grows from 0 at **Min Value** to the maximum glow (default 30 px) at **Max Value**. Fields: **Min Value**, **Max Value**, **Glow Color**.

12.3.19 Animation Speed [`analog_speed`] — *CSS Animation Speed by Value*

Changes the speed of the object's running animations (rotation, pulse, blink) from the value, for example making a fan turn faster as the drive speed rises.

Field	Meaning	Default
Min Value / Max Value	Value range.	0 / 100
Slowest (s)	Cycle time at Min Value.	5
Fastest (s)	Cycle time at Max Value.	0.2

Combine it with **Rotate when True** on the same object.

12.3.20 Color by State [`multistate_color`]

Maps integer values to colours. The value is rounded to the nearest integer.

Field	Meaning	Default
States	Rows <i>value</i> → <i>colour</i> ; + Add State adds a row.	0 grey, 1 green, 2 amber, 3 red

A value without a row leaves the object's normal colour.

12.3.21 Image by State [`multistate_image`]

Maps integer values to pictures. Each row: value and **Image URL or SVG path**. Use it to swap a valve symbol between open, closed and travelling images.

12.3.22 Text by State [`multistate_text`]

Maps integer values to texts (**Label text**). Default: 0 `OFF`, 1 `ON`, 2 `FAULT`. On text objects the state text replaces the normal text.

12.3.23 Scale, Move Horizontal, Move Vertical [`analog_scale`, `analog_move_x`, `analog_move_y`] — Not implemented

These types appear in the **Animation Type** list, but the Viewer Builder offers no settings for them and the runtime viewer ignores them. Do not use them on screens.

12.3.24 Color by Alarm State, Blink on Alarm [`alarm_color`, `alarm_blink`] — Not implemented

The builder shows settings for these two types — **Alarm State Color** with **Normal**, **High/Low**, **HiHi/LoLo** and **Bad Quality** colours, and **Blink on Alarm** with **Blink on Alarm Level (Any Alarm, High/Low (Warning), HiHi/LoLo (Critical))** and **Blink Speed** — but the runtime viewer does not apply them to screen objects.

To show alarm states on a screen, use one of these instead:

- A **Color by State** animation on an integer alarm-state tag produced by the PLC or by a calculated tag (Chapter 9).
- **Color Change (On/Off)** with **Greater Than** on the analog value and the alarm limit as **Compare Value**.
- The **Current Alarms** or **Alarm Panel (Live)** components, or the alarm banner in the header (Chapter 17).

12.4 Summary of runtime support

Type	Screen objects	Children of super objects (Chapter 13)
Visibility, Color Change, Blink, Rotate, Disable, Flip, Pulse, Horizontal Pulse, Glow	Yes	Yes
Value Display (Number)	Yes	Yes
Value Display (String)	Yes	—
Color Gradient, Rotation (analog)	Yes	Yes
Fill Level, Opacity, Text by Value Range, Border Color Gradient, Glow Intensity, Animation Speed	Yes	See Chapter 13

Type	Screen objects	Children of super objects (Chapter 13)
Color / Image / Text by State	Yes	See Chapter 13
Scale, Move Horizontal, Move Vertical	No	See Chapter 13
Color by Alarm State, Blink on Alarm	No	See Chapter 13

12.4.1 Evaluation order and conflicts

All animations of an object are evaluated together:

1. Visibility animations are evaluated first. If the object is hidden, nothing else is done.
2. The other animations are applied in list order. When two animations set the same property (for example two colour animations), the one **later** in the list wins.
3. **Disable** overrides the opacity set by **Opacity**.
4. **Value Display**, **Text by Value Range** and **Text by State** all set the text of a text object; **Text by State** has priority, then **Text by Value Range**, then **Value Display**.

12.5 Tag bindings list

Below the animation settings, the **Tag Bindings (n)** block ("*Bind multiple tags to this object for display or control.*") lists the tags linked to the object for display. + **Add Tag** adds a binding. For each binding you set a label, the tag (-- **Select Tag** --), **Array Indices (0-based)** and the attribute to show:

Attribute	Shows
Value (EU Value)	The engineering value.
Tag Name	The tag name.
Label	The tag's label (or its name).
Comment	The tag's comment.
Unit	The engineering unit.
Min Range / Max Range	The tag's engineering range.
Alarm State	The current alarm state (<code>normal</code> , ...).
Quality	The tag quality.
Status	The tag status.

The first binding is the object's linked tag.

12.6 Click actions (Action tab)

12.6.1 Primary action

Select the object, open the **Action** tab and choose the **Click Action**.

Click Action	Settings	Runtime behaviour
No Action	—	Nothing.
Navigate to Screen	Target Screen	Shows the target screen in the same viewer (in a pop-out window, replaces the screen in that window).
Open Popup	Popup Screen, Popup Size (Width, Height), Popup Title	Opens the popup screen as a floating window (Chapter 10, Section 10.9). If the chosen screen is not of type popup, it is shown in a modal window.
Close Popup	—	Closes the popup that contains the object.
Open Faceplate	—	Opens the tag faceplate (Chapter 13) of the object's linked tag or action tag. A double-click also opens it.
Write Tag Value	Target Tag, Array Indices (0-based), Input Method (Fixed Value with Value to Write, or Keyboard Input)	Writes the value to the tag (Section 12.6.2).
Toggle Tag (Bool)	Target Bool Tag	Writes 0 if the tag is currently true, otherwise 1.
Download Recipe	Select Recipe	Opens the recipe selection window listing all enabled recipes; the operator confirms the download to the PLC (Chapter 20).
Upload Recipe (PLC → Recipe)	Select Recipe	Opens the upload window: read the current PLC values into an existing recipe or a new one.
Open Recipe Manager	—	Opens the recipe manager in a window.
Run Script (JavaScript)	the On Click Script of the Script tab	Runs the script in the operator's browser (Section 12.9).
Run Python Script	Python Script Path, Arguments (optional)	Runs a Python script on the server. {tagName} and {value} in the arguments are replaced by the object's linked tag and its value.
Run Python (Inline Code)	Python Code	Intended to run inline Python on the server — see the note below.
Run System Command	Command, Arguments (optional)	Runs a program on the server.
Open External URL	URL (accepts an expression)	Opens the URL in a new browser tab.
Show Visual Report	Report Type (Standard / Batch), Select Report Layout, Parameter Mappings (Fixed, Tag Value, DS Field), and for batch reports Batch Datalog, Batch Number Mode, Batch Number Tag	Opens the report with the parameters resolved (Chapter 21).
Log Tag Snapshot (Batch)	Batch Datalog	Writes one row of the batch datalog's tags into the running batch (Chapter 20).

Click Action	Settings	Runtime behaviour
Attach Recipe to Batch	Batch Datalog, Recipe	Stores the recipe's parameters in the running batch's header for use in reports.
Shutdown Client	—	Desktop (Electron) client only: asks "Shutdown Application?" and closes the client.
Windows Access (Restore Desktop)	—	Desktop client only: asks for confirmation and re-enables the Windows key, taskbar, Alt+Tab and system shortcuts that kiosk mode disables.

Important: **Run Python Script**, **Run Python (Inline Code)** and **Run System Command** are executed by the server only for users with the `admin` role. For any other user the request is refused and nothing happens on screen. Output and errors are written only to the browser console, not shown to the operator. Characters `;` & `|` `$`` are removed from paths and arguments. The time limit is 30 seconds.

Note: In version 2.6.1.26 **Run Python (Inline Code)** is refused by the server ("Script path or command is required") because the client does not send a script path with inline code. Use **Run Python Script** with a script file instead.

12.6.2 Writing values to the PLC

WARNING: A **Write Tag Value** or **Toggle Tag (Bool)** action writes directly to the PLC of a running plant. Protect every control object with a security letter (Section 12.6.4), use **Require Confirmation** for actions that start, stop or open equipment, and test on a simulator or in a commissioning window before deploying.

Write Tag Value behaves as follows:

- **Fixed Value** with a **Value to Write**: the value is written immediately when the object is clicked (after confirmation, if required).
- **Keyboard Input**, or **Fixed Value** with an empty value: the operator is asked for the value. On a touch-screen client (touch screen mode on) or with **Keyboard Input**, the on-screen keyboard opens ("On-screen numeric keyboard will open when operator clicks this object"): a numeric keypad with the tag's min/max range and unit for numeric tags, a full keyboard for string tags. Otherwise a simple input window opens. Cancelling, or confirming the unchanged value, writes nothing.
- The value must lie inside the tag's engineering range (min/max). Otherwise the write is refused with "Value Out of Range!".
- If the station requires a reason for actions, the operator must choose or type a reason before the write. If it requires login confirmation, the operator must enter a user name and password; the confirming user is recorded with the write (Chapter 25).
- The server accepts writes only from users with the `operator` or `admin` role, and only with a valid licence or during the trial period. A refused write shows "Write Failed!" with the reason.
- A write to a PLC tag succeeds only when the PLC accepted the value. If the PLC connection is down or the PLC rejects the value, the screen shows "Write Failed!" with *Not written: the PLC connection "<name>" is not connected or rejected the write.* and the value is not changed (Chapter 16, 16.4.5).

Important: Toggle Tag (Bool) and the **Additional Actions** write directly: they do **not** ask for a reason or login confirmation, even when the station requires them, and **Toggle Tag (Bool)** ignores array indices. Where writes must be justified and attributable (21 CFR Part 11), use **Write Tag Value** with a fixed value of **1** or **0** instead of **Toggle Tag (Bool)**.

12.6.3 Additional Actions

"Run more actions in sequence after the primary Click Action above." Click + **Add Additional Action** (a primary action must be set first). Each additional action (**Action 2**, **Action 3** ...) has a **Type**:

Type	Settings
Write Tag	Target Tag (type or pick), Value (number or text)
Toggle Tag (Bool)	Target Tag
Close Popup	—
Open Popup	popup screen
Navigate to Screen	target screen (-- Select Screen --)

The additional actions run in order immediately after the primary action has been started. They do not ask for input, reason or confirmation.

Important: If the primary action waits for operator input (a keyboard), the additional actions still run immediately — before, and even without, the operator's entry. Do not combine a keyboard-input write with additional actions that depend on the new value.

12.6.4 Security and confirmation

Field	Meaning
Security Access Control — Security Letter (A-Z)	"User's role must have this security letter enabled to execute action." No Security Required or a letter. A user without the letter who clicks the object sees "Access Denied: You need security letter "X" to perform this action.". The check also blocks the object's popup.
Security — Require Confirmation	Asks for confirmation before the action.
Confirmation Message	The question shown; accepts an expression. Default "Are you sure?".

Security letters are assigned to roles in Chapter 24.

Note: The **Popup Security Letter (A-Z)** (Popup tab) and **Script Security Letter (A-Z)** (Script tab) are stored with the object but are not checked by the runtime in version 2.6.1.26. Protect popups and scripts with the **Security Letter (A-Z)** of the **Action** tab, which is checked before any click behaviour.

12.7 Object popups (Popup tab)

Tick **Enable Popup on Click** to open a built-in popup when the object is clicked.

Field	Meaning
Popup Type	Tag Faceplate, Custom Faceplate, Trend Chart, Info Panel, Custom HTML.
Tag for Faceplate	Tag shown in a Tag Faceplate (if empty, the object's linked tag).
Custom Faceplate Design	Opens the faceplate editor for a Custom Faceplate (Chapter 13).
Faceplate Title	Title of the popup.
Tags for Trend / Time Range	For Trend Chart : tags (Ctrl/Cmd for multiple) and Last 15 Minutes ... Last 7 Days .
Info Text	Text for Info Panel .
Custom HTML	HTML for Custom HTML .
Popup Size	Width / Height .

The popup can be dragged by its title bar.

Note: In version 2.6.1.26 the **Trend Chart** popup type shows only the list of selected tags, not a chart. Use a popup screen containing a **Real-Time Chart** component instead.

Important: An enabled object popup replaces the click action: the **Click Action** of the object is not executed. Custom HTML is inserted as-is; only administrators should author it.

12.8 Expressions

Several text properties accept an **expression**: a formula that starts with `=` and is recomputed from live tag values. Fields that accept expressions show the placeholder "(or =expression)", a ? help button and a live preview (→ result) or an error (⚠ message) under the field.

Fields with expression support: **Text Content** and **Label** (text objects), **Tooltip (hover text)**, **Popup Title**, **URL** (Open External URL) and **Confirmation Message**. At runtime every top-level text property of an object whose value begins with `=` is evaluated.

12.8.1 Syntax

Element	Syntax
Tag	Tank_Level1 ; UDT member Tank_101.Level1 ; name with spaces or special characters in brackets: [Value Display]
Literals	numbers 12.5 , strings 'text' or "text" , true , false , null
Arithmetic	+ - * / % (+ joins text when one side is text; division by zero gives NaN)
Comparison	== != > < >= <=
Logic	`&&` !`
Conditional	condition ? a : b
Math functions	round(x, digits) , min(...) , max(...) , abs , floor , ceil , sqrt , pow(x, y)

Element	Syntax
Transforms	<code>scale(v, inMin, inMax, outMin, outMax), clamp(v, lo, hi), lerp(a, b, t), map(v, k1, v1, k2, v2, ..., default)</code>
Text functions	<code>lower, upper, len, trim, startsWith, endsWith, contains, replace(s, a, b), left(s, n), right(s, n), substring(s, start, end), concat(...)</code>
Type functions	<code>toNumber, toString, toBool, isNaN, isNull</code>
Format	<code>format(n, digits)</code>
Time	<code>now()</code> (milliseconds), <code>today()</code> , <code>date()</code> (YYYY-MM-DD), <code>time()</code> (HH:MM:SS)
Other	<code>if(cond, a, b)</code>

An unknown tag evaluates to `null`. An expression with a syntax error is not applied: the property keeps its stored text. The evaluator does not execute JavaScript and has no access to anything but tag values.

12.8.2 Examples

```
= "Level: " + format(Tank_101.Level, 1) + " %"
= Pump_P201.Running ? "RUNNING" : (Pump_P201.Fault ? "FAULT" : "STOPPED")
= map(Valve_XV101.State, 0, 'Closed', 1, 'Open', 2, 'Travelling', '?')
= "Flow " + round(scale(FT_101_Raw, 0, 27648, 0, 250), 1) + " m3/h"
= upper(trim(Batch_ID))
```

Note: Colours set with the colour pickers cannot hold expressions. For colours driven by values use the colour animations of Section 12.3.

12.9 Event scripts and attributes

12.9.1 Script tab

The **Script** tab ("*Event Scripts — Add JavaScript code to execute on events.*") holds:

Field	Runtime behaviour in 2.6.1.26
Script Security Letter (A-Z)	Stored; not checked (see Section 12.6.4).
On Click Script	Executed when the object's Click Action is Run Script (JavaScript) . The variables <code>object</code> (the object), <code>tags</code> (all tags) and <code>value</code> (value of the linked tag) are available.
On Value Change Script	Stored; not executed .
On Load Script	Stored; not executed .

WARNING: Despite the note in the tab ("*Scripts run in a sandboxed environment*"), the **On Click Script** runs in the operator's browser with the operator's session and full access to the page. Scripts must be written and reviewed by administrators only. Server-side automation belongs in the script engine of Chapter 9.

```
// On Click Script – log the current level to the browser console
console.log('Tank_101 level is', tags['Tank_101.Level']?.value);
```

12.9.2 Attr tab

Custom Attributes ("*Add custom key-value attributes for this object.*") stores free **Key / Value** pairs with the object, for documentation or for use by scripts (`object.customAttributes`). The **Object Metadata** block shows the object's internal id, type, position and size.

12.10 Worked examples

The tags used below:

Tag	Type	Meaning
Tank_101.Level	REAL, 0-100, unit %	Tank level; alarms H 80 %, HH 90 %, L 20 %, LL 10 % (Chapter 17)
Valve_XV101.Open	BOOL	Open limit switch
Valve_XV101.Closed	BOOL	Closed limit switch
Valve_XV101.Cmd	BOOL	Open command (write)
Pump_P201.Running	BOOL	Running feedback
Pump_P201.Fault	BOOL	Fault feedback
Pump_P201.StartCmd / Pump_P201.StopCmd	BOOL	Start and stop commands (pulses reset by the PLC)
Pump_P201.State	INT	0 stopped, 1 running, 2 fault (calculated or PLC)

12.10.1 Tank level with value and colour

1. Select the **Vertical Tank** Tank_101 . In **Anim** click + **Add Animation**, choose **Tag / Instance** Tank_101 , **Member** Level , **Animation Type** Fill Level.
2. Set **Min Value (0%)** 0 , **Max Value (100%)** 100 , **Fill Direction** Bottom to Top, **Fill Color** #0ea5e9 .
3. Add a **Value Display** text under the tank. + **Add Animation**, tag Tank_101.Level , type **Value Display — Number, Decimal Places 1, Suffix / Unit** % , **Font Size** 16, **Text Color** #0f172a .
4. To show high level, add to the same text a second animation **Color Gradient** with **Min Value** 80, **Max Value** 90, **Min Color** #f59e0b , **Max Color** #dc2626 . Below 80 % the colour stays at amber; to keep it neutral below 80 %, use instead a **Color Change (On/Off)** as the *first* animation of a separate background rectangle, **Condition Type** Greater Than, **Compare Value** 80 , **Color when FALSE** #e2e8f0 , **Color when TRUE** #f59e0b , with **Blink when condition is TRUE** Normal (1s).
5. Optionally set the text's **Tooltip (hover text)** to `= "Tank 101: " + format(Tank_101.Level, 1) + " % – HH 90 %"` .

12.10.2 Valve with state colours and control

1. Select the **Ball Valve** `xv-101` . Add **Color by State** on `Valve_XV101.State` if the PLC provides a state integer (0 closed grey, 1 open green, 2 travelling amber, 3 fault red).
2. Without a state integer, use two animations: first **Color Change (On/Off)** on `Valve_XV101.Open` (**Color when FALSE** `#9ca3af` , **Color when TRUE** `#22c55e`), then **Blink when True** on a calculated tag `Valve_XV101.Travel` (true when neither limit switch is made).
3. In **Action** set **Click Action Write Tag Value**, **Target Tag** `Valve_XV101.Cmd` , **Input Method Fixed Value**, **Value to Write** `1` , **Security Letter (A-Z)** `B` , **Require Confirmation** on, **Confirmation Message** `"Open valve XV-101?"` .
4. Put a second object (a **Stop Button**) next to it with **Value to Write** `0` for closing.

12.10.3 Pump with running and fault feedback

1. Select the **Centrifugal Pump** `P-201` and add **Color by State** on `Pump_P201.State` : `0` → `#9ca3af` , `1` → `#22c55e` , `2` → `#dc2626` .
2. Add **Blink when True** on `Pump_P201.Fault` , **Blink Speed Fast (0.5s)**, **Blink Color 1** `#dc2626` .
3. Put a **Rotation Indicator** (category **Animation**) on the pump impeller and give it **Rotate when True** on `Pump_P201.Running` , **On/Off, Clockwise, Normal (2s/rotation)**.
4. Add a text label with **Text by State** on `Pump_P201.State` : `0` `STOPPED` , `1` `RUNNING` , `2` `FAULT` .
5. Add a **Start Button** with **Write Tag Value** `Pump_P201.StartCmd = 1` , security letter `B` , confirmation, and a **Disable (Gray Out)** animation on `Pump_P201.Running` (**Disable when TRUE**) so that it cannot be pressed while the pump runs. Add a **Stop Button** writing `Pump_P201.StopCmd = 1` with **Disable when FALSE** on `Pump_P201.Running` .
6. On the pump symbol set **Click Action Open Popup** with the popup screen `Popup_Pump_P201` (Chapter 10, Section 10.9).
7. Save and deploy, then test with the simulator: set `Pump_P201.State` to 0, 1 and 2 and check colours, text and blink.

12.11 Troubleshooting animations

Symptom	Cause and remedy
Value shows <code>XXXX</code> in red	The tag does not exist, the PLC is disconnected, or the quality is bad. Check the tag name (Find & Replace Tag) and the connection (Chapter 6).
Color Change always shows the FALSE colour	It is not the first animation of the object, or it was created by dropping a tag. See Section 12.3.2.
Alarm colour animation does nothing	Color by Alarm State and Blink on Alarm are not implemented (Section 12.3.24).
Scale/Move animation does nothing	Not implemented (Section 12.3.23).
Click does nothing	Enable Popup on Click is on without a popup type; a Disable animation is active; or the action is a script/command and the user is not an administrator.
"Access Denied: You need security letter ..."	The user's role lacks that letter (Chapter 24).
"Value Out of Range!"	The value is outside the tag's min/max.

Symptom	Cause and remedy
Animation stopped after ungrouping	Ungroup removes animations and actions (Chapter 10, Section 10.6.5).

13. Super objects, faceplates and templates

This chapter explains super objects — reusable graphic templates with their own animations, actions and popup faceplates — and how they are bound to tags and UDT instances when you place them on screens. It also describes visual variants, super object instances, the faceplate editors and the standard tag faceplate. It is written for engineers who build a project's object library; operators see the result as equipment symbols and faceplates (Chapter 16).

13.1 Concepts

Term	Meaning
Super object	A named template: a small canvas of child objects (shapes, symbols, texts) plus animations, actions, attributes and popups that refer to tag variables instead of real tags.
Tag variable	A placeholder used inside the super object. Two kinds: a Tag variable (tag1 ... tag8), bound to one real tag, or a UDT variable, bound to a UDT instance, whose members the animations use.
Placed super object	A super object dropped on a synoptic screen, with its own position, size and tag bindings (variable → real tag or instance).
Super object instance	A pre-bound configuration saved in the list of instances (name, label, unit, bindings), which can be dragged onto screens ready-bound.
Variant	An alternative drawing of the same super object (for example horizontal and vertical valve) sharing the same variables, animations and actions.
Faceplate	A popup window that shows values and controls for the equipment represented by an object.

The super object definition is stored once. Screens store only a reference to it plus the per-placement data (position, size, rotation, bindings, variant, datalog link, security). The viewer rebuilds every placed super object from the current definition when a screen is loaded.

13.2 The super object list

Open **Config > Viewer Builder > Super Objects**.

Element	Function
Import Super Object	Imports a super object JSON file.
+ Create Super Object	Opens the super object builder with an empty template.
Card	Name; a badge TAG , UDT or MIXED showing which kind of variables its animations use; the type (e.g. <i>pump</i>); a PENDING ... badge when staged; the counts Objects , Animations , Instances .
Edit	Opens the builder.
Export	Downloads the super object as JSON.
	Duplicates the super object as <name>_Copy .

Element	Function
+ Instance	Creates a super object instance (Section 13.6). Disabled with "Add animations to create instances" while the super object has no animations.
×	Deletes the super object ("Are you sure you want to delete this super object?").

Below, **Super Object Instances** lists the saved instances with **Edit Bindings** and **Delete**.

Creating, changing and deleting super objects and instances is reserved for the `admin` role and follows the project's staging rule: when staging is on you see "Super object update has been staged for deployment" and must deploy the change (Chapter 5).

13.3 The super object builder

The builder is a smaller version of the Viewer Builder (Chapter 10) with additional tabs for the template logic.

13.3.1 Header

Element	Meaning
Title	Create Super Object or Edit Super Object .
Name	Enter Super Object name... — required ("Please enter a name for the Super Object").
Type	Custom, Pump, Tank, Valve, Motor, Sensor, Indicator, Compressor, Heat Exchanger . Descriptive only.
Mode indicator	Editing Main Canvas or Editing Popup (while you draw a popup canvas).
Variant tabs	One button per variant; + New ,  Clone , × (Section 13.5).
Hide Toolbars / Show Toolbars	Hides the side panels.
Export / Import	JSON export/import of the super object.
 Sync Instances	Pushes the saved definition to every placed copy on every screen (Section 13.4.4).
Cancel	Leaves without saving (with an unsaved-changes prompt).
Create Super Object / Update Super Object	Saves.

13.3.2 Canvas and toolbar

The left panel is an **Object Library** with the categories Tanks, Pumps, Compressors, Valves, Heat Exchangers, Instruments, Shapes, Piping, Indicators (gauge, digital display, real-time trend, history trend, input value) and Text (Text Label, Value Display, Title), plus the **SVG Library**.

The toolbar has select and pan tools, zoom and **Reset**, **Undo (Ctrl+Z)** / **Redo (Ctrl+Y)**, **Snap to Grid** (on, 10 px), the canvas size, **Align** (Left, Center H, Right, Top, Center V, Bottom), **Size (Same Width, Same Height, Same Size)**, **Group selected objects**, **Ungroup this object**, **Duplicate (Ctrl+D)** and **Delete (Del)**.

Canvas size presets: **Small (200x150)**, **Faceplate (300x200)**, **Default (400x300)**, **Medium (500x400)**, **VGA (640x480)**, **SVGA (800x600)**, **XGA (1024x768)**, **HD 720p (1280x720)**, **Full HD (1920x1080)**, **QHD**

(2560x1440), Custom... (width × height). Draw the template to fill the canvas: when the super object is placed on a screen and resized, its content is scaled from the canvas size to the placed size.

13.3.3 Tabs

Tab	Content
Properties	Properties of the selected child object: Name, Tooltip, Type, Position & Size , colours, text.
Animations	Animations of the selected child object, referring to tag variables (Section 13.3.4).
Actions	Click actions of the selected child object, including per-object popups (Section 13.3.5).
Attributes	Custom properties of the super object: name, Type (Number, String, Boolean, Color, Tag Reference (for animations)), Default, Description . <i>"Attributes are custom properties that can be set when using this Super Object."</i>
Data	Data Variables — named references to a Datalog, Tag Reference or Expression , with Display Name, Description and Required ; they are bound in the screen builder's Data tab (Section 13.4.5).
Scripts	Python Scripts with triggers On Initialize, On Update (every scan), On Click, On Value Change, Enabled and Security Level .
Popup	Popup Designer : lists the child objects that have popup actions; click one to edit its faceplate.

Note: In version 2.6.1.26 the **Python Scripts** of a super object are stored but not executed by the runtime or the server. Put logic in the server script engine (Chapter 9) or the PLC.

13.3.4 Animations on child objects

Select a child object, open **Animations** and add an animation. The type list is grouped as  **Boolean (On/Off)**,  **Analog (Range)**,  **Multi-State** and  **Alarm** and uses the same types as screen objects (Chapter 12, Section 12.3), plus **Visibility on Alarm**.

Each animation has a **Variable Type**:

Variable Type	Fields
Tag (tag1-tag8)	Tag Variable: tag1 ... tag8 . At placement, each tag variable is bound to one real tag.
UDT (User Defined Type)	Select UDT (the UDT type), then UDT Member (Attribute): Value (default) , an Instance Property (Instance Label (Text), Instance Unit (Text), Instance Description (Text), Instance Name (Text)) or a UDT Member of that UDT. Member Details shows the member's label, unit, type, default, description and range. <i>"When placing in synoptic, bind to a UDT instance (e.g., TT-0-01)."</i>

An **Attribute Source** of **Standard** reads a tag attribute (value, unit, limits, alarm state, quality, timestamp, status); **UDT Member** offers common members (Eu_value, Status, Manual, AL_High_High ... AL_Low_Low, their _EN enables, Scale_High, Scale_Low, Raw_value) or a custom member name.

The **Value Display** animation of a super object additionally offers a text format (**# (Integer), #.# (1 decimal) ... #,###.## (thousands + 2 dec), % (Percentage), Scientific, Text (as-is)**), prefix, suffix, unit from tag, text colour, and an option to colour the value by alarm state (HH red, H amber, L blue, LL purple). A value that cannot be resolved is shown as **---** (or the configured bad-quality text).

Important: The runtime support for super object children is narrower than the builder's type list. Confirmed at runtime: all **Boolean** types, **Value Display**, **Color Change (Gradient)** and **Rotation (Analog)**. Treat the other analog, multi-state and alarm types as unsupported on super object children in version 2.6.1.26, and test every animation on a screen before release. Where you need fill level or multi-state colour, place a normal screen object on top of the super object and animate it directly (Chapter 12).

13.3.5 Actions on child objects

Actions attaches click actions to the selected child object: **Set Tag Value**, **Toggle**, **Increment**, **Decrement**, **Open Popup**, **Navigate**, **Run Script**. The target is a tag variable (**Tag (tag1-tag8)**) or a UDT member (**UDT (User Defined Type)**). For writes, **Value** ("*leave empty to prompt user*") and **Use on-screen keyboard for input**. Every action has **Enabled** and **Security Level (Any User or a role)**.

For **Open Popup**, **Enable popup for this object** with **Popup Type Custom Faceplate** or **Canvas Designer**, **Popup Title**, **Width**, **Height** ("*This popup is specific to this object only*"). Edit the popup in the **Popup** tab or with the faceplate editor (Section 13.7.2); after saving the faceplate you are reminded "*Faceplate saved! Remember to click 'Update Super Object' to save all changes.*"

At runtime, clicking a child object that has an enabled popup action with a configured faceplate opens that faceplate for the placed super object, with all variables resolved to the bound tags.

13.4 Placing super objects on screens

13.4.1 Adding

In the Viewer Builder click **Super Objects** in the Object Library (hint "*Drag super objects to add smart components with tag bindings*"). Use **Search super objects...** and the **Grid view / List view** buttons. Drag the super object onto the canvas or click **Add to canvas**.

13.4.2 Binding tag variables

Select the placed super object and open the **Anim** tab. **Tag Variable Bindings** lists each variable of the super object:

- For a **Tag** variable, choose a tag in its list. A filter box above each list narrows long tag lists; **X** clears it ("*No tag or instance matches that search.*" when nothing matches).
- For a **UDT** variable, choose a UDT instance of the matching UDT type. All member references of the animations resolve automatically (`<instance>.<member>`).
- **Copy / Paste** copy a binding from one row to another; **Paste to all n TAGs** writes it to every tag row; **Clear** forgets the copied binding. With several placed super objects selected, **Paste** binds them all at once.

You can also drag a tag or UDT instance from the  **Tags** browser onto the placed super object; the menu **Bind tag to super-object variable**: asks which variable receives it (Chapter 10, Section 10.5).

Faceplate Variable Bindings ("*Link faceplate variables to actual tags in your system.*") binds the variables used by custom faceplates.

13.4.3 Visual variant and child offsets

- **Visual Variant** — "*Choose the visual layout for this instance (e.g., horizontal/vertical)*". Selecting a variant changes the drawing and, if the variant has a different canvas size, the object's size.

- **Child Position Offsets** — moves individual child objects of this placement by **dx** / **dy** (design pixels) without changing the definition; the reset button restores the original position.

13.4.4 Changing the definition

 **Edit Super Object** (Anim tab of a placed super object) opens the super object builder for the definition. *"Edits the master definition. After saving, all placed instances on every synoptic auto-sync."*

From the super object list, use  **Sync Instances** in the builder to push a saved definition to every placed copy. The sync first shows a preview (*"Push the current "<name>" definition to n placed instance(s) across m synoptic screen(s)?"*) and keeps per-placement position, size, rotation, tag bindings, datalog bindings and security settings. If a variable was renamed in the definition and a placement has exactly one existing binding, that value is copied to the new variable name. The sync cannot be undone.

Important: Save the super object before syncing (*"Save the Super Object first, then re-open it to sync instances."*). Renaming tag variables breaks existing bindings on screens; re-check the bindings after renaming.

13.4.5 Data tab

For a placed super object, the extra **Data** tab (**Data Configuration**) links a datalog: **Select Datalog** (**-- No Datalog Linked --**), **Datalog Information**, **Select Tag from Datalog** (*"Select the primary tag to use for history trends in this Super Object"*) and **Additional Tags for History Trend**. These feed the **History Trend** elements of the super object's faceplate.

13.5 Variants

Variants are alternative drawings of one super object. All variants share the same tag variables, animations and actions; each has its own child objects and canvas size.

Control	Function
Variant button	Switches the canvas to that variant. Double-click to rename.
+ New	Adds an empty variant (asks for Variant name:).
 Clone	Adds a copy of the current variant with new object ids and copies of its animations and actions, so the two variants can be edited independently.
x	Deletes the active variant (<i>"Delete this variant? This cannot be undone."</i>). The last variant cannot be deleted.

The first variant is the default. Placed objects without a selected variant, and placed objects whose variant has been deleted, use the default.

To build a valve with horizontal and vertical variants:

1. Create the super object `valve_OnOff` with the horizontal drawing (canvas 120 × 60) and animations on `tag1` (open feedback) and `tag2` (fault).
2. Click  **Clone**, name the variant `Vertical`.
3. Rotate and rearrange the drawing, set the canvas to 60 × 120.
4. Check that the animations of the cloned objects refer to `tag1` and `tag2` as in the original.
5. Click **Update Super Object**.

6. On a screen, place `Valve_OnOff` twice and select **Vertical** in **Visual Variant** for the second.

Note: Export in the builder exports the current canvas and the super object as last loaded; to transfer a super object with all its variants, export it from the super object list after saving.

13.6 Super object instances

An instance stores a complete binding of a super object under its own name, so that engineers can drop ready-bound equipment on screens (library category **Instances (n)**, grouped by super object, hint *"Drag instances to add pre-configured components with bound tags"*).

To create an instance, click **+ Instance** on the super object card. The dialog contains:

Field	Meaning
Name	Instance name, e.g. <code>Tank_TK101</code> .
Label	<i>"Display label for this instance"</i> , e.g. <code>Tank 101 Level</code> .
Unit	Engineering unit; Browse opens a searchable unit list.
Description	Optional.
Tag Variable Bindings	<i>"Map each tag variable to an actual tag in your system."</i> One list per variable (-- Select Tag --); bound rows are marked Bound .

Dropping an instance on a super object that is already bound to another instance asks *"This super-object is already bound to instance ... Replace with ...? Existing bindings for keys not present in the new instance will be kept."*

13.7 Faceplates

OptiZeus has three kinds of faceplate.

Faceplate	Opened by	Content
Tag faceplate	Open Faceplate click action or a double-click on an object with that action; Popup Type Tag Faceplate ; clicking a tag in the dashboard tag list	Standard window for one tag (Section 13.7.1).
Custom faceplate	Popup Type Custom Faceplate on a screen object	A small designed panel (Section 13.7.2).
Super object faceplate	Popup action of a super object child	A designed panel whose elements use the super object's variables (Section 13.7.2).

13.7.1 The tag faceplate

The tag faceplate shows, for one tag:

- The tag name and description, and a close button (✕, or click outside the window).
- The current value with unit and alarm state.
- **Min (1h)**, **Avg (1h)**, **Max (1h)** and a small **Trend (Last 1 Hour)** from the historian (*"No historian data in the last hour"* if the tag is not logged).

- **Alarm Limits: Low Limit, High Limit, Range.**
- **Write Value** with an input (**Enter value...**) — shown only when the faceplate is opened from the dashboard tag list by a user who may control the plant.
- **View in Historian** and **Close**.

Important: Faceplates opened from synoptic screens are read-only (no **Write Value** block). The **Write Value** block of the dashboard faceplate writes through `POST /api/tags/<name>/write` with the same roles, security areas and failure reporting as other writes (Chapter 16, Section 16.4.5); it does not ask for a reason or login confirmation. (Up to 2.6.1.24 this block used a route that does not exist and nothing was written.)

13.7.2 Custom faceplate editors

Custom Faceplate Editor (screen objects: Popup tab > **Custom Faceplate Design**) and the super object faceplate editor share the same design:

Area	Content
Elements tab	Element palette: Label , Tag Value , Tag Name , Tag Label (super object only), Tag Unit (super object only), Input Value , Button , Toggle , Slider , Gauge , Progress , Indicator , Trend , History Trend (super object only), Divider , Image (screen object only), Alarm ; the super object editor also has Shapes (Rectangle, Circle, Ellipse, Triangle, Line, Arrow, Polygon, Star, Pipe, Tank).
Variables tab	Faceplate variables: Variable Name , Display Name , Available Attributes (click to copy) ; Remove . "No variables defined. Add a variable to link external tags."
Settings tab (super object)	Faceplate width, height and appearance.
Canvas	Grid (default on, 10 px, snap on), multi-select, copy/paste, undo/redo. "Shortcuts: Ctrl+C/V/D, Ctrl+Z/Y, Del, Arrow Keys".

Element types:

Element	Purpose
Label	Static text.
Tag Value	Live value with unit and decimals.
Tag Name / Tag Label / Tag Unit	The tag's name, label or unit.
Input Value	Field in which the operator types a value that is written to the tag.
Button	Writes a value to a tag (e.g. Start = 1).
Toggle	ON/OFF switch with onText/offText and colours.
Slider	Writes a value between min and max.
Gauge / Progress	Graphic display between min and max.
Indicator	Coloured lamp for a boolean.
Trend	Short real-time trend (default 60 s).
History Trend	Trend from the datalog linked in the Data tab (Section 13.4.5).
Divider	Separator line.

Element	Purpose
Alarm	Alarm status of a tag with icon and text.

WARNING: Buttons, toggles, sliders and input fields in a faceplate write to the PLC. Give them the appropriate security settings and test them on a simulator first.

13.7.3 SensorFaceplate

The frontend contains an older sensor faceplate component that is not used by any screen in version 2.6.1.26. Use the tag faceplate or a custom faceplate.

13.8 Worked example: a pump super object bound to a UDT

Prerequisite: a UDT `Motor_UDT` with members `Running` (BOOL), `Fault` (BOOL), `Speed` (REAL, rpm), `StartCmd` and `StopCmd` (BOOL), and instances `Pump_P201` and `Pump_P202` (Chapter 7).

1. **Config > Viewer Builder > Super Objects > + Create Super Object.** Name `Pump_Std`, type **Pump**, canvas **Faceplate (300x200)**, then **Custom...** 120 × 140.
2. Place a **Centrifugal Pump** (100 × 100) at the top and a **Value Display** text below it.
3. Select the pump. **Animations > add Color Toggle (On/Off), Variable Type UDT (User Defined Type)**, UDT `Motor_UDT`, member `Running`, false grey / true green.
4. Add **Blink (On/Off)** on member `Fault`, fast, red.
5. Select the text. Add **Value Display** on member `Speed`, format **## (1 decimal)**, suffix `rpm`.
6. Select the pump again. **Actions > Open Popup, Enable popup for this object, Custom Faceplate**, title `Pump`, 300 × 200. Design the faceplate: a **Tag Value** for `Speed`, an **Indicator** for `Running`, an **Indicator** for `Fault`, **Button Start** writing 1 to `StartCmd`, **Button Stop** writing 1 to `StopCmd`.
7. Click **Create Super Object** (deploy if staging is on).
8. Open a screen, click **Super Objects**, drag `Pump_Std` twice onto the canvas.
9. Select the first copy; in **Tag Variable Bindings** choose the instance `Pump_P201`. Bind the second copy to `Pump_P202`.
10. Save the screen and check at runtime: colours follow `Running`, blinking follows `Fault`, the speed is displayed, and a click on the pump opens the faceplate for the correct pump.

Tip: When a third pump is added later, create its UDT instance and drop another `Pump_Std`; no drawing work is needed. If you change `Pump_Std`, run  **Sync Instances** and every pump on every screen is updated.

14. Layouts, viewers and navigation

This chapter describes how synoptic screens and other views are arranged on the operator's display: the dashboard layout modes, the Layout Builder with its panel types and header editor, per-user and per-station layouts (station layouts are keyed to the client's IP address), screen navigation panels, header navigation buttons, pop-out windows and kiosk operation. It also lists the settings of the data widgets that you can place directly on synoptic screens. It is for engineers and administrators who set up operator stations; supervisors will find how a station decides which layout it shows.

14.1 How the display is organised

After login, the main window consists of:

- The **header**: logo and title, navigation buttons, the current-alarms counter, user information, connection status, theme toggle, record button, help link and logout (which elements appear is configurable).
- The **dashboard** below it, whose arrangement is set by the **Layout Mode**:
 - **Custom Layout** — the panels designed in the Layout Builder (Section 14.3).
 - **Single Column, 2 Columns, 3 Columns, 4 Columns** — the built-in column arrangement of standard panels (alarms, real-time monitoring, synoptic viewer, navigation, trends ...). Column widths can be dragged with the mouse.

Other views (Alarms/Events, Trends, Historian, Screen Replay, KPI Dashboard ...) are opened with the header buttons and replace the dashboard until you return.

14.2 Display settings

Open **Config > Display** (heading **Display Settings**).

Section	Fields
Visual Layout Builder	 Open Layout Builder (Section 14.3).
Layout Mode	Cards Custom Layout, Single Column, 2 Columns, 3 Columns, 4 Columns . Selecting a card applies it immediately.
Panel Widths	Left Panel Width (px) (200-600), Center Panel Width (px) (300-1000), Right Panel Width (px) (200-600), Panel Gap (px) (0-50), Tag Columns (Auto, 1 Column ... 6 Columns) . Used by the column modes.
Panel Visibility	One checkbox per standard panel of the column modes.
Screen Recording	Show Record Button in Header — "Enables screen recording from the header toolbar. Records the entire application view as a WebM video." On by default.
Navigation Buttons	Header buttons (Section 14.6).
Footer	Save Display Settings ("Display settings saved for your account!") and Reset to Defaults ("Reset all display settings to defaults?").

Display settings are saved **for your user account** on the server. How user and station layouts combine is explained in Section 14.5.

14.3 The Layout Builder

The Layout Builder designs the **Custom Layout**: a set of panels placed on a grid covering the whole screen, plus the header.

It is opened from:

- **Config > Display >  Open Layout Builder** — edits your own layout.
- **Config > Admin > Clients & Shutdown > Connected Clients > Registered Clients > Edit Layout** — edits the layout of that station (banner "*Editing station layout: <id>*").
- **Config > Admin > Clients & Shutdown > Station Layouts > Edit.**

14.3.1 Window

Element	Function
Layout Builder / name field	Layout name (Layout Name , default <i>New Layout</i>).
 Panels /  Header	Switches between panel design and header design.
 Preview /  Edit Mode	Shows the layout without editing handles, or returns to editing.
 Presets	Layout Presets: replaces the panels with a preset (Section 14.3.4).
Cancel / Save Layout	Leave or save.

14.3.2 Canvas

- All positions and sizes are **percentages** of the screen (the canvas is labelled "*100% x 100% (Full Screen)*"). A layout therefore adapts to any monitor resolution.
- Zoom **+**, **-**, **Reset** (50-200 %).
- **Show Grid** (on) and **Snaps** (on) with grid size **5%** (default), **10%**, **20%**.
-  **Undo** (Ctrl+Z) and  **Redo** (Ctrl+Y / Ctrl+Shift+Z), 50 steps.
- Drag panels from the panel library onto the canvas. A new panel is 20 % × 20 % and titled with the panel type name. Drag to move; use the eight handles to resize (minimum 10 %; panels stay inside the canvas).
- With a panel selected:  **Duplicate** (Ctrl+D; copy offset by 5 % and titled (*Copy*)),  **Front**,  **Back**,  **Delete** (Del/Backspace).

14.3.3 Panel types

The panel library is organised in the categories **Core**, **Data**, **System**, **Navigation**, **Custom**. The **Runtime** column states what version 2.6.1.26 shows at runtime.

Panel type	Name	Library description	Runtime
alarm_panel	Alarm Panel	Active alarms display	Compact live alarm list (Chapter 17).

Panel type	Name	Library description	Runtime
<code>event_log</code>	Event Log	System event history	Compact live event list (Chapter 18).
<code>trend_chart</code>	Trend Chart	Real-time trend display	Real-time trend of the first four tags of the project. The Tags and Duration (h) settings are not used.
<code>synoptic_view</code>	Synoptic View	Process visualization	The current synoptic screen, stretched to fill the panel. The panel's Screen setting is not used: the panel shows the screen selected in the navigation (initially the last viewed screen, else the first non-popup screen).
<code>tag_list</code>	Tag List	Tag values table	Grid of all tags with live values; clicking a tag opens its faceplate.
<code>tag_grid</code>	Tag Grid	Tag values in grid format	Same as Tag List.
<code>gauge_panel</code>	Gauge Panel	Analog gauges display	Placeholder only.
<code>datalog_viewer</code>	Datalog Viewer	Historical data viewer	Placeholder only.
<code>system_status</code>	System Status	PLC/Server connection status	Connection state (Connected/Disconnected), number of tags and current alarms.
<code>user_info</code>	User Info	Current user information	User name and role.
<code>clock_widget</code>	Clock	Date and time display	Time of day. The panel does not tick by itself; it updates when the page redraws for other reasons.
<code>notification_center</code>	Notifications	System notifications	Placeholder only.
<code>screen_nav</code>	Screen Navigation	Synoptic screen navigator	Buttons for the synoptic screens (Section 14.4).
<code>quick_links</code>	Quick Links	Shortcut buttons	Placeholder only.
<code>breadcrumb</code>	Breadcrumb	Navigation breadcrumb	Placeholder only.
<code>html_widget</code>	HTML Widget	Custom HTML content	Placeholder only.
<code>iframe_widget</code>	IFrame Widget	Embed external content	Placeholder only.
<code>image_widget</code>	Image Widget	Display an image	Placeholder only.

Panel type	Name	Library description	Runtime
<code>text_widget</code>	Text Widget	Static text or label	Placeholder only.
<code>spacer</code>	Spacer	Empty space placeholder	Placeholder only.

"Placeholder only" means the panel shows the text "*Panel: <type>*" at runtime. Put such content on a synoptic screen instead (text, image, IP camera and chart objects, Chapter 11).

Note: Because a **Synoptic View** panel always shows the currently selected screen, a layout with two Synoptic View panels shows the same screen twice. To show two screens side by side, use a pop-out window (Section 14.7) on a second monitor.

14.3.4 Presets

Preset	Description	Panels (x, y, width, height in %)
Default Layout	Standard SCADA layout with alarms and synoptic	Process View (synoptic) 0,0,70,70; Alarms 70,0,30,50; Events 70,50,30,50; Trends 0,70,70,30
Monitoring Layout	Focus on alarms and system status	Current Alarms 0,0,50,60; Event History 50,0,50,60; System Status 0,60,30,40; Tag Values (tag list) 30,60,70,40
Analysis Layout	Focus on trends and data analysis	Trend Chart 1 0,0,60,50; Trend Chart 2 60,0,40,50; Historical Data (datalog viewer) 0,50,100,50

Choosing a preset replaces all panels and the layout name.

14.3.5 Panel properties

With a panel selected, the **Properties** pane shows (with nothing selected: "*Select a panel to edit its properties*"):

Section	Fields
Panel Info	Title , Type (read-only).
Position & Size	X (%) , Y (%) , Width (%) , Height (%) (minimum 5).
Panel Configuration	Type-specific: Screen (Synoptic View — not used at runtime), screen list and colours (Screen Navigation, Section 14.4), Tags / Duration (h) (Trend Chart — not used), HTML , URL , Image URL , Text / Font Size (custom widgets — not used).
Page Background	Type (Solid , Gradient , Transparent); Color with 12 swatches, or Start , End , Angle (default 180°).
Layout Description	Free text.

14.3.6 Header design (🎨 Header tab)

The **Header** tab shows a **Header Preview** and the **Header Settings**:

Group	Fields	Defaults
General	Show Header; Height (px) (40-120); Bg Type (Solid/Gradient/Transparent) with Bg Color or Grad Start, Grad End, Angle; Border Color	shown, 60 px
Logo & Title	Show Logo; Logo Width (px) (20-200), Logo Height (px) (20-80); Custom Logo Upload (max 2 MB) / Remove; Title Text; Title Color; Title Size (px) (12-36)	title <i>OptiZeus</i> , 20 px
Navigation Buttons	Show Nav; Button Color; Text Color ; per button: visible, icon, label, action, Admin, x; + Add Button	Alarms/Events, Trends, Historian, Config (admin)
Elements Visibility	Alarm Counter; Default Alarm Group (— None (show all groups) — or an alarm group / sub-group); Security Letter (A-Z) to change filter ("User's role must have this letter enabled to change the in-popup group filter."); User Info; Connection Status; Logout Button; Help Link	—
Header Instruments	"Display real-time tag values in the header bar": Show Instruments ; per instrument Select Tag... , custom label, style Value / Gauge / Bar, Label, Unit, Trend, BG, Text, W (80-300 px); + Add Instrument	off
Panel Menu Bars	"Control the visibility and appearance of panel title bars": Show Menu Bars, Height (px) (20-60), Background, Text Color	shown, 32 px

Button actions available in this tab: **Alarms/Events, Trends, Historian, Screen Replay, Shift Report, KPI Dashboard, OEE, Audit Trail, Batch Control, Smart Alarms, Notepad, SOE, Procedures, Voice, Config, Help.**

The custom header is used only when the **Layout Mode** is **Custom Layout** and **Show Header** is on; otherwise the standard header is shown.

Important: Header buttons that open synoptic screens and dropdown groups are created in **Config > Display > Navigation Buttons** (Section 14.6), not in this tab. The Layout Builder's button editor does not know these button types; after creating them in the Display tab, avoid editing the button list in the Layout Builder, or the groups may be changed.

14.3.7 Saving a layout

When you click **Save Layout** from **Config > Display**:

1. The layout is stored as your personal layout and the **Layout Mode** is set to **Custom Layout**.
2. OptiZeus asks: "Layout saved to your account. Also save as this station's layout? Station: <name>. Station layouts apply to ALL users who log in from this station."
3. Click **OK** to also store it as the layout of the current station (administrators only; otherwise "Station layout save failed"), or **Cancel** to keep it only for your account.

When the Layout Builder was opened for a station (**Edit Layout**), **Save Layout** stores the layout directly for that station ("Layout saved for station: <id>").

There is no library of named layouts and no assignment by role: each user has one layout, each station can have one layout, and the product defaults apply otherwise.

14.4 Screen navigation

14.4.1 Screen Navigation panel (custom layout)

A **Screen Navigation** panel shows one button per synoptic screen (popup screens are never listed). Its **Panel Configuration** contains:

Setting	Meaning
Screen Navigation list	"Check to show, drag to reorder. If none checked = show all (default)." All Screens (default order) or individual screens, which you can drag into the order you want.
Bg Type / Panel Bg	Panel background: Solid , Gradient (Gradient Start , Gradient End , Direction : Top → Bottom, Bottom → Top, Left → Right, Right → Left, Top-Left → Bottom-Right, Top-Right → Bottom-Left, Radial) or Transparent . Default #1a1a2e .
Button Bg, Button Text, Border, Active Btn	Button colours; the button of the displayed screen uses Active Btn .

At runtime:

- Buttons are laid out horizontally when the panel is wider than it is tall, otherwise vertically.
- Screens that belong to a dropdown group of the Display tab (Section 14.6) with **Navigation** ticked appear as expandable group buttons, including sub-groups.
- Buttons can show blinking badges for alarm groups or tags bound to them.
- ↗ ("Open in new window (multi-monitor)") opens the screen in a separate window (Section 14.7).
- The panel can float: ⌵ (*Undock (float)*) turns it into a tab at the screen edge that opens an overlay when clicked; ↗ docks it again. Clicking outside the overlay closes it. The docked/floating choice is remembered per user and panel in the browser. Neighbouring panels use the space of a floating navigation panel.

14.4.2 Navigation panel (column layouts)

In the column layout modes the standard **Navigation** panel shows a card per screen with its description and a button. The look of each screen's card and button is set in the Viewer Builder's **Navigation Button** tab (administrators only):

Field	Meaning	Default
Button Text:	Text of the button on the card.	View
Style:	Solid (one Button Color) or Gradient (Gradient Start , Gradient End , Gradient Direction : → Left to Right, ← Right to Left, ↓ Top to Bottom, ↑ Bottom to Top, ↘ Diagonal, ◎ Center Outward).	Solid #4299e1
Shape	Rounded , Square , Pill .	Rounded
Card:	Card background colour.	#ffffff

Each card also has a ↗ button for a pop-out window. With no screens the panel shows "No synoptic screens available".

Note: The **Screen Navigation** panel of a custom layout ignores these per-screen button settings; it uses the panel colours of Section 14.4.1.

14.4.3 Navigation inside screens

Buttons on synoptic screens with the **Navigate to Screen** action (Chapter 12) change the displayed screen. There is no separate "home" or "back" button: provide a navigation panel or navigation buttons on your screens.

14.5 Station layouts and user layouts

14.5.1 Client identity

Each browser or desktop client registers with the server when it starts. The server identifies a station by the **IP address** the request comes from and gives it the id `client_<ip>` with dots replaced by underscores (for example `client_192_168_1_20`). New stations are named *Station <ip>*. The id is kept in the browser, and the client sends a heartbeat every 60 seconds.

Important: Because the station id is derived from the IP address, give operator stations fixed IP addresses (or DHCP reservations). If a station's address changes, it is registered as a new station without a layout. Several browsers on the same PC share one station id; clients behind the same proxy or NAT address also share one id.

14.5.2 Which layout a station shows

At login the layout is chosen in this order:

1. The **station layout** of the client's station, if one exists.
2. Otherwise the user's own **user layout**.
3. Otherwise the product defaults.

Note: In version 2.6.1.26 the station layout is applied at login. When the page is reloaded without logging in again, the user layout is fetched and may override parts of the station layout. After changing a station layout, log out and in on that station.

14.5.3 Managing station layouts

Config > Admin > Clients & Shutdown (administrators):

Place	Function
Connected Clients > Registered Clients	Table Client Name (editable), Client ID , IP Address , Touch Screen , Last Seen , Actions: Set Layout (copies your current layout to that station), Edit Layout , Clear ("Station will use user/default layout."), Remove .
Station Layouts	Stations that have a layout: Station , IP , Nav Buttons , Updated , Actions (Edit, Remove) . The station you are using is marked THIS STATION .

Note: In the **Station Layouts** table of version 2.6.1.26 the **Station** column may show the raw client id and the **IP** column may show "-"; the layouts themselves are applied correctly.

To set up a control-room station:

1. On the target PC, start OptiZeus once so that the station registers.
2. On any PC, log in as administrator and open **Config > Admin > Clients & Shutdown > Connected Clients**.
3. Rename the station (e.g. `Control Room 1`) and tick **Touch Screen** if it has a touch monitor.
4. Click **Edit Layout**, choose the **Default Layout** preset, make the synoptic panel larger, set the header buttons, and click **Save Layout**.
5. Log in on the station and check the result.

14.6 Header navigation buttons

Config > Display > Navigation Buttons ("Quick toggle — check to show in header. For full customization (icons, labels, order) use Layout Builder → Header."):

- A grid of the 16 system views — Alarms/Events, Trends, Historian, Screen Replay, Shift Report, KPI Dashboard, Config (admin only), Audit Trail (admin only), Batch Control, Help, OEE, Smart Alarms, Notepad, SOE, Procedures, Voice. Tick a view to show its button. A binding list per button (**No binding**,  **Alarm Groups**, or a tag) adds a blinking badge.
- **Synoptic Screens** — "Add synoptic screens as header buttons or group them into dropdowns." One entry per screen.
- **+ Create Dropdown Group** — creates a **Dropdown Group**: with name and icon, **Header** and **Navigation** checkboxes (show in the header and/or in Screen Navigation panels), **+ Add item...** (views or synoptic screens), **+ Add Sub-Group**, and **Delete Group** ("Delete this group? Items will become standalone buttons.").

At runtime, a synoptic-screen button selects that screen and returns to the dashboard; a group opens a dropdown. Buttons marked admin-only are hidden from other users. Colours are set for all buttons together in the header design (Section 14.3.6).

Important: In version 2.6.1.26 each change in **Navigation Buttons** is also written to the current station's layout. An administrator configuring buttons at an operator station changes that station's layout. Configure navigation buttons from an engineering station, or re-check the station layouts afterwards.

14.7 Pop-out windows and multiple monitors

The ↗ button on navigation cards and navigation panel buttons opens the screen in a separate browser window (1280 × 900) addressed as `#screen=<id>`. Such a window shows only the synoptic screen, without header or navigation, and can be moved to another monitor and maximised. **Navigate to Screen** actions inside it change the screen of that window. If the browser blocks the window you see "Popup blocked. Please allow popups for this site to use multi-screen mode."

14.8 Kiosk and full-screen operation

Kiosk operation is a property of the user's **role** (**Config > Admin > Security > Roles > Kiosk Mode**: "When enabled, users with this role will run in fullscreen locked mode and cannot exit or access desktop", see Chapter 24). For users with a kiosk role:

- The browser switches to full screen after login.
- Esc, Alt+F4, Ctrl+W, Ctrl+Q, F11, Alt+Tab, the Windows key and right-click are blocked.

- Leaving the page asks "Are you sure you want to leave? You are in kiosk mode."
- If full screen is left, an overlay **KIOSK MODE** — "This workstation is configured for kiosk mode. You must operate in fullscreen mode." — with **Enter Fullscreen** covers the screen.

The desktop (Electron) client adds operating-system level restrictions; the **Windows Access (Restore Desktop)** click action (Chapter 12) lifts them.

Note: **Config > Admin > Clients & Shutdown > Browser Client** also has  **Enter Fullscreen**,  **Lock Fullscreen** and  **Enable Kiosk Mode** buttons. In version 2.6.1.26 the lock and kiosk buttons only store a setting in the current browser and do not enforce anything; use the role setting above.

14.9 Data widgets on synoptic screens

The **Components** category of the Viewer Builder (Chapter 11) places data widgets directly on a screen. Their settings in the **Props** tab:

Widget	Settings
Real-Time Chart / Historical Chart	Chart Type (Real-Time Chart, Historical Chart (Datalog)); Select Datalog (Datalog Details: name, log mode, available tags); Select Tags to Display ; Date Range / Range Type (Duration (Last X hours/days) with Duration and Unit Minutes/Hours/Days/Weeks, or Custom Date Range with Start Date & Time / End Date & Time); Refresh Rate (seconds) (default 5); Chart Colors (Background Color, Text Color). Without datalogs: "No Datalogs Created Yet" with instructions.
Alarm Panel (Live)	Max Alarms to Display ; Display Settings : Font Size (px) , Font Family , Row Height (Compact/Normal/Comfortable).
Event Log (Live)	Max Events to Display ; font, row height, Icon Size (px) , Show Severity Icon .
Historical Alarms	Max Alarms to Display ; Date Range Type (Relative Duration / Custom Date Range); Filter Status (All Alarms, Active Only, Acknowledged Only); font, row height, Header Background , Text Color , Header Text Color .
Historical Events	As above with Category Filter (All Categories, System, User, Alarm, Configuration, Communication) and Severity Filter (All Severities, Critical, Warning, Info).
Current Alarms	Table Title , Max Rows , Filter Severity (All Severities, Critical Only, High & Above, Medium & Above, Low & Above), Columns (Show Timestamp, Show Value, Severity Color Band, Show ACK Button, Blink Unacknowledged), display settings.
Weekly Scheduler	Title ; Tag Binding (Boolean) — "Tag will be 1 during scheduled times, 0 otherwise"; Show Current Time Indicator ; Compact Mode (6-hour blocks) ; colours (Active Hours Color , Inactive Hours Color , Current Time Indicator , Header Background).
SPC Control Chart	Title , Tag (Data Source) , Data Points , Unit , Show Control Limits (UCL/LCL) , Show Warning Limits (2-sigma) , Show Mean Line , Show Statistics Panel , Override Limits (Optional) (UCL Override, LCL Override, Target Override).
SPC Histogram	Title , tag, Data Points , Number of Bins , Unit , Show Normal Curve Overlay , Show Statistics Panel , Show Specification Limits (USL, LSL, Target) .
Capability Analysis	Title , tag, Data Points , Unit , Specification Limits (Required) (USL, LSL, Target (Optional)), Show Capability Gauges (Cp, Cpk, Pp, Ppk) , Show Process Distribution Visual , Show Statistics Panel .

Widget	Settings
OpAnalogGauge	Title , tag, Unit , Min Value , Max Value , Decimals , Thresholds (High Threshold, Low Threshold) , Show Value , Show Tick Marks , Show Labels .
OpLinearGauge	Title, tag, unit, min, max, Orientation (Horizontal/Vertical), thresholds High-High , High , Low , Low-Low , Show Thresholds .
OpDigitalDisplay	Title, tag, unit, Decimal Places , Display Digits , Blink on Alarm .
OpStatusIndicator	Label , Tag (Boolean) , Shape (Circle/Square/Rectangle), Size (Small/Medium/Large), Color On , Color Off , Label Position , Blink When On , Show Label .
OpBulletGraph	Title, Subtitle , tag, unit, Min , Max , Target , orientation, Show Value .
OpSparkline	Title, tag, unit, Data Points , High Threshold , Low Threshold , Show Current Value , Show Min/Max , Fill Area Under Line .
OpTankLevel	Title, tag, unit, min, max, Tank Style (Vertical/Horizontal/Cylinder), Liquid Color , thresholds, Show Value , Show Scale , Show Wave Animation .
OpMultiState	Title, tag, size, Show State Label , States Configuration ("Define states with value, label, color, and icon").
Recipe Table	Select Recipe ("Shows recipe values vs live PLC values side by side."), Table Title , Show Description Column , Highlight Match/Mismatch , display settings.
Array Table	Array Tag * , Orientation , dimension overrides, Row Window / Column Window (slices), Show header , Show row labels , Decimals (numeric cells) , Cell Alignment , Font Size (px) , Max Cells (safety cap) , colours.
State Text	Linked Tag (Analog/Int) , State Texts (Edit States opens the State Text Editor : per state Text to Display , Bg Color , Text Color , Blink , Blink Speed ; state 1 = tag value 1, ...), Show placeholder for unknown values / Unknown Value Text , font and colour settings, Padding (px) , Border Radius (px) .
Query Table	Named Query , Parameter Bindings (tag name or <code>=literal</code>), Refresh Interval (seconds) — 0 = never auto-refresh , Max Rows , Show Title , Title , Show Column Headers , colours.
Server Topology, Process Viewer, Process Status Badge, IP Camera	See Chapter 15.

14.10 Worked example: an operator station for Line 1

Goal: a control-room PC 192.168.10.21 that shows the Line 1 screens large, with navigation on the left, alarms on the right and trends below.

1. Create screens `Line1_Overview`, `Line1_TankFarm`, `Line1_Pumps` (Chapter 10).
2. In **Config** > **Display** > **Navigation Buttons**, tick **Alarms/Events**, **Trends** and **Historian**; create a dropdown group `Line 1` with **Header** and **Navigation** ticked and add the three screens.
3. Start OptiZeus on 192.168.10.21. On the engineering PC open **Config** > **Admin** > **Clients & Shutdown**, rename the station `Line 1 Control`, click **Edit Layout**.
4. Add panels: **Screen Navigation** at 0,0,15,100; **Synoptic View** 15,0,60,70; **Alarm Panel** 75,0,25,70; **Trend Chart** 15,70,85,30.

5. In the Screen Navigation panel tick the three Line 1 screens in the desired order and set **Active Btn** to `#16a34a`.
6. In  **Header**, set **Title Text** `Line 1`, enable **Alarm Counter**, and add a header instrument for `Tank_101.Level1` with style **Bar**.
7. **Save Layout**. On the station, log in and check. If the station should run locked, give its operators a role with **Kiosk Mode** (Chapter 24).

15. Digital twin, live video and advanced views

This chapter describes the advanced views of OptiZeus: the Digital Twin equipment map, live video from IP cameras, Screen Replay of historian data on synoptic screens, screen recording, the Process Viewer and Server Topology widgets, and pop-out screen windows. It is for engineers who configure these views and for supervisors who use Screen Replay to investigate events. Each section states the current limitations of version 2.6.1.26 plainly; read them before planning a project around a feature.

Note: OptiZeus has no 3D view. The "Digital Twin" is a two-dimensional equipment map with a slight depth effect, and the "3D Tank" symbol in the object library is a shaded 2D drawing. None of the views in this chapter requires a licence add-on.

15.1 Digital Twin

15.1.1 What it is

The Digital Twin is a grid (by default 20 × 12 cells of 60 px) on which you place equipment icons. Each item is linked to one **status tag**; the icon is coloured by the state of that tag and refreshed every 3 seconds. It gives a compact plant-wide status map for a supervisor's screen.

It does **not** contain a 3D model, simulation, historical playback or anomaly highlighting. For historical playback on graphics use Screen Replay (Section 15.3).

15.1.2 Opening it

Config > Admin > Digital Twin. Any logged-in user can view the map through the API; only administrators can change it.

15.1.3 Toolbar

Element	Function
Edit Layout / Exit Edit Mode	Switches edit mode on and off.
+ Add Equipment	(Edit mode) Opens the add form.
Refresh Data	Reloads the live values immediately.
Status line	" <i>n equipment items</i> <i>Auto-refresh: 3s</i> <i>n live tags</i> ".

15.1.4 Adding equipment

To add a pump to the map:

1. Click **Edit Layout**, then **+ Add Equipment**.
2. Fill in the form:

Field	Meaning
Name	Equipment name (Equipment name). Only the first 7 characters are shown on the grid.
Type	Motor, Pump, Valve, Tank, Sensor, Conveyor, PLC, HMI (selects the icon).

Field	Meaning
Status Tag	The tag that gives the status (-- Select Tag --).
Grid X, Grid Y	Cell position, starting at 1.
Threshold Low, Threshold High	Optional limits for numeric tags.

3. Click **Add**. The layout is saved to the server.

In edit mode, drag an item to another cell (it snaps to the grid and is saved when you release it). Click an item to select it; the panel "Selected: <name> (<type>)" offers **Remove** ("Remove "<name>"?"). An item cannot be edited: to change its tag or thresholds, remove it and add it again.

15.1.5 Status colours

Status	Colour	Condition
offline	grey	No status tag, the tag is not found, or its quality is not good.
alarm	red	Non-numeric value other than <code>true / 1 / on</code> ; or numeric value above Threshold High or below Threshold Low .
warning	amber	Numeric value above 90 % of Threshold High or below 110 % of Threshold Low .
good	green	All other cases, including <code>true / 1 / on</code> .

Important: A boolean status tag that is **false** is shown as **alarm** (red). Use a tag that is true when the equipment is healthy (for example `Pump_P201.Healthy` , or a calculated tag `NOT Fault`), not a fault bit. The warning band assumes positive thresholds; with **Threshold Low** = 0 there is no low warning.

15.1.6 Viewing

Outside edit mode, click an item to see its details: status badge, **Status Tag** (*Not bound* if empty), **Current Value**, **Position**, **Low Threshold / High Threshold**, and **Close**. The legend at the bottom explains the status colours and types.

15.1.7 Limitations

- Grid size, cell size and background colour cannot be changed in the user interface.
- Grid position 0 cannot be entered (it is stored as 1).
- If a non-administrator moves or adds items, the change appears on screen but is refused by the server and disappears on reload.
- Every 3 seconds the view loads all tags of good quality; on very large projects keep the view closed when not needed.

15.2 Live video (IP cameras)

15.2.1 Supported sources

Source	Supported
MJPEG stream over HTTP/HTTPS (e.g. <code>http://192.168.1.50/mjpg/video.mjpg</code>)	Yes
JPEG snapshot URL	Yes, as a still image (it is not refreshed automatically)
Camera or video-management web page	Yes, embedded as a web page (iframe)
RTSP, HLS, WebRTC, H.264/H.265 streams	No — no transcoding is included. Use the camera's MJPEG output or a web viewer page.

The browser loads the camera URL directly, so the operator's PC must be able to reach the camera's address, and the camera must allow access without an interactive login (or the credentials must be part of an URL the camera accepts).

15.2.2 Placing a camera on a screen

1. In the Viewer Builder open the **Components** category and add **IP Camera** (480 × 360 px).
2. In the **Props** tab (**IP Camera Settings**) set:

Field	Meaning	Default
Title	Caption (Camera 1).	—
Camera URL	"MJPEG stream or web page URL".	—
Type	MJPEG Stream or Web Page (iframe) .	MJPEG Stream
Camera ID (proxy)	Id of a camera registered in Config > Admin > Cameras , to route the stream through the OptiZeus server.	empty
Show Title	Yes / No .	Yes
Border Color	Frame colour.	—

3. Save the screen.

At runtime the tile shows the stream. If the image cannot be loaded it shows "*Camera offline*"; with no URL it shows "*No camera URL configured*".

Important: In version 2.6.1.26 the server proxy (**Camera ID (proxy)**) does not work from screens: the stream request from the image element carries no login token and is refused, so the tile shows "*Camera offline*". Leave **Camera ID (proxy)** empty and enter a **Camera URL** that the operator PCs can reach directly. If cameras are on a separate network that operator PCs cannot reach, provide a web viewer on a reachable host and use **Web Page (iframe)**.

Note: Pages shown in an iframe must allow embedding. Many camera web interfaces forbid it; the tile then stays empty.

15.2.3 Camera register

Config > **Admin** > **Cameras** (**Camera Management**, "Configure IP cameras and video streams. Supports MJPEG streams and iframe-embedded feeds.") keeps a list of cameras:

- **+ Add Camera** asks in turn for **Camera name**; **Camera stream URL**; **Stream type (mjpeg / iframe)**; and **Location (optional)**; then reports "Camera added! Click Refresh."
- **Refresh** shows the table **Name, URL, Type, Location, Status, Actions (Preview, Edit, Delete)**.

Important: In version 2.6.1.26: the **Status** column always shows *Unknown*; **Preview** opens the proxy address and shows an authentication error instead of the picture; and **Edit** clears the camera's location and marks it disabled (the disabled flag has no effect elsewhere). To change a camera, delete it and add it again. The register is useful mainly as documentation of the camera URLs.

WARNING: Camera images are for information only. Do not use a video tile as the only confirmation of a machine state for safety purposes: streams can freeze without a visible error.

15.3 Screen Replay

15.3.1 What it is

Screen Replay plays back historian data on an existing synoptic screen, so that you can see the screen as it looked during a past period — for example the minutes before a trip. It is not a video: the screen is redrawn from the logged tag values.

15.3.2 Opening it

- Historian view: ► **Screen Replay**.
- A **Screen Replay** header button, if enabled (Chapter 14, Section 14.6).

Back returns to the dashboard.

15.3.3 Loading a replay

Control	Meaning
Screen:	The synoptic screen (-- Select Screen --).
From: / To:	Start and end date and time (default: the last hour).
Load Replay	Loads the screen and the historian data (Loading...).

OptiZeus collects the tags used by the screen (linked tags, animation tags, action tags, super object bindings), queries the historian for the period (interpolated values, up to about 4000 points) and shows "*n pts | n logged | n no data*".

Messages:

Message	Meaning
<i>No tags found on this screen</i>	The screen has no tag references.
<i>End date must be after start date</i>	Correct the time range.

Message	Meaning
<i>No historian data found for this time range. Make sure tags on this screen have historian logging enabled.</i>	None of the tags is logged, or there is no data in the period (Chapter 19).

15.3.4 Playback

Control	Function
► / II	Play / pause. At the end, ► restarts from the beginning.
1x, 2x, 5x, 10x, 50x, 100x	Playback speed (default 10x).
Time slider	Jumps to a point in time (1-second steps); dragging pauses playback.
Time display	The replayed date and time.

Between logged samples the last known value is kept.

Tag Status ("Show tag logging status") opens **Tag Logging Status** with **⚠ NO HISTORIAN DATA (n)** and **✓ LOGGED (n)**, grouped by UDT instance. With it on, objects whose tags have no historian data are outlined in red dashes and labelled **NO LOG**.

15.3.5 Limitations

- The replayed screen is **read-only**. Buttons, set-point fields, toggles, popups and other clickable objects on it do not react, so nothing on a replayed screen writes to the PLC.
- Widgets that load their own data — charts, alarm and event tables, Process Viewer, Server Topology, recipe tables — show **live** data, not the replayed period.
- Only logged tags are replayed; unlogged tags show their live values. Check **Tag Status** before drawing conclusions.
- Long periods are replayed at a coarse resolution (about 4000 points over the whole range). Replay short periods for detailed analysis.

Note — fixed in 2.6.1.24: In 2.6.1.23 the replayed screen was not locked, and its controls acted on the live plant. From 2.6.1.24 the replay opens the screen in replay mode, in which all clicks and writes are blocked (Chapter 16, 16.3.3).

15.4 Screen recording

A screen recording captures the application window (or, in the Viewer Builder, the builder canvas) as a video file.

Where	Control
Header	● Record (Start screen recording), shown when Show Record Button in Header is on (Chapter 14, Section 14.2).
Viewer Builder	● Record above the canvas.

While recording: ● **REC m:ss**, II (Pause recording), ► (Resume recording), ■ **Stop** (Stop and download recording). **Processing...** is shown while the file is prepared.

The recording is a WebM video (VP9 or VP8, about 2.5 Mbit/s, 5 frames per second) without sound. It stops automatically after 5 minutes. The file is downloaded by the browser as `scada-recording-YYYY-MM-DDTHH-MM-SS.webm`; nothing is stored on the server. Camera images and embedded web pages may appear blank in the recording.

Note: A screen recording is not an audit record. For regulated evidence use the audit trail and historian (Chapters 18 and 19).

15.5 Process Viewer and Process Status Badge

The **Process Viewer** component (library category **Components**) shows a multi-stage process — for example a sterilisation cycle — as a list of stages coloured as running, completed or pending, with the parameters of each stage and hold points.

Process Viewer Settings:

Field	Meaning
Title	Caption (Process View).
Link Name (for Hold Alert)	Name by which Process Status Badge widgets find this viewer (e.g. <code>Sterilizer-A</code>).
Current Stage Tag	"Integer tag that holds the current running stage number (1-based)."
Total Stages	From Tag ("Integer tag that holds the total number of stages.") or Fixed Number (1-999).
Hold Point Tag	"When true on the running stage, a popup asks the operator to approve."
Stage Parameters	Edit Parameters (or double-click the object) opens the parameter editor: per-stage parameters, overrides and comments, Excel export/import.
Display	Font Size (px), Font Family, Header Background, Running Stage Color, Completed Stage Color, Pending Stage Color, Text Color.

When the hold point tag becomes true on the running stage, a popup asks the operator to approve; approving writes `false` to the hold point tag with the reason "*Hold point approved by operator*".

WARNING: Approving a hold point releases the process in the PLC. Make sure the operator's role and procedure allow it.

The **Process Status Badge (Process Hold Alert Settings)** is a small indicator that follows a Process Viewer by its link name: **Link Name, Linked Process Viewer, Poll Interval (seconds)**, labels and colours for **RUNNING, HOLD** (alarm), **STOPPED** (idle) and **COMPLETE**, **Text Color, Font Size, Border Radius (px), Blink when alert is active, Hide entirely when idle (no alert)**, and **Click to navigate to Process Viewer screen with Target Synoptic Screen ID**. Place badges on overview screens so that operators see a hold anywhere in the plant.

15.6 Server Topology

The **Server Topology** component shows the OptiZeus servers and deployment targets of the installation with their health, and optionally the PLC devices. **Server Topology Settings: Show PLC Devices, Health Check Interval (seconds)** (10-300, default 30), **Compact Layout**.

The widget also offers sync buttons that push the complete configuration (screens, tags, alarms, PLC connections, recipes, super objects) to a target server ("*Sync to <label> succeeded*"). Server deployment is described in Chapter 27.

WARNING: A sync from the Server Topology widget overwrites the configuration of the target server. Place the widget only on screens used by administrators.

15.7 Pop-out screens and multi-monitor operation

Any synoptic screen can be opened in its own window with the ↗ button (*Open in new window (multi-monitor)*) of the navigation cards or navigation panels (Chapter 14, Section 14.7). The window shows the screen only, without header or navigation, and can be placed on another monitor. This is the recommended way to show several screens at once, for example an overview on a wall display and a detail screen on the desk monitor.

15.8 Choosing the right view

Need	Use
Plant-wide equipment status at a glance	Digital Twin, or an overview synoptic screen with super objects
See what the screen looked like at 03:12 last night	Screen Replay (only logged tags)
Watch a machine area	IP Camera component with a direct MJPEG URL
Show a work instruction or demonstration	Screen recording
Follow and approve a staged process	Process Viewer with Process Status Badges
Several screens on several monitors	Pop-out windows

OPTIZEUS SCADA · OPERATION MANUAL

Part IV — Operation

16. Operating the plant: screens, controls and faceplates

This chapter describes how an operator runs the plant from OptiZeus once the project has been engineered: reading synoptic screens, understanding value quality, operating buttons and switches, writing setpoints with the on-screen keyboard, the confirmation, reason and re-login dialogs, faceplates, what the operator sees when a PLC or the server is lost, and what each role may or may not do. It is written for operators and shift supervisors, and for the engineers who must decide which safeguards to switch on. How the screens are built is described in Chapters 10–14; alarms are described in Chapter 17.

WARNING: Every write described in this chapter goes to live equipment. OptiZeus is a supervisory system. It is not a safety system and must never be the only protection against a hazardous condition. Interlocks, trips and emergency stops belong in the PLC and in hard-wired safety circuits.

16.1 The operator's workspace

After login the operator sees the dashboard layout that the administrator configured for the station (Chapter 4 and Chapter 14). A typical operator station shows:

Area	Content
Header	Navigation buttons (synoptic screens and views such as Alarms/Events , Trends , Historian , SOE , Smart Alarms), connection status, user name, logout
Main area	The selected synoptic screen
Side or bottom panels	Optional panels: Alarms panel, Event Log , Real-Time Monitoring , Equipment Control
Overlays	Alarm popup, faceplates, popup screens, touch keyboard, confirmation dialogs



Synoptic screen at runtime

To change screen, click a screen button in the header, or click a navigation object on the current screen. A navigation button can also carry a badge: if the administrator linked the button to a tag or to an alarm group, the button blinks and shows a count when that tag is non-zero or when the group has unacknowledged alarms. This lets you see at a glance which area needs attention.

16.2 Reading a synoptic screen

16.2.1 Values and units

Numeric displays show the engineering value (after any scaling configured on the tag, Chapter 7) with the number of decimals set on the object. Hover over a value object to see a tooltip with the tag name and, when the quality is not good, a `Quality:` line.

16.2.2 Quality and the XXXX indication

OptiZeus never shows a stale number as if it were live. Values of tags that come from a PLC are replaced by the text `XXXX` when:

Condition	What you see
The client lost its WebSocket connection to the OptiZeus server	Every PLC-sourced value shows <code>XXXX</code> ; a red banner SERVER CONNECTION LOST - All values showing XXXX is shown across the top of the screen
All tags of one PLC connection have bad quality (the PLC is unreachable)	Values of that PLC show <code>XXXX</code> ; a red banner PLC DISCONNECTED: <name> - Affected values showing XXXX is shown
A single tag has bad or unknown quality	That value shows <code>XXXX</code> (some animated objects show <code>---</code> or a configured "bad quality" text instead)

Internal (memory) tags have no PLC and are not replaced by `xxxx` when a PLC is lost. PLC connections that are disabled in the configuration are not listed in the banner.

Important: When you see `xxxx`, the screen is **not** telling you the value is zero. Do not operate equipment whose state you cannot see. Use local indication or contact the field operator.

16.2.3 Colours and animations

Colour, blink, fill level, rotation, visibility and similar effects are driven by tag values (Chapter 12). The meaning of colours is project specific; your site should publish a colour legend. A common convention, used by the OptiZeus symbol library, is:

Colour / effect	Typical meaning
Green	Running / open / normal
Grey	Stopped / closed
Red, blinking	Active alarm, not acknowledged
Red, steady or orange	Active alarm, acknowledged
Yellow / amber	Warning, transition, or manual mode

16.3 Operating objects on a screen

What happens when you click an object is decided by the object's click action, set by the engineer (Chapter 12). The actions an operator meets are:

Click action	What happens
Open faceplate	Opens the tag faceplate (Section 16.6) for the object's tag
Open popup	Opens a popup screen or the object's built-in popup (for example a pump faceplate of a super object)
Close popup	Closes the popup the object sits in
Write tag	Writes a fixed value, or asks you for a value (Section 16.4)
Toggle tag	Writes 0 if the tag is 1/true, otherwise writes 1
Navigate / open screen	Opens another synoptic screen
Download recipe / Upload recipe / Open recipe manager	Recipe operations (Chapter 20)
Open URL	Opens a web page in a new window
Run script / Python / command	Runs an engineer-defined script (Chapter 9)
Shutdown	Closes the desktop (Electron) client after confirmation

16.3.1 Security letters

An object can be protected with a **security letter** (A–Z). Each role has a set of letters (defined in the role editor, Chapter 24). If your role does not own the letter of the object you click, the action is refused with the message:

Access Denied: You need security letter "D" to perform this action.

In the default roles, `admin` owns all letters A–Z, `operator` owns A, B and C, and `viewer` owns none. Faceplate popups of super objects can also require a security level; the refusal message then reads "Access Denied: You need security level "<level>" to open this popup."

16.3.2 Confirmation dialog

If the engineer ticked *require confirmation* on the object, a confirmation dialog shows the configured message (default: **Are you sure?**) before anything is written. Click **OK** to continue or **Cancel** to abandon the action. Nothing is written if you cancel.

Tip: Engineers: use a specific message, for example "Start pump P-201? Check that suction valve V-201 is open." A generic "Are you sure?" is quickly clicked through out of habit.

16.3.3 Replay mode

When a screen is shown in **Screen Replay** (historical playback, Chapter 15, Section 15.3), all clicks and writes are blocked. The screen is display-only; nothing on it can write to the PLC (since version 2.6.1.24).

16.4 Writing a setpoint or value

16.4.1 Fixed-value writes

A button configured with a fixed value (for example **START** writes `1` to `Pump_P201_StartCmd`) writes that value immediately after the security and confirmation checks. It does not open a keyboard, even on a touch-screen station.

16.4.2 Entered-value writes

If the object has no fixed value, or the engineer chose keyboard input, you are asked for the value:

- On a station where touch-screen mode is on (set per station, Chapter 4 and Chapter 14), the **touch keyboard** opens (Section 16.5).
- On other stations the **Enter Value** dialog opens.

The **Enter Value** dialog shows the tag name, the allowed range (`Min: ... Max: ...` with the unit) when the tag has limits, and an input field pre-filled with the current value (formatted with the object's decimals, or two decimals by default).

To change a setpoint with the Enter Value dialog:

1. Click the setpoint object, for example the `TIC_101 SP` value on the reactor screen.
2. Check the tag name and the range shown in the dialog.
3. Type the new value.
4. Press **Enter** or click **OK**. Press **Escape** or click **Cancel** to abandon.
5. If the site requires a reason or a re-login (Section 16.4.4), complete that dialog.
6. Watch the value on the screen. It updates immediately (see the warning in Section 16.4.5).

If you leave the value unchanged or empty, nothing is written and no reason is asked for.

16.4.3 Range checks

Two range checks apply:

Check	Where	Behaviour
Display range (tag Min/Max)	Client, before sending	A value outside the range is refused with " <i>Value Out of Range! Entered value: ... Minimum allowed: ... Maximum allowed: ...</i> ". Nothing is written.
Clamping (tag option, engineering limits)	Server	If clamping is enabled on the tag, a value outside the engineering limits is silently clamped to the nearest limit and that limit is written.

String tags (type `String`, `Text`, `WString`) are written as text and are not range-checked.

16.4.4 Reason for change and login confirmation

Two site-wide options (Config → **Admin** → **Security** → **CFR 21 Part 11 Compliance**, see Chapter 25) add dialogs to every **Write tag** action:

Option	Dialog shown to the operator
Require Reason for Write Tag Actions	Reason for Change Required: choose a reason from the list (default list: Process adjustment, Maintenance, Calibration, Testing, Emergency correction) or choose Custom (enter below) and type one
Require Login Confirmation for Critical Actions	Confirm Your Identity / Login Confirmation Required: enter a user name and password. The write is attributed as <i>confirmed by</i> that user.

The reason and the confirming user are stored with the write in the event log (`tag_write` event, category `control`), for example:

```
Tag TIC_101_SP value changed: 85 → 88 | Reason: Process adjustment | Confirmed by: shiftlead1
```

Note: These dialogs apply to the **Write tag** click action. The **Toggle tag** action and faceplate writes do not show them in version 2.6.1.26.

16.4.5 What the server does with a write

When the write reaches the server:

1. The user's role is checked. Only `admin` and `operator` may write (HTTP 403 *Insufficient privileges* otherwise).
2. Security areas that enforce writes (Chapter 25) are checked; a blocked write returns *Write blocked by security area* and is logged as `area_write_denied`.
3. Clamping and reverse scaling (engineering → raw) are applied.
4. The value is written through the tag's PLC connection (OPC UA, Modbus TCP, S7, EtherNet/IP, BACnet, MQTT publish, ...).
5. If the controller accepted the value, the new value is broadcast to all clients, and the tag is protected from being overwritten by the next poll for about 2 s so the screen does not flicker back to the old value.
6. A `tag_write` event is logged with old value, new value, user, reason and confirming user.

If the write does **not** reach the controller — the PLC connection is down, the PLC rejects the value, or an OPC UA server answers with a bad status — the write fails:

- the screen shows **Write Failed!** with the message *"Not written: the PLC connection "<name>" is not connected or rejected the write."*;
- the displayed value stays at (or returns to) the old value. For OPC UA connections the new value is shown on all screens as soon as it is sent; when the PLC refuses it, the old value is put back at once;
- a `tag_write_failed` event is logged (Chapter 18).

Tags that have no PLC behind them — **Memory** tags and simulated tags — are updated in OptiZeus only; the server answers *"Memory tag updated"* or *"Simulated tag updated (no PLC behind this tag)"* and the event log entry ends with `(local)`.

Note — changed in 2.6.1.24: Up to version 2.6.1.23 a write to a PLC tag that did not reach the PLC was stored locally and shown as if it had been written. This no longer happens. Still, after any critical write, confirm the result on a PLC-fed feedback signal (for example the running feedback of the pump, not the start command you wrote): a successful write means the PLC accepted the value, not that the equipment reacted.

16.5 The touch keyboard

On touch-screen stations the touch keyboard opens for synoptic value entry and, since version 2.5.4.276, for any text or number field in the application.

Element	Function
Title bar	Tag name (or field name). Drag the bar to move the keyboard. ✕ closes the keyboard (on a synoptic screen: cancels, nothing is written).
Display	The value being edited. For numeric tags with limits it also shows <code>Min: ... / Max: ...</code> and turns red with OUT OF RANGE while the typed value is outside the range.
Numeric layout	Digits, <code>.</code> , <code>-</code> — used for all non-string tags
Alphanumeric layout	QWERTY keys, caps lock, <code>.</code> , <code>-</code> , Space — used for string tags and text fields
✕ Back	Deletes the last character
Clear	Empties the display
✓ Done	Accepts the value (on a synoptic screen: writes it; Enter on a physical keyboard does the same)

The first key you press **replaces** the pre-filled value; subsequent keys append. ✕ **Back** and **Clear** edit the existing value instead.

For synoptic writes, only ✓ **Done** (or **Enter**) writes the displayed value. ✕ and a tap outside the keyboard **cancel**: the keyboard closes and nothing is written. The **OUT OF RANGE** indication only warns; the out-of-range value is then refused by the range check of Section 16.4.3. In other fields of the application (text and number fields outside synoptic screens) ✕ and a tap outside still close the keyboard and keep what was typed.

Note — changed in 2.6.1.26: Up to 2.6.1.25 ✕ and a tap outside the keyboard also handed the displayed value to the synoptic write, so a key pressed by mistake followed by a tap outside changed the PLC value.

16.6 Faceplates

16.6.1 Tag faceplate

The tag faceplate opens from an object whose click action is **Open faceplate**, or from the **Real-Time Monitoring** panel. It shows:

Section	Content
Header	Tag name, description, current value with unit and quality
Trend (Last 1 Hour)	Sparkline from the historian with Min (1h) , Avg (1h) , Max (1h) . Shows <i>No historian data in the last hour</i> if the tag is not historized (Chapter 19).
Alarm Limits	Low Limit , High Limit , Range , and a HIGH ALARM / LOW ALARM banner while the value is beyond a limit
Write Value	Input and Write button — shown only to <code>admin</code> and <code>operator</code>
Actions	View in Historian

Type the value and click **Write** (or press Enter). The faceplate sends the value to `POST /api/tags/<name>/write`, the same server route as the synoptic write objects, so the role check, security areas and failure handling of 16.4.5 apply. The faceplate shows *Value written successfully* when the controller accepted the value, or the server's error message (for example *Not written: the PLC connection ... is not connected or rejected the write.*) when it did not. The reason and login-confirmation dialogs of 16.4.4 are not shown.

Note — fixed in 2.6.1.25: Up to 2.6.1.24 the **Write Value** field sent its request to a route that does not exist (`/api/tags/write`), so nothing was written to the PLC.

16.6.2 Super-object faceplates

Equipment symbols built as super objects (Chapter 13), for example a pump, a valve or a PID loop, usually open their own faceplate popup: status, mode, start/stop or open/close buttons, setpoint and output fields. The buttons on these faceplates are ordinary write actions and follow the rules of Section 16.4 (security letter, confirmation, reason, range). What each faceplate shows is defined by the engineer.

16.6.3 Sensor faceplate

Some older screens use the **Sensor Faceplate** (current value, **Low Threshold**, **High Threshold**, location and last-updated time). It is display-only.

16.7 The Equipment Control panel

The **Equipment Control** dashboard panel shows cards with **Start/Stop** (pumps, motors) and **Open/Close/Position** (valves) buttons. When the user may not operate, the panel title shows **READ ONLY** and the buttons are disabled with the tooltip *Insufficient privileges*.

Important: In version 2.6.1.26 this panel operates a built-in demonstration equipment list (`pump1`, `pump2`, `valve1`, `valve2`, `motor1`) held in the server's memory. It is **not** connected to any PLC tag. Do not add it to a production operator layout; build real equipment control with synoptic objects and super objects instead.

16.8 Roles and read-only operation

Role	May view screens, alarms, trends	May acknowledge alarms	May write tags / operate equipment	May change configuration
admin	Yes	Yes	Yes	Yes
operator	Yes	Yes	Yes	Only screens and some operator items (Chapter 24)
viewer	Yes	No	No — every write is refused with <i>Insufficient privileges</i>	No

Write permission is checked by the server on every write, so a viewer cannot bypass it from another client. Custom roles created in the role editor are checked by their role id; the server's write and acknowledge routes accept only the ids `admin` and `operator`.

A station can also be locked into kiosk mode (Chapter 4), which hides configuration and system access; kiosk mode does not change the write permissions of the logged-in user.

16.9 When communication is lost

Event	What the operator sees	What to do
PLC connection lost	Red PLC DISCONNECTED banner; that PLC's values show <code>XXXX</code> ; a communication alarm <i>Lost connection to PLC: <name></i> (severity critical by default) is raised	Treat the area as blind. Writes to it fail with <i>Write Failed!</i> and are not sent later (Section 16.4.5). Inform maintenance.
PLC connection restored	Values return; the alarm message changes to <i>Connection restored to PLC: <name> (awaiting ACK)</i> and stays until acknowledged	Acknowledge the alarm after checking the values are plausible
Client lost connection to the OptiZeus server	Red SERVER CONNECTION LOST banner; all PLC values <code>XXXX</code> ; a local alarm <i>Communication Lost - Server disconnected</i> is added in the client	The client reconnects automatically (it polls the server every 2 s and reconnects when ready). If the banner stays, check the network or call the administrator.
Server restarting	The client waits until the server reports <i>ready</i> and reconnects	Wait; do not operate until values are live

Communication alarms are created automatically when a PLC connection is added, and can be edited under **Alarms → Device Connections** (Chapter 17).

16.10 Safe operating practice

1. **Verify before you write.** Read the tag name in the dialog. Many tags have similar names.
2. **Check feedback, not command.** After starting a pump, check its running feedback and flow, not the start-command lamp.
3. **Never operate while blind.** If the value shows `XXXX` or a disconnect banner is shown, do not write to that equipment.

4. **Use the reason field honestly.** The reason and your user name are permanent records in the event log and, for alarm acknowledgements, in the audit trail.
5. **Log out when you leave.** Writes are attributed to the logged-in user. Configure auto-logout (Chapter 25) on shared stations.
6. **Report screen faults.** A value that never changes, a wrong unit or a button that does nothing is a configuration fault. Report it; do not work around it.

16.11 Worked example: changing a tank level setpoint

Plant: Tank_101 with level transmitter LT_101 (0–5 m), inlet valve controller with setpoint LIC_101_SP (tag limits Min 0.5, Max 4.5, unit m). The site has **Require Reason for Write Tag Actions** enabled. The station is a touch panel.

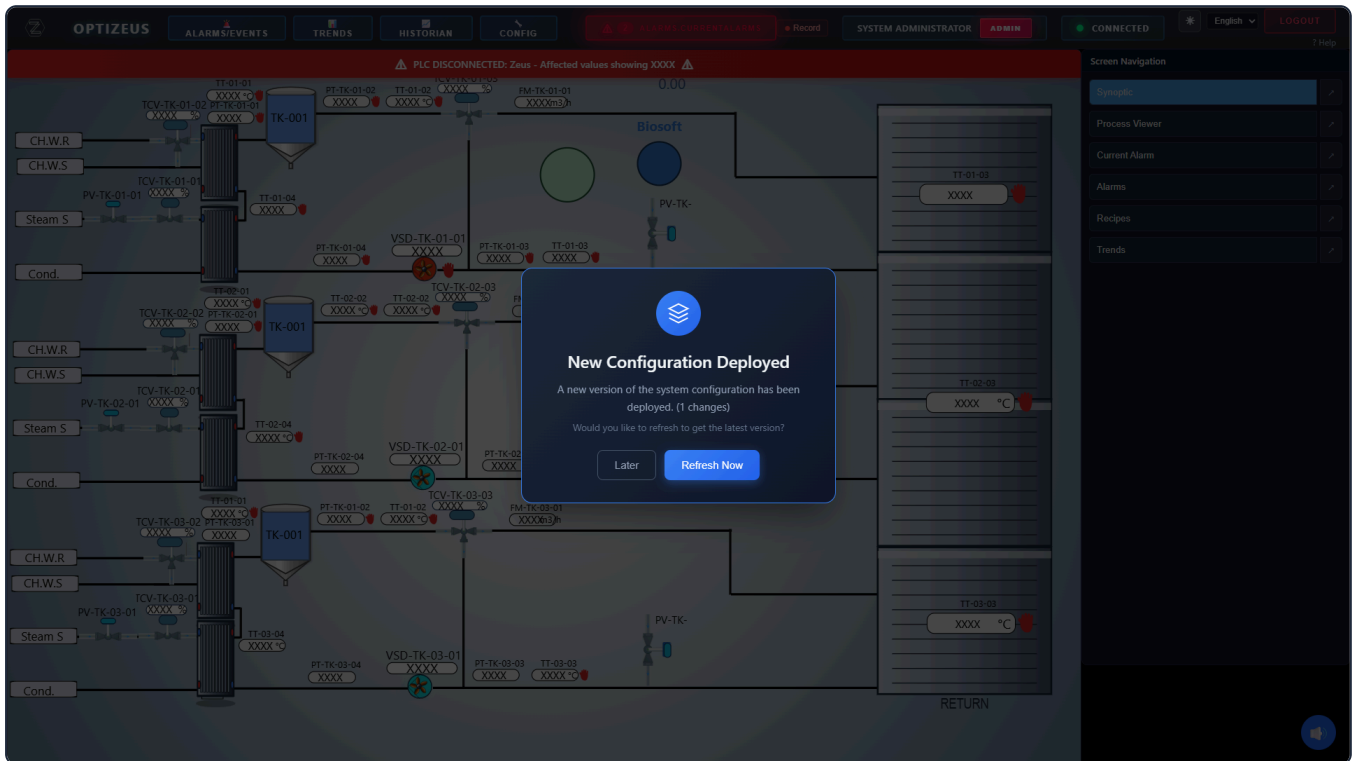
1. On the **Tank Farm** screen, check LT_101 shows a live value (not xxxx) and no PLC banner is shown.
2. Tap the LIC_101_SP value box. The touch keyboard opens showing LIC_101_SP , 2.50 , and Min: 0.5 Max: 4.5 m .
3. Tap 3 , . , 2 . The display shows 3.2 .
4. Tap ✓ **Done**.
5. In **Reason for Change Required**, select **Process adjustment** and confirm.
6. The setpoint shows 3.20 . Watch LT_101 and the valve output over the next minutes to confirm the loop follows.
7. The event log (Chapter 18) now contains:

```
tag_write | control | Tag LIC_101_SP value changed: 2.5 → 3.2 | Reason: Process adjustment
```

If you had typed 5.2 , the keyboard would have shown **OUT OF RANGE**, and after ✓ **Done** the client would have refused the write with *Value Out of Range!*.

17. Alarms

This chapter covers the whole alarm system: how alarms are defined on tags, severities and notification priorities, the alarm states and how an alarm moves between them, deadband and delays, the alarm panel and the alarm popup, acknowledgement (including electronic signatures), shelving, smart alarm grouping and flood detection, notifications by e-mail, SMS, WhatsApp, Telegram and voice, alarm pipelines, alarms bridged from external OPC UA servers, and alarm history. Engineers need Sections 17.1–17.4 and 17.9–17.12; operators and supervisors need Sections 17.5–17.8 and 17.13. The chapter ends with a complete worked example for a tank level.



Alarm configuration

17.1 How alarms work in OptiZeus

OptiZeus evaluates alarms **in the server**, on every tag update cycle, independently of any open screen. When a condition becomes true the server:

1. adds the alarm to the list of current alarms (held in memory, at most 1000 entries — the oldest acknowledged alarms are dropped first when the limit is reached);
2. writes a row to the local `alarms` table (and to an external alarm datalog if one is configured, Chapter 5);
3. logs an `alarm_triggered` event (Chapter 18) and a sequence-of-events entry;
4. broadcasts the new alarm list to every connected client;
5. starts notifications, matching alarm pipelines, alarm scripts and webhooks.

There are two sources of alarm definitions:

Source	Where configured	Notes
Alarm configurations (recommended)	Config → Alarms → Alarm Configurations	Full set of fields: per-limit severity, message, deadband, group, area, notification. Types: analog, Boolean, bit, communication.
Tag thresholds (legacy)	Tag dialog (Chapter 7): High-High / High / Low / Low-Low limits and their enable flags	Fixed severities (HH and LL critical, H and L warning), no deadband, no custom message. Use Sync from Tags to convert them into alarm configurations.

A third source is the **external alarm bridge**: alarms raised by an OPC UA server that implements Alarms & Conditions (Section 17.12).

17.2 The Alarms configuration tab

Open Config → **Alarms**. The tab has six sub-tabs:

Sub-tab	Purpose
Alarm Configurations	Create, edit, copy, delete, import and export alarm definitions
Groups & Areas	Maintain the hierarchical alarm groups and the list of areas
Global Settings	Acknowledgement rules, popup and sound, priority levels, display fields
Notification	Notification channels, users, groups, areas, templates, SMS acknowledge, log (Section 17.10)
Pipelines	Multi-step alarm pipelines (Section 17.11)
Device Connections	Communication-loss alarms of PLC connections, data sources and servers

Only the **admin** role may create, change or delete alarm configurations. When staging is enabled (Chapter 5), changes are held as pending changes, marked **PENDING**, **UPDATE PENDING** or **DELETE PENDING** in the list, and take effect only when deployed.

17.2.1 The alarm configuration list

The list shows **Name**, **Group/Area**, **Type**, **Thresholds**, **Status**, **Notification** and **Actions (Edit, Copy, Delete)**. Use the search box (*Search alarms by name, tag, group, area...*) and the status filter (**All Status**, **Enabled**, **Disabled**); **Clear Filters** resets them. Tick rows and click **Delete N Selected** for bulk deletion.

Toolbar buttons:

Button	Function
+ Create Alarm	Opens the Create Alarm Configuration form
Sync from Tags	Creates alarm configurations from the thresholds already set on tags (Section 17.2.3)
 Clear All Active Alarms	Admin: closes every open alarm (status <i>cleared</i>) and empties the current alarm list; the alarm history is kept
Export	Downloads all alarm configurations as CSV
Import	Imports a .csv or .json file of alarm configurations

WARNING: Clear All Active Alarms closes every open alarm: each open row of the local `alarms` table is set to status `cleared` with the current time as its clear time (`cleared_at`), and the current alarm list is emptied on every screen. The rows themselves are kept, so the alarm **history** remains. The evaluation state is reset, so alarms whose condition is still true fire again on the next cycle. The action writes the event `alarms_cleared_all` (with the number of alarms closed) and an audit record with the administrator's name and client IP (Chapter 18). It cannot be undone. Use it for commissioning, not as a routine way to acknowledge alarms in production.

Note — changed in 2.6.1.25: Up to 2.6.1.24 **Clear All Active Alarms** deleted every row of the `alarms` table — the local alarm history as well as the current alarms — and left no audit record. The API variant `DELETE /api/alarms/clear-all` clears only the in-memory list of current alarms and the evaluation state; it also writes the event and the audit record.

17.2.2 Alarm configuration fields

Basic settings

Field	Meaning	Default / range
Alarm Name *	Unique name, e.g. <code>Tank_101_Level1</code> . Dots are not allowed (use <code>_</code> or <code>-</code>).	Required
Tag *	The monitored tag. For Communication type the field is Tag Name and holds PLC: <code><connection name></code> (filled automatically).	Required except communication
Alarm Type *	Analog (Threshold-based), Boolean (On/Off), Bit (Read specific bit from integer), Communication (PLC Disconnect)	Analog
Description	Free text, shown in the alarm panel and in notifications	—
Alarm Group	Group path from Groups & Areas , e.g. <code>TankFarm/Tank101</code>	—
Area	Area from Groups & Areas , e.g. <code>Tank Farm</code>	—
Enabled	Unticked = never evaluated. Disabling a configuration also removes its current alarms.	On

Analog thresholds (analog type)

Field	Meaning	Default
Deadband (for alarm clearing)	Hysteresis applied when the alarm <i>clears</i> : a high alarm clears when the value falls below <i>limit</i> – <i>deadband</i> , a low alarm clears when the value rises above <i>limit</i> + <i>deadband</i>	0
High-High / High / Low / Low-Low — enable tick	Enables that limit	Off
Threshold	Limit value in engineering units. High limits trigger at <i>value</i> ≥ <i>limit</i> , low limits at <i>value</i> ≤ <i>limit</i> .	—
Severity	Critical, Warning, Info	HH, LL: Critical; H, L: Warning

Field	Meaning	Default
Delay (sec)	On-delay: the limit condition must hold continuously for this time before the alarm is raised (see note)	0 = raise at once
Message	Alarm text; if empty, e.g. Tank_101_Level High-High alarm: 4.62 >= 4.6	—

While High-High is active, the High alarm is not raised again (it is raised only for values between H and HH); the same applies to Low / Low-Low.

How the delays work. The alarm engine starts a timer for a limit on the first scan in which its condition is true (for example $value \geq High\ threshold$). The alarm is raised only when the condition is still true after **Delay (sec)**. If the condition becomes false before that, the timer is reset and starts again from zero the next time the condition is true. **0** raises the alarm on the first scan. For analog limits the delay applies only to raising the alarm: an analog alarm clears as soon as the value is back inside $limit \mp deadband$. Boolean and Bit alarms have both an on-delay and an off-delay (see below). Use the deadband against chattering around the limit and the delay against short spikes; timing that is safety-relevant still belongs in the PLC.

Note — changed in 2.6.1.25: Up to 2.6.1.24 the **Delay (sec)**, **Delay On (sec)** and **Delay Off (sec)** fields were stored but not applied, so every alarm triggered on the first sample that crossed its limit. After upgrading, check existing alarm configurations: a delay that was entered but had no effect now delays the alarm.

Boolean alarm settings (Boolean type)

Field	Meaning	Default
Bit Number (optional)	0–31: evaluate only this bit of an integer tag; empty = whole value as Boolean	none
Trigger Alarm When	Value is ON (True/1) or Value is OFF (False/0)	ON
Severity (On Trigger)	Critical / Warning / Info	Warning
Alarm Message (On)	e.g. <i>Motor fault detected</i>	—
Clear Message (Off)	e.g. <i>Motor fault cleared</i>	—
Delay On (sec) / Delay Off (sec)	Delay On: the trigger state must hold this long before the alarm is raised. Delay Off: the value must stay in the non-trigger state this long before the alarm clears. Each timer restarts if the state changes back before it elapses.	0 = at once
Related Analog Tag (for Trend View)	Analog tag shown in the historian when the alarm is opened there	none

Bit alarm settings (Bit type) — **Bit Position (0-31)** (bit 0 = least significant), **Trigger Alarm When Bit Is 1 (Set) or 0 (Clear)**, **Severity**, **Alarm Message**, **Related Analog Tag (for Trend View)**. Bit alarms use the Boolean **Delay On / Delay Off** values of the configuration (stored in `bool_delay_on` / `bool_delay_off`, for example through the CSV import in 17.2.4).

Communication alarm settings — Alarm Message (*Message shown when PLC disconnects*; default `Lost connection to PLC: <name>`) and **Severity** (default Critical). These configurations are created automatically when a PLC connection is added; use **Device Connections** → **Backfill All Connection Alarms** to create them for older connections.

Notification Settings

Field	Meaning	Default / range
Enable Notifications	Send this alarm to the notification system (Section 17.10)	Off
Notification Priority	1–1000, <i>lower = higher priority</i> . Groups filter on this number.	500

Click **Create Alarm** / **Update Alarm** to save.

The **Enable Notifications** and **Notification Priority** fields of the alarm configuration decide whether its alarms are notified, for every alarm type — Analog, Boolean, Bit and Communication. Only legacy alarms that do not come from an alarm configuration fall back to the tag's own notification flag and priority (`notification_enabled` / `notification_priority` of the tag, default off / 500).

Note — changed in 2.6.1.25: Up to 2.6.1.24 **Analog** alarm configurations ignored their own **Enable Notifications** switch and used the tag's notification flag, which has no field in the tag dialog and is off by default, so analog alarms were not notified even though the alarm list showed **Notification** on. After upgrading, analog alarm configurations with **Enable Notifications** on start sending notifications; review the notification groups (17.10) before the upgrade if that would be unwanted.

17.2.3 Sync from Tags

Sync from Tags converts tag-level thresholds into alarm configurations. The dialog **Sync Alarms from Tags** asks:

Field	Options
Which tag types to add?	Analog (Analog, Float, Int, Dint), Bool / Bit, Both (All tag types)
Tag name filter (optional)	Case-insensitive substring, or * wildcard for anchored patterns (<code>M70*LL</code> , <code>*LL</code> , <code>M70*</code>)
Which triggers to sync?	HIGH_HIGH, HIGH, LOW, LOW_LOW, BOOL ON, BOOL OFF
Alarm Group / Area	Target group and area, or Auto-match group from folder (group taken from the tag's folder)
Default alarm state after sync?	Enabled or disabled

Click **Sync Now**.

17.2.4 Import and export

Export writes a CSV with the columns `name`, `tag_name`, `description`, `alarm_type`, `alarm_group`, `area`, `high_high_value`, `high_value`, `low_value`, `low_low_value`, `high_high_enabled`, `high_enabled`, `low_enabled`, `low_low_enabled`, `high_high_delay`, `high_delay`, `low_delay`, `low_low_delay`, `bool_trigger_on`, `bool_severity_on`, `bool_severity_off`, `bool_message_on`, `bool_message_off`, `bool_delay_on`, `bool_delay_off`, `high_high_severity`, `high_severity`, `low_severity`, `low_low_severity`, `priority`, `enabled`, `notification_enabled`, `notification_priority`, `bool_bit_number`, `deadband`, `comm_message`, `related_analog_tag`. **Import** accepts this CSV (older column aliases are mapped) or a JSON export.

Note: The analog limit messages (`high_high_message` ... `low_low_message`) and the Bit-type fields (`bit_position` , `bit_trigger_on` , `bit_severity` , `bit_message`) are not in the CSV. Re-enter them after an export/import round trip, or copy alarms with **Copy** instead.

17.2.5 Groups and areas

Groups & Areas maintains:

- **Alarm Groups** — a tree. Add top-level groups with *Add new top-level group...* and **Add Group**; subgroups are addressed as paths (`M70/Reactor`). Selecting a parent group in the configuration list shows the alarms of all its subgroups. Deleting a group deletes its subgroups.
- **Alarm Areas** — a flat list of physical locations (e.g., *Tank Farm*, *Pump House*).

Click **Save Groups & Areas**. Groups and areas are used for filtering, for navigation-button badges (a header button can show the count of unacknowledged alarms of a group), for smart grouping (Section 17.8) and for notification routing (Section 17.10).

17.3 Severity and priority

OptiZeus uses three related values; do not confuse them:

Value	Set where	Values	Used for
Severity	Per limit in the alarm configuration	<code>critical</code> , <code>warning</code> , <code>info</code>	Colour in the alarm panel, flood grouping, e-signature text
Priority Levels	Global Settings → Priority & Escalation	Critical <code>#dc2626</code> , High <code>#ea580c</code> , Medium <code>#ca8a04</code> , Low <code>#2563eb</code>	Display reference only
Notification Priority	Per alarm configuration	1–1000 (lower = more important)	Which notification groups receive the alarm

The notification system converts the numeric priority into a category: **1–250 critical, 251–500 high, 501–750 medium, 751–1000 low**. When an alarm configuration has no explicit priority the default 500 (*high*) is used. Communication alarms are *system alarms*: they are sent to every enabled notification group regardless of priority and area.

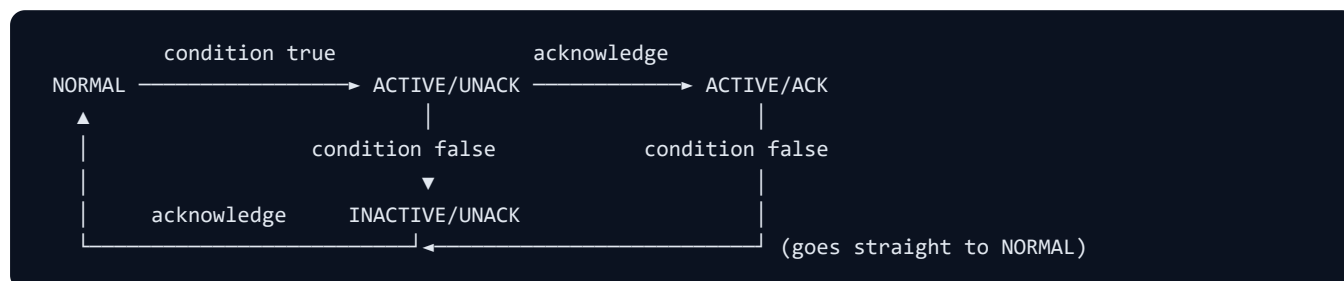
Note: The **Default Priority**, **Enable alarm escalation** and **Escalation delay (minutes)** fields under Global Settings → **Priority & Escalation** are saved but not used by the alarm or notification engine in version 2.6.1.26. Configure escalation with an alarm pipeline (Section 17.11).

17.4 Alarm states

OptiZeus follows the ISA-18.2 state model for acknowledged, latched alarms:

State	Condition	Acknowledged	Shown in the alarm panel as
Normal	False	—	Not shown
Active – Unacknowledged	True	No	Red, blinking, Latest Alarm section with Acknowledge and Shelve buttons

State	Condition	Acknowledged	Shown in the alarm panel as
Active – Acknowledged	True	Yes	Orange, not blinking, section Active - Acknowledged (... <i>waiting for condition to clear</i>)
Inactive – Unacknowledged (return to normal)	False	No	Orange, section Inactive - Unacknowledged (... <i>needs acknowledgment</i>) with Acknowledge
Resolved	False	Yes	Removed from the current list; remains in history



Rules worth knowing:

- An alarm acknowledged while active does **not** fire again until its condition has cleared (for analog alarms: gone past the deadband). Tracking entries of this kind older than 24 h are purged.
- If an alarm returns to normal unacknowledged and then becomes active again, the old inactive row is replaced by a new active one.
- If **Auto-acknowledge all alarms** is enabled (Global Settings), every new alarm is acknowledged at once by **SYSTEM** with the configured reason.
- A PLC communication alarm that recovers before acknowledgement stays visible as *Connection restored to PLC: <name> (awaiting ACK)*.

17.5 The alarm panel and the alarm views

17.5.1 The Alarms panel

The **Alarms** panel is a dashboard panel (and can also be placed on a synoptic screen). The header shows the total number of current alarms and two modes, **Live** and **Historical**.



Current alarms

In **Live** mode the panel shows, from top to bottom: the **Latest Alarm** (newest unacknowledged alarm, with (+N more)), **Ack All (N)** if bulk acknowledgement is allowed, up to five **Active - Acknowledged** alarms, up to five **Inactive - Unacknowledged** alarms, and the **Shelved Alarms** list (▼ Hide / ► Show). With no alarms it shows **No active alarms**.

In **Historical** mode the panel lists alarm history with **ACTIVE/INACTIVE** and **ACKNOWLEDGED/UNACKNOWLEDGED** badges, a timeline (⚠ Activated, ⚡ Acknowledged ... by user, ○ Inactive), the acknowledgement reason, paging (**Previous / Next**) and **Export CSV / Export PDF**.

Which fields are shown is set in Global Settings → **Alarm Display Fields** (Section 17.9).

17.5.2 The alarm popup

When **Enable popup notifications** is on, a popup appears for each new unacknowledged alarm, showing the alarm type, "<TYPE> Threshold Exceeded", value and threshold. **Flash screen on critical alarms** adds a flashing overlay; **Enable alarm sounds** plays a tone (volume from **Sound Volume (%)**). The popup offers **Acknowledge** and **Dismiss**. **Dismiss** only closes the popup; the alarm stays unacknowledged. With **Auto-dismiss notifications**, the popup closes after **Auto-dismiss duration (seconds)** (5–300).

17.5.3 Alarms/Events view

The header button **Alarms/Events** opens a full-page view with two tabs: **Alarm History** and **Event Log** (the latter is described in Chapter 18). The alarm tab offers:

Filter	Options
Status:	All Alarms, Active Only (not acknowledged), Acknowledged Only
Sort:	Newest First, Oldest First
Time Filter:	All Time, Relative Time (Last 1 hour, Last 4 Hours, Last 8 Hours, Last 24 hours, Last 7 days), Date Range

plus **Export CSV**, **Export PDF** and **Refresh**. The table shows 50 rows per page.

Note: The view loads the most recent $50 \times \text{page-number}$ alarms and filters those in the browser. A date range far in the past therefore shows only what falls within the loaded rows. For audits over long periods use a report (Chapter 21) or the external alarm datalog.

17.6 Acknowledging alarms

Only `admin` and `operator` may acknowledge. Since 2.6.1.24 this, and the password check for electronic signatures, apply equally to acknowledgments sent over the live connection (Chapter 25, 25.6.1).

To acknowledge an alarm:

1. Click **Acknowledge** on the alarm (panel, popup, or Smart Alarms view).
2. The **Acknowledge Alarm** dialog shows **Tag**, **Type**, **Value** and **Threshold**.
3. Choose a reason under **Quick Select Reason**, or type one in **Or Enter Custom Reason** (the counter shows *n / m characters minimum*).
4. If electronic signatures are required, enter your password under **Electronic Signature Required (CFR 21 Part 11)**.
5. Click **Acknowledge** (or **Sign & Acknowledge**).

The acknowledgement is recorded with user, time and reason on the alarm row, as an `alarm_acknowledged` event, and as an `acknowledge` entry in the hash-chained audit trail (Chapter 18). With e-signatures on, a signed record "*I acknowledge this alarm and confirm the corrective action*" is stored; a wrong password is refused (*Invalid password - acknowledgment denied*) and logged as `esig_failed`. Acknowledging a bridged OPC UA alarm also acknowledges it on the source server.

Acknowledgment Settings (Global Settings):

Field	Meaning	Default
Require acknowledgment reason	Used by the popup: when on, Acknowledge in the popup opens the reason dialog; when off, the popup acknowledges with the reason <i>Acknowledged from popup</i>	On
Allow custom reason text	Shows the free-text field	On
Minimum reason length	0–100 characters	10
Allow Acknowledge All button	Shows Ack All (N) in the alarm panel	Off
Auto-acknowledge all alarms / Auto-acknowledge reason	Immediate system acknowledgement of every new alarm	Off / <i>Auto-acknowledged by system</i>
Predefined Reasons	Quick-select list	Sensor malfunction, Process adjustment, Operator override, Scheduled maintenance, False alarm - sensor calibration needed

Note: In the Acknowledge dialog a reason is always required and must reach the minimum length, whatever **Require acknowledgment reason** says. If neither custom reasons nor predefined reasons are allowed, the dialog shows *No acknowledgment options available. Please contact an administrator.*

Important: **Ack All** acknowledges the listed alarms one by one with the same reason. Use it only after reviewing each alarm; ISA-18.2 discourages blind bulk acknowledgement.

17.6.1 Acknowledging by SMS and by voice call

If a GSM modem is configured, an operator can reply to an alarm SMS:

SMS text	Effect
ACK	Acknowledges the only unacknowledged alarm; if there are several, the reply lists how many and the oldest id
ACK <id>	Acknowledges that alarm
ACK <id> <reason>	Acknowledges with that reason

The sender's number must be listed under **Notification** → **Inbound (SMS-Ack + Voice)** → **SMS Acknowledge — Authorised Numbers** (phone in E.164 form, user name, role, *can acknowledge*). The acknowledgement is attributed to that user, and the audit trail records the phone number and modem. Unknown numbers are ignored and logged as `sms_in_unauthorised`. A Twilio voice callout can acknowledge with key **1** in the same way.

Important: In version 2.6.1.26 the SMS and voice acknowledge handlers look up alarms in a server variable (`currentAlarms`) that is not defined, so an inbound `ACK` is not processed. Test SMS acknowledgement on your installation before relying on it, and keep on-screen acknowledgement as the primary method.

17.7 Shelving

Shelving suppresses **notifications** for an alarm for a limited time.

To shelve an alarm:

1. In the alarm panel click  **Shelve** on the alarm.
2. Choose a **Shelve Duration: 1 Hour, 4 Hours, 8 Hours, 24 Hours** or **Custom** (minutes, max 10080).
3. Enter **Reason for Shelving *** and click  **Shelve Alarm**.

Shelved alarms are listed under **Shelved Alarms** with **Unshelve**. Shelving and unshelving are logged (`alarm_shelved` , `alarm_unshelved`).

Limitations in version 2.6.1.26:

- Shelving works only when alarm suppression is enabled in `notification-settings.json` (`suppression.enabled: true`); it is **off by default** and the settings page for it is not reachable from the menus. With suppression off the panel reports *Alarm suppression is disabled*.
- The duration is capped by `suppression.maxShelveTime` (default 3600 s), so 4/8/24-hour choices are cut to one hour unless the cap is raised.

- A shelved alarm still appears in the current alarm list and must still be acknowledged; only notifications are suppressed.
- The shelved list is kept in memory and is lost when the server restarts.

17.8 Smart Alarm Grouping and flood detection

The **Smart Alarms** view (header button  **Smart Alarms**, title **Smart Alarm Grouping — ISA-18.2**) groups the current alarms by area (alarms without an area are grouped as *Ungrouped*) and refreshes every 3 s. It shows **Total Active, Unacknowledged, Groups** and **Alarms/min Rate** tiles, one expandable card per group (sorted by highest severity) with a table of **Tag, Type, Value, Severity, Time** and **Status**, and a group **ACK** button that opens a reason dialog (**Reason (optional); Acknowledge All**).

Flood detection is fixed in the server: when 10 or more alarms are raised within 60 s, an alarm flood starts (event `alarm_flood_start`), the view shows **ALARM FLOOD DETECTED**, and alarms during the flood are counted per group. The flood ends (event `alarm_flood_end`) when a new alarm arrives and fewer than 5 alarms remain in the last 60 s. Flood detection only reports; it does not suppress alarms.

Important: Group acknowledgement from this view marks the alarms acknowledged in the live list and writes one `alarm_group_ack` audit entry, but it does not update the individual alarm history rows, does not request an electronic signature, and does not cancel their notifications or pipelines. Where acknowledgements must be individually traceable (GMP), acknowledge from the alarm panel.

17.9 Global display settings

Global Settings → **Alarm Display Fields** selects the columns of the current-alarm view (**Tag Name, Alarm Type, Value, Threshold, Timestamp, Description, Group, Area**), each with a column width and font size in pixels, and the fields of the historical view. Changes are pushed to all connected clients without a page refresh.

17.10 Notifications

Alarm notifications are configured under Config → **Alarms** → **Notification**. The page has eight sub-tabs: **Overview, Channels, Users, Groups, Areas, Templates, Inbound (SMS-Ack + Voice)** and **Log**. All settings require the `admin` role.

17.10.1 How an alarm is routed

For every alarm that has **Enable Notifications** on:

1. The master switch (**Alarm Notifications** on the Overview) must be on.
2. The alarm's numeric priority (Section 17.3) and area are compared with each enabled **group**: the group accepts the alarm if the priority is inside **Priority Min–Priority Max** (or the category is in **Alarm Priorities**), the area is in **Alarm Areas** (empty = all areas; alarms without an area always pass) and the group **Schedule** is active now.
3. For each enabled user of an accepting group, the group's **Notification Channels (in priority order)** are filtered to channels that are globally enabled, enabled for the user, and for which the user has a contact (phone, e-mail, WhatsApp number, Telegram chat id).
4. In **Fallback (try next on failure)** mode the channels are tried in order until one succeeds; in **Send to all selected** mode all are used.

5. The same happens when the alarm returns to normal (templates `...Clear`).

If no group accepts the alarm, or a user has no usable channel, a **skipped** row explaining the reason per group is written to the **Log**. Check the Log first when "the test works but real alarms don't arrive".

Note: **Repeat Interval (minutes)**, **Escalation** and **Escalation Delay (seconds)** on the Overview are stored but not acted on in version 2.6.1.26. Use a pipeline (Section 17.11) for repeats and escalation. The **Voice** channel is not used by group routing in this version; **Local Speaker TTS** speaks alarms independently of groups.

17.10.2 Channels

Channel	Configuration fields	Notes
GSM Modem (SMS)	Connection Type: USB / Serial (COM Port , Baud Rate), Ethernet / TCP (IP Address / Hostname , TCP Port , raw AT commands), Teltonika RUTOS (Router IP / Hostname , Username , Password , Use HTTPS (port 443) , Accept self-signed certificate), Sierra Wireless ALEOS (Modem IP / Hostname , Telnet AT Port), SMSEagle (Host , Port , API Version , Access Token); Modem Name	A modem that stops answering raises a communication alarm (checked every 60 s). Inbound SMS is used for SMS acknowledgement.
Email (SMTP)	SMTP Host , Port , user, password, From Address , From Name , Secure (TLS)	Test and verify buttons send a test mail
WhatsApp	WhatsApp Type: Personal WhatsApp (<i>Connect via QR code scan from your phone</i>) or WhatsApp Business API (API URL , Access Token , Phone Number ID)	Personal mode runs a headless Chrome session; the QR code appears after 10–30 s. Hard Reset deletes the session.
Telegram Bot	Bot token; users need a Telegram Chat ID	
Local Speaker TTS	Voice, volume, rate, minimum priority	Speaks the alarm on the server PC
Voice Calls (Twilio)	Account SID, auth token, from-number, voice, language	Also used by the voice callout of the Inbound tab

WhatsApp session consent. A personal WhatsApp session is resumed automatically at server start only if WhatsApp is enabled in the configuration, or if a saved session exists **and** WhatsApp notifications are switched on. A session folder copied from another PC, or left behind after WhatsApp was switched off, is ignored. This prevents a foreign or unwanted session from being reactivated and avoids a start-up delay. Treat the `whatsapp-session` folder as a credential: never copy it between sites or include it in a release.

17.10.3 Users, groups and areas

Users (not the same as login users): **Name ***, **Phone**, **Email**, **WhatsApp Number**, **Telegram Chat ID**, **Enabled**, **Channels**.

Groups: **Group Name ***, **Enabled**, **Users**, **Notification Channels (in priority order)**, mode (**Fallback (try next on failure) / Send to all selected**), **Alarm Priorities**, **Alarm Areas (empty = all areas)**, **Priority Min**, **Priority Max**, **Schedule (Always Active or Weekly Schedule** with a start and end time per day; overnight shifts such as 18:00–06:00 are supported).

Areas: the notification area list; alarm areas are matched by id or by name.

17.10.4 Templates

The **Templates** tab overrides the message text per channel and kind (trigger / clear, and e-mail subject and body). Placeholders:

Placeholder	Value
{tagName}	Tag name
{description}	Alarm description; {descSep} is " - " when a description exists
{value}, {unit}, {threshold}	Value, unit, limit
{category}	CRITICAL / HIGH / MEDIUM / LOW
{priority}	Numeric priority
{area}	Area or N/A
{message}	Alarm message
{timestamp}, {timestampISO}	Local time, ISO time
{id}, {idHint}	Alarm id; {idHint} = [id ..., reply ACK ...]

Default SMS trigger template:

```
{category} ALARM: {tagName} = {value}{unit} (threshold: {threshold}{unit}){descSep}{description}{idHint}
```

17.10.5 Testing and the log

Use **Trigger Test Alarm** (Overview) to push a test alarm through the routing, and the per-channel test buttons to test a single channel. The **Log** tab lists every sent, failed and skipped notification with channel, recipient, status and error; it can be cleared by an administrator.

17.11 Alarm pipelines

A pipeline is a named, multi-step flow started automatically when a matching alarm is raised. Open Config → **Alarms** → 🚨 **Pipelines**, click + **New**, fill in the form and click **Save**. ▶ **Test** starts the pipeline with a simulated alarm; **Delete** removes it.

Section	Field	Meaning
General	Name * , Description , Enabled	
🎯 When to trigger	Min Severity (Any, Low, Medium, High, Critical), Alarm Type , Tag Prefix	Empty fields match all alarms
🛑 Stop pipeline if alarm is...	Acknowledged , Cleared (returned to normal)	Running executions are aborted on these events (both on by default)
▶▶ Pipeline Steps	Ordered steps, moved with ↑ ↓, removed with ×	

Step	Fields	Behaviour
Notify	Channels (email, sms, whatsapp, telegram, voice, webhook), Recipients (comma separated), Message Template ({{alarm.message}} , {{alarm.tag}} , {{alarm.severity}} , {{alarm.value}})	Sends a message
Delay	Wait for N minutes or seconds	Delays the next step
Condition	Check: Still unacknowledged, Has been acknowledged, Still active, Has been cleared; If TRUE go to step / If FALSE go to step (1-based), or or abort pipeline	Branching
Set Tag	Tag Name, Value	Writes the tag, e.g. a horn output (same write path as an operator write, see below)
End	—	<i>Pipeline will terminate here.</i>

Executions are stored in the database, checked every 5 s, and survive a server restart. Each start is logged as `pipeline_started`.

Important: Two limitations apply in version 2.6.1.26. (1) The **Notify** step calls a notification function that does not exist in the server, so Notify steps send nothing — test with ► **Test** and the notification **Log** before relying on a pipeline. **Set Tag**, **Delay** and **Condition** steps work. (2) **Min Severity** compares against the scale low/medium/high/critical, while alarm configurations use critical/warning/info; `warning` and `info` alarms therefore never pass a Min Severity filter. Leave **Min Severity** at **Any** and filter with **Tag Prefix** or **Alarm Type** instead.

The **Set Tag** step writes through the same server code as an operator write (`POST /api/tags/<name>/write` , Chapter 16): every protocol that an operator can write (including EtherNet/IP Logix5000 and MicroLogix/PCCC), Memory and Simulation tags, scaling and clamping. Security areas are not applied. The write is recorded as a `tag_write` event with user `system` ; a write that fails is logged as `tag_write_failed` , and the pipeline continues with the next step.

Note — changed in 2.6.1.26: Up to 2.6.1.25 the **Set Tag** step used a separate, simplified write routine that did not write Memory tags or EtherNet/IP and PCCC connections, and failures were not logged.

WARNING: A **Set Tag** step writes to the PLC without operator involvement. Use it only for annunciation outputs (horns, beacons) that are safe to switch at any time.

17.12 Alarms from external systems and Zeus PLC controllers

An OPC UA server that implements **Alarms & Conditions** can forward its alarms into OptiZeus. Edit the OPC UA PLC connection (Chapter 6) and, under **External Alarm Bridge (A&C)**, tick **Subscribe to remote alarms (OPC UA Alarms & Conditions)**, then set **Severity Min / Severity Max** (OPC UA severity 1–1000), **Default Area** and **Default Group**.

Bridged alarms appear in the normal alarm list with source `opcua-ac` , run through notifications and pipelines, and are acknowledged back on the source server when acknowledged in OptiZeus (two-way acknowledgement). Their

OPC UA severity is mapped to a priority label: ≥ 800 Critical, ≥ 600 High, ≥ 400 Medium, otherwise Low. Alarm definitions stay on the source system; OptiZeus only receives live events.

For **Zeus PLC** controllers, use the bridge if the controller's OPC UA server publishes Alarms & Conditions events. Otherwise map each alarm bit of the controller to a Boolean or Bit tag and define Boolean/Bit alarm configurations (Section 17.2.2), which works with any controller.

17.13 Alarm history

Every alarm occurrence is a row in the local `alarms` table: time, tag, type, value, limit, acknowledged flag, acknowledged by/at, acknowledgement reason, cleared time and status. Acknowledgement rows carry an HMAC checksum. If an alarm datalog is configured on an external database (Chapter 5), alarm history queries read from that database instead.

Alarm history is shown in the **Historical** mode of the alarm panel, in the **Alarms/Events** view, as markers on historian and trend charts, and in reports. Automatic deletion of old local alarm rows is set by the administrator (Config → **Admin** → **Database** → **Auto-Purge Internal Database** → **Alarm History**, in days; 0 = keep forever, which is the default).

17.14 Worked example: Tank_101 level alarms

Tank_101 is 5 m high. The level transmitter tag is `LT_101` (0–5 m). The requirements:

Limit	Value	Severity	Operator response	Notification
High-High	4.6 m	Critical	Stop filling immediately	SMS and e-mail to the on-call group
High	4.2 m	Warning	Check inlet valve	Screen only
Low	0.8 m	Warning	Check outlet	Screen only
Low-Low	0.4 m	Critical	Pump P-201 protection	SMS and e-mail

Step 1 – Area and group. In **Groups & Areas** add the area `Tank Farm` and the group `TankFarm/Tank101`. Click **Save Groups & Areas**.

Step 2 – Alarm configuration. Click + **Create Alarm** and enter:

Field	Value
Alarm Name	<code>Tank_101_Level</code>
Tag	<code>LT_101</code>
Alarm Type	Analog (Threshold-based)
Description	<code>Tank 101 level</code>
Alarm Group / Area	<code>TankFarm/Tank101 / Tank Farm</code>
Deadband	<code>0.05</code>
High-High	enabled, Threshold <code>4.6</code> , Severity Critical, Delay (sec) <code>2</code> , Message <code>Tank 101 overfill – stop filling</code>

Field	Value
Low-Low	enabled, 0.4 , Critical, delay 2 , Tank 101 empty – P-201 at risk
Enable Notifications	ticked, Notification Priority 100

Click **Create Alarm**. Then create a second configuration `Tank_101_Level_Warn` on the same tag, group and area, **Deadband** 0.05 , with **High** enabled (4.2 , Warning, delay 5 , Tank 101 level high) and **Low** enabled (0.8 , Warning, delay 5 , Tank 101 level low), and **Enable Notifications** left unticked. Deploy if staging is on.

Note: The **Enable Notifications** switch and the priority apply to all limits of one configuration. Splitting the limits over two configurations, as here, notifies only on HH and LL while H and L stay screen-only. With the delays, a level that touches a limit for less than 2 s (HH/LL) or 5 s (H/L) raises no alarm.

Step 3 – Notification group. Under **Notification** → **Users** add the on-call engineers with phone and e-mail. Under **Groups** create `On-call Tank Farm`: users selected, channels **sms** then **email**, mode **Fallback**, **Priority Min** 1, **Priority Max** 250, **Alarm Areas** `Tank Farm` , schedule **Always Active**. Enable SMS and e-mail on the Overview.

Step 4 – Test. With the process safe, force `LT_101` in the simulator or PLC:

Value	Expected result
4.30 (held longer than 5 s)	High alarm, warning, no SMS
4.65 (held longer than 2 s)	High-High alarm, critical, SMS to the on-call engineer (check the Log)
4.58	High-High still active ($4.58 \geq 4.6 - 0.05$)
4.50	High-High clears (inactive, awaiting acknowledgement); return-to-normal SMS sent

Step 5 – Operate. The operator acknowledges each alarm with a reason. The audit trail (Chapter 18) shows who acknowledged what and when; the historian (Chapter 19) with **Alarms** overlay shows the alarm markers on the `LT_101` trend.

18. Events, sequence of events and the audit trail

OptiZeus keeps three separate records of what happens in the plant and in the system: the **event log** (a broad operational log: alarms, writes, logins, configuration changes, system messages), the **sequence of events** (SOE, a time-ordered record of digital state changes and alarms for post-incident analysis) and the **audit trail** (a hash-chained record of who changed what, intended for 21 CFR Part 11 and GAMP 5 environments). This chapter explains what each record contains, where it is stored, how to view, filter and export it, and how the audit trail's integrity is verified. Operators use the event views; supervisors and quality staff use the SOE and audit trail; administrators configure retention.

18.1 The three records compared

	Event log	Sequence of events	Audit trail
Purpose	Operational history	Incident reconstruction	Regulatory record of changes
Storage (local)	<code>event_log</code> table	<code>sequence_of_events</code> table	<code>audit_trail</code> table
Written by	Almost every server function	Tag broadcast cycle, alarm engine, API	Security-relevant actions (Section 18.4.1)
Integrity protection	None	None	HMAC-SHA256 hash chain with sequence numbers
External copy	Yes, if an event datalog is configured	No	Yes, if an event datalog is configured
Viewed in	Alarms/Events → Event Log , Event Log panel, Config → Admin → System Logs	SOE view	Audit Trail view
Who may view	All users	All users	<code>admin</code> only

18.2 The event log

18.2.1 What is recorded

Every event has a time, an **event type**, a **category**, a **severity** (`info` , `warning` , `critical` , some `debug`), a **source** (user name, `system` , `operator` , `alarm_config` , ...), a description and optional JSON details. Typical events:

Category	Examples of event types
<code>alarm</code>	<code>alarm_triggered</code> , <code>alarm_cleared</code> , <code>alarm_acknowledged</code> , <code>alarm_shelved</code> , <code>alarm_flood_start</code> , <code>pipeline_started</code> , <code>alarm_bridge_started</code>

Category	Examples of event types
control	tag_write (old → new value, reason, confirming user; (local) for a Memory or simulated tag, which has no PLC behind it), tag_write_failed (a write to a PLC tag that did not reach the PLC; severity warning), recipe_downloaded, equipment_control. Since 2.6.1.26 writes made by the server itself are logged the same way: source scheduler with reason <i>Weekly schedule <title></i> , system for automation actions and pipeline Set Tag steps, and the downloading user with reason <i>Recipe download <name></i>
config	alarm_config_created/updated/deleted, alarm_settings_updated, alarms_cleared_all (warning; Chapter 17), sensor_added, historian_tier_config_updated, translation and language changes
security	unauthorized_action, area_write_denied, default_password_login, sms_in_unauthorised, recipe_download_denied (since 2.6.1.26), audit_export
system	client connect/disconnect, service start, backups
maintenance	auto_purge

18.2.2 Live event display

The **Event Log** panel (dashboard or synoptic object) shows the most recent events with category, severity and user. Its **Display Settings** (gear) choose **VISIBLE COLUMNS** (**Time**, **Category**, **Severity**, **User**, **Description**, **Details**), **FONT SETTINGS** (**Size (px)**, **Font Family**), **ICON** and **ROW HEIGHT**. A time filter offers **All Time**, **Relative** (Last 5 min ... Last 30 days) and **Custom Range**.

18.2.3 Alarms/Events → Event Log

Click **Alarms/Events** in the header and choose the **Event Log** tab.

Filter	Options
Category:	All Categories, System, User, Alarm, Configuration, Communication
Severity:	All Severities, Critical, Warning, Info
Sort:	Newest First, Oldest First
Time Filter:	All Time, Relative Time, Date Range

Columns: **Time**, **Category**, **Severity**, **Source**, **Message**, **Details**. **Export CSV** and **Export PDF** (report title *Event Log Report*) export the filtered list; **Refresh** reloads it.

Important: When no external event datalog is configured, this tab shows only the **last 100 events kept in server memory** since the last server restart — not the full local event table. The category filter values *User*, *Configuration* and *Communication* do not match the category names the server writes (`config`, `control`, `security`, ...), so those filters return nothing for most events. To search the complete local event history use Config → **Admin** → **System Logs** → **Event Logs** (Chapter 29), which queries the `event_log` table with hours, category, severity and text filters.

18.2.4 External event datalog

If a datalog with event logging enabled is configured on an external database (SQL Server, MySQL, PostgreSQL; Chapter 5), every event, audit entry and electronic signature is also written there, and the event and audit views read from it. The local SQLite tables are always written as well.

18.3 Sequence of events (SOE)

18.3.1 What is recorded

An SOE row holds: timestamp (ISO, millisecond offset stored separately), tag name, previous value, new value, event type, source, area, equipment, description and quality. Rows are written:

Event type	Written when	Source
change	A tag broadcast finds that a digital value changed (Boolean/Digital tags, or any value changing to or from 0/1/true/false), or that a tag's quality changed	plc
alarm	An alarm is raised	system
any (command , manual , communication , ...)	An external system or script posts an entry to <code>POST /api/soe</code>	as supplied

Internal `_System.*` tags are excluded.

Important: SOE timestamps are taken by the OptiZeus server when the change is detected in its update cycle (normally 1 s), not by the PLC. Two changes inside the same cycle get the same time. OptiZeus SOE is therefore suitable for sequencing at the scan-cycle level, not for first-out analysis of events milliseconds apart. For true first-out resolution, time-stamp the events in the PLC and write them to tags or to `POST /api/soe`.

18.3.2 The SOE view

Click  **SOE** in the header (the button must be enabled for the station, Chapter 14). The view **Sequence of Events** opens with the last hour.

Filter	Meaning
Start / End	Time range (default: last hour to now)
Tag	Substring of the tag name (<i>Filter by tag...</i>)
Area	All Areas or one area recorded in SOE rows
Equipment	All Equipment or one equipment value
Type	All Types, Alarm, Change, Command, Communication, Manual

Click **Query**. The summary shows **Total Events**, **Unique Tags** and counts per type. The table columns **Timestamp (ms)**, **Tag**, **Previous**, **New Value**, **Type**, **Source**, **Area**, **Equipment**, **Description** can be sorted by clicking the header. A query returns at most 1000 rows; narrow the time range for busy plants. **Export CSV** saves the displayed rows.

18.3.3 Reconstructing an incident

To reconstruct the trip of pump P-201:

1. Open **SOE**, set **Start** to five minutes before the trip and **End** to two minutes after.
2. Leave **Type** at **All Types** and **Tag** empty; click **Query**.
3. Sort by **Timestamp (ms)** ascending.

4. Read the sequence, for example: `LT_101` quality change → `P201_Running` 1 → 0 → alarm `LT_101` Low-Low → `P201_Fault` 0 → 1.
5. Open the **Historian** (Chapter 19) for the same window with `LT_101` and `P201_Running` and the **Alarms** overlay to see the analog context.
6. Open **Alarms/Events** → **Alarm History** for the acknowledgement times and reasons.

18.4 The audit trail

18.4.1 What is recorded

Each audit record contains: timestamp, user id and user name, **action**, **entity type**, entity id, **entity name**, **field**, **old value**, **new value**, IP address, details (JSON), record hash, previous hash and sequence number.

Actions recorded in version 2.6.1.26 include:

Area	Actions
Users and login	<code>login</code> , <code>login_oidc</code> , <code>logout</code> , <code>password_changed</code> , <code>password_change_failed</code> , <code>mfa_enabled</code> , <code>mfa_disabled</code> , <code>user create</code>
Alarms	<code>acknowledge</code> (every individual acknowledgement), <code>alarm_group_ack</code> , <code>sms_alarm_ack</code> , <code>voice_alarm_ack</code> , <code>alarms_cleared_all</code> (Clear All Active Alarms , with the number of alarms closed; since 2.6.1.25)
Electronic signatures	<code>esig_applied</code> , <code>esig_failed</code> (alarm acknowledgement, recipe approval, generic signing)
Tags	<code>create</code> , <code>update</code> , <code>delete</code> of tags
Settings	<code>update</code> of settings (historian configuration, system and Electron settings, ...)
Recipes and batch	<code>approve</code> of batch recipes
Procedures	<code>procedure_approved</code>
Automation	<code>create</code> , <code>update</code> , <code>delete</code> of automation actions; edge gateway <code>start</code> , <code>stop</code> , <code>update</code>
Imports and backups	screen and project imports, alarm imports, backup <code>download</code>
Notification	SMS acknowledge ACL add/update/delete, voice callout configuration
Redundancy	<code>failover</code>
PLC programs	PLC program actions
Historian	<code>delete</code> (purges)

Note: Operator tag writes are recorded in the **event log** (`tag_write`, with reason and confirming user), not in the hash-chained audit trail. Many configuration changes (alarm configurations, screens, PLC connections) are also recorded as events rather than audit records. If your validation requires every operator write in the audit trail, record this as a gap in your risk assessment (Chapter 31).

18.4.2 The Audit Trail view

The **Audit Trail** header button is shown to administrators. The view **Audit Trail** has these filters:

Filter	Options
Entity Type:	All Entities or tag, alarm, user, role, screen, datalog, plc, recipe, chart, settings, super_object
Action:	All Actions or create, update, delete, acknowledge, login, logout, start, stop, export, import
User:	Exact user name (<i>Filter by user...</i>)
Time Filter:	All Time, Relative Time (Last 1 hour ... Last 30 days), Date Range (Start / End)
Sort:	Newest First, Oldest First

The table shows **Timestamp**, **User**, **Action** (coloured badge: create green, update blue, delete red, acknowledge orange), **Entity Type**, **Entity Name**, **Field**, **Old Value** and **New Value** (truncated to 50 characters). 50 records per page; the query returns at most 1000 records.

Note: The filter lists are fixed lists; actions such as `esig_failed` or `password_changed` are not offered in **Action:** but are shown with **All Actions**.

The footer shows *Showing n of m records* and the buttons **Refresh**, **Export PDF**, **Export CSV** and **Verify Integrity**.

18.4.3 Exporting the audit trail

- **Export PDF** creates a PDF of the records currently loaded in the view.
- **Export CSV** asks the server for a CSV with the columns `Timestamp`, `User`, `Action`, `Entity Type`, `Entity Name`, `Field`, `Old Value`, `New Value`, `IP Address`, `Details` and logs an `audit_export` event.

Important: In version 2.6.1.26 **Export CSV** reads only from an external event datalog. Without an external datalog the CSV contains just the header line. Use **Export PDF** for local audit trails, or configure an external event datalog.

18.4.4 Integrity: the hash chain

Every new audit record gets a sequence number and an HMAC-SHA256 hash computed over its user, action, entity, field, old and new value, IP address, details and the **previous record's hash** (`GENESIS` for the first). Changing, inserting or deleting any record breaks the chain from that point.

To verify the audit trail:

1. Open **Audit Trail**.
2. Click **Verify Integrity**.
3. Read the result badge:

Result	Meaning
CHAIN VALID (n verified, m legacy)	All chained records verify. <i>Legacy</i> records are older records created before hashing was introduced; they are skipped.

Result	Meaning	
CHAIN BROKEN	A record's link or hash does not match. The server reports the id and sequence of the first bad record.	
`...`	k gaps`	Sequence numbers are missing (records were deleted)

The check always runs on the local `audit_trail` table, even when the view displays records from an external datalog. The HMAC key is the server's secret; keep it with the backup of the server (Chapter 26), because a verification on a restored system with a different secret reports every record as broken.

WARNING: Automatic purging of old audit records (Section 18.5) deletes the start of the chain. After the first purge, **Verify Integrity** reports **CHAIN BROKEN** at the oldest remaining record. In regulated environments keep audit retention at 0 (unlimited) and archive through backups.

18.4.5 Electronic signatures

Electronic signatures (Chapter 25) are stored in their own table with user, time, action, meaning, record type and id, record hash, signature and reason; a signature can be checked through `GET /api/esignature/verify/<recordType>/<recordId>`. Applying and failing a signature are also audit records.

18.5 Retention

Record	Setting	Default
Local alarm history	Config → Admin → Database → Auto-Purge Internal Database → Alarm History (days)	0 = unlimited
Local event log	Same section, Event Log (days)	0 = unlimited
Local audit trail	Same section, Audit Trail (days)	0 = unlimited
SOE	No automatic purge	—
External datalogs	Datalog retention (Chapter 5)	per datalog

The purge runs 45 s after start-up and then every hour. Alarm purges are logged as `auto_purge` events. The database also grows with SOE rows; on plants with many digital signals plan disk space or purge SOE rows during maintenance (Chapter 26).

18.6 Worked example: a monthly audit review

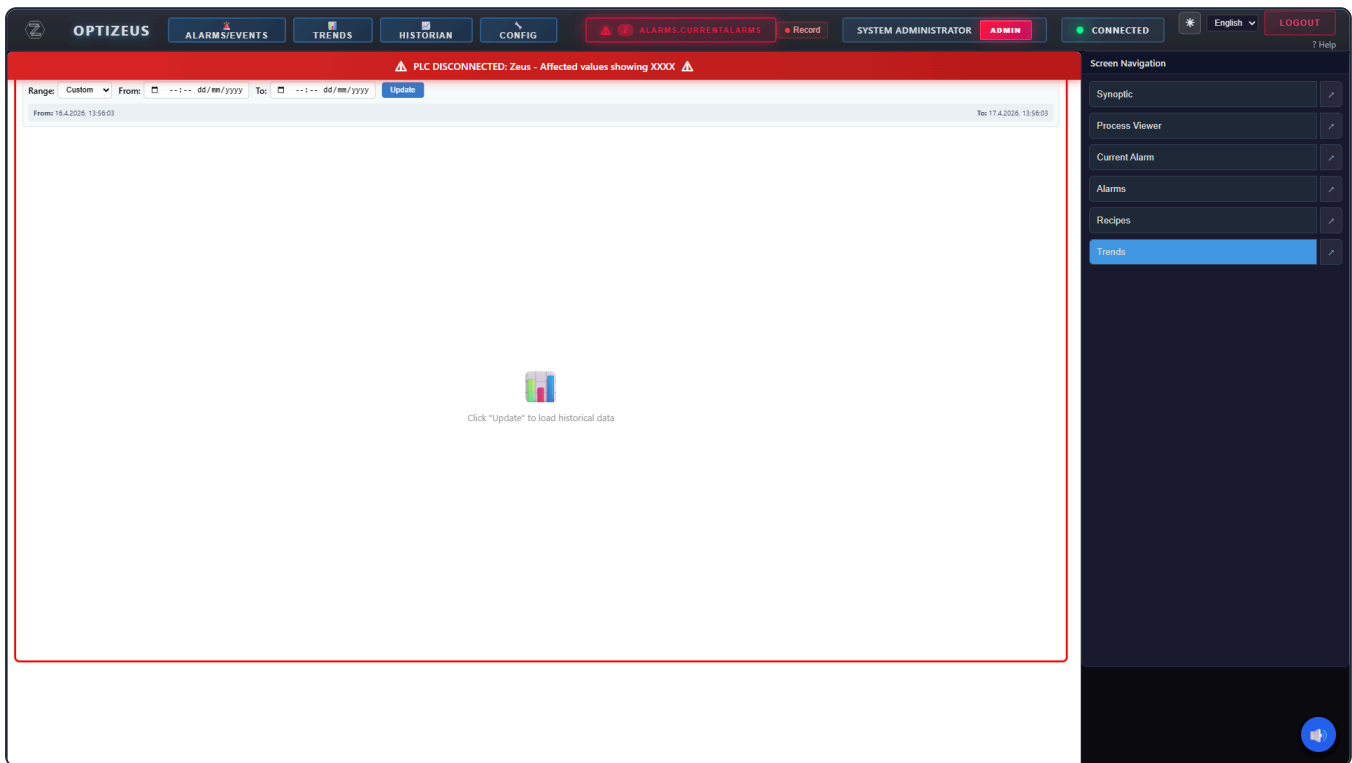
A quality reviewer must confirm, each month, who acknowledged critical alarms and who changed settings.

1. Log in as an administrator and open **Audit Trail**.
2. Click **Verify Integrity** and note the result (for example *CHAIN VALID (4 812 verified, 0 legacy)*) in the review record.
3. Set **Time Filter**: to **Date Range**, **Start** 1st of the month 00:00, **End** last day 23:59.

4. Set **Action:** to `acknowledge` . Review user names and reasons; follow up acknowledgements with generic reasons.
5. Set **Action:** back to **All Actions** and **Entity Type:** to `settings` ; review each change with its old and new value.
6. Click **Export PDF** and attach the PDF to the review record.
7. For operator writes, open Config → **Admin** → **System Logs** → **Event Logs**, filter category `contro1` for the same period and export.

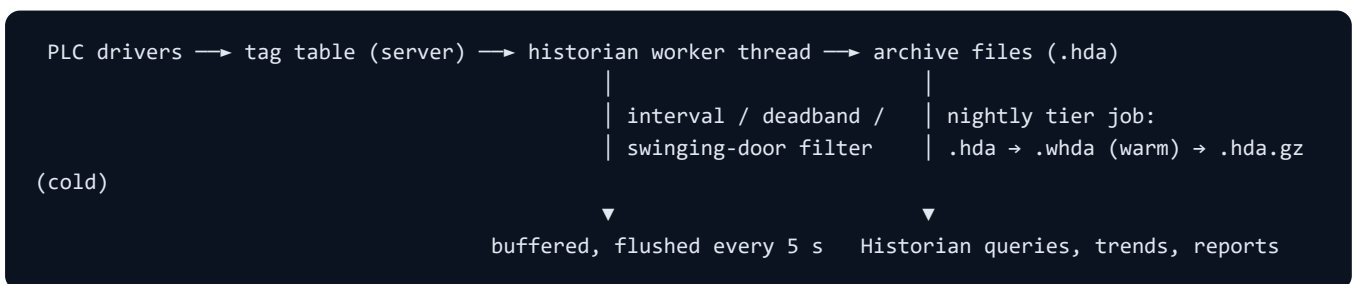
19. Historian and trends

This chapter describes how OptiZeus records process values over time and how you view them: the historian recorder and its storage, the recording methods, deadband and compression, the hot/warm/cold storage tiers and retention, the **Historian** view with its chart types, retrieval modes, period comparison, correlation analysis, annotations, templates and export, the live **Trends** view and the trend charts on screens, and performance guidance. Engineers need Sections 19.1–19.4; operators, process engineers and supervisors need Sections 19.5–19.9.



Historian view

19.1 Architecture



- Recording runs in a dedicated **historian worker thread**, so disk writes never delay PLC polling or screen updates. The server passes the values of all historized tags to the worker on every update cycle.
- Only values with **good** quality are recorded. While a PLC is disconnected nothing is stored for its tags; the trend shows a gap.
- Only **numeric** values are recorded. Values that cannot be read as a number (text, and Boolean values delivered as `true / false` rather than `1 / 0`) are skipped.

- The worker buffers records and flushes them every 5 s (or when 500 records are waiting for one tag). A hard stop of the server can lose up to the last 5 s of history.
- Files are stored per tag and per day: `<archive path>\<tag>\<tag>_YYYYMMDD.hda`, 17 bytes per sample (8-byte time, 8-byte value, 1-byte quality). Dots and characters not allowed in file names are replaced by `_`.
- If the worker cannot start, the server records inline instead (to the archive, to a local SQLite table or to an external database, depending on the settings in Section 19.2).

19.2 Configuring the historian

Open Config → **Data Storage** → **Historian**. The page **Historian Configuration** has:

Field	Meaning	Default
Master Enable - Historian Recording	Master switch. Nothing is recorded while it is off.	Off
External Data Source	External database (SQL Server, MySQL, PostgreSQL, ...) used when the file archive is off, and for the optional file index	none
Table Name Prefix	Table prefix in the external database	historian
Retention (days, 0=unlimited)	Deletion age for raw archive files (Section 19.4.3); 0 = keep for ever	90
Default Interval (ms)	Default recording interval for newly enabled tags	1000
Default Deadband	Default deadband	0
Default Method	Interval, On Change, Both	Interval
File-Based Archive Storage (Industrial Mode)	Store history in binary archive files (.hda)	Off
Archive Storage Path	Folder for the archive; empty = <code>historian-archives</code> in the project folder	empty
Default Compression Deviation	Swinging-door deviation used for tags without their own value; 0 = compression off	0 (see note)

Click **Save Configuration**. With the archive enabled, the page also shows **Archive Size**, **Total Records**, **Archived Tags** and a per-tag list (files, records, oldest and newest date).

Important: Enable **File-Based Archive Storage (Industrial Mode)** on every installation that uses the historian. In version 2.6.1.26 the historian worker always writes to the archive files, but the Historian view, trends, exports, comparison and correlation read from the archive only when this option is on. With the option off they read the local SQLite or external table, which the worker does not fill, and you see *No data available for the selected time range*.

Note: A saved **Default Compression Deviation** of 0 switches swinging-door compression off: every sample accepted by the recording method (interval, on change, deadband) is stored. Enter a value in engineering units to compress. On a new installation where the historian configuration has never been saved, the recorder uses 0.1 although the page shows 0 — click **Save Configuration** once so that the value shown is the value used. **Note — fixed in 2.6.1.25:** Up to 2.6.1.24 a deviation of 0 (default or per tag) was treated as 0.1, so compression could not be switched off.

19.2.1 Selecting tags to record

Tags are selected in the **Tag Recording Selection** list on the same page (search box, **Select All**, **Deselect All**; click a tag to toggle it — the badge shows its method and interval), or in the tag dialog (Chapter 7) under **Historian** → **Enable Recording**:

Tag field	Meaning	Default
Enable Recording	Record this tag	Off
Recording Method	Interval, On Change (Deadband), Both	Interval
Interval (ms)	Minimum time between samples (min. 100)	1000
Deadband	Minimum change in engineering units	0

The per-tag compression deviation can be set through the API only (`PUT /api/tags/<name>/historian` , field `historianCompressionDeviation`); a per-tag value of `0` switches compression off for that tag even when the default is higher. Tags without their own value use the default. The per-tag value is held in memory and is not kept across a restart of the service; after a restart the default applies again.

Note: Saving the historian configuration, enabling or disabling recording of tags (per tag or in bulk), purging historian data and deleting archive files require the `admin` role, as do creating, editing and deleting datalogs. Since 2.6.1.24 the server enforces this for every request; other users receive *Insufficient privileges*. Reading history (Historian view, trends, exports) is open to every logged-in user.

19.3 Recording methods, deadband and compression

19.3.1 Methods

Method	A sample is stored when...
Interval	the interval has elapsed and the value changed; an unchanged value is stored at most every 10 × interval, but at least every 30 s
On Change (Deadband)	the value differs from the last stored value by at least the deadband (0.001 if the deadband is 0)
Both	the interval has elapsed, or the change exceeds the deadband

19.3.2 Swinging-door compression

After the method has accepted a sample, and when the deviation is greater than 0, the swinging-door filter decides whether it is needed to reproduce the curve within $\pm deviation$. Samples that lie within the "door" opened from the last stored sample are discarded. A sample is forced at least every 5 minutes, so flat signals still produce regular points.

Signal	Suggested deviation
Slow temperature (°C)	0.1–0.5
Level in m	0.005–0.02
Flow, pressure	0.5–1 % of span

Signal	Suggested deviation
Digital 0/1	0 (use On Change)

Tip: Size the deviation from the instrument's accuracy. Compression tighter than the transmitter's noise only stores noise; much looser compression hides real events.

19.3.3 Storage estimate

Raw storage is 17 bytes per stored sample. 500 tags stored every second produce about 15.8×10^9 samples per year, roughly 270 GB raw. Interval recording of unchanged values, deadband and compression usually reduce this by a factor of 5–50; tiering (Section 19.4) reduces older data further.

19.4 Storage tiers and retention

19.4.1 Tiers

Open Config → **Data Storage** and click  **Historian Tiers** (the page opens inside the Admin tab). The page shows  **HOT (raw, full resolution)**,  **WARM (downsampled)** and  **COLD (gzipped)** with file count, size and records, and a **Configuration** form:

Field	Meaning	Default
Hot Retention (days)	Files newer than this stay as raw <code>.hda</code>	30
Enable Warm Tier	Create downsampled <code>.whda</code> files for older days. <i>Off</i> → skip warm; cold gzip kicks in immediately after hotDays.	On
Warm Bucket (ms)	10 seconds, 30 seconds, 1 minute (default) , 5 minutes, 10 minutes, 1 hour. <i>One avg sample per bucket.</i>	1 minute
Warm Retention (days)	Warm files older than this are deleted (the cold gzip copy remains)	365

Click **Save Configuration**. The tier job runs 30 s after start-up and every night at 00:05; **Run Migration Now** runs it immediately and reports *Warm generated*, *Warm purged* and *Cold gzipped*.

The nightly job, for each day older than **Hot Retention**:

1. creates a warm `.whda` file with one average per bucket (if the warm tier is on);
2. compresses the raw `.hda` into `.hda.gz` (cold) and removes the raw file;
3. deletes warm files older than **Warm Retention**.

The tier scheduler runs only when **File-Based Archive Storage** is enabled.

19.4.2 Which file a query reads

For each day in the requested range the historian reads the raw file if it exists, otherwise the cold `.hda.gz` (full resolution, decompressed on the fly), otherwise the warm `.whda` file. Old ranges therefore still return full-resolution data while the cold files exist, at the cost of decompression time.

19.4.3 Retention

- **Retention (days, 0=unlimited)** on the Historian page deletes raw `.hda` files older than that age (hourly check). **0** switches this deletion off: raw files are kept for ever. (Up to 2.6.1.24 a stored 0 was treated as 90 days; fixed in 2.6.1.25.) If no retention was ever saved, 90 days applies. Because the tier job converts raw files to `.hda.gz` after **Hot Retention** days, raw files rarely reach the retention age when tiering is active.
- Warm files are deleted after **Warm Retention**.
- **Cold `.hda.gz` files are not deleted by any automatic job** in version 2.6.1.26. Monitor the archive disk and archive or delete old cold files manually (by date in the file name) according to your data-retention policy (Chapter 26).

19.4.4 Historian backups

Historian archives are large and are backed up separately (**Historian Archive Backup**, Chapter 26).

19.5 The Historian view

Click  **Historian** in the header. The view has a tag panel on the left and the chart area on the right.

19.5.1 Selecting tags

The left panel lists historized tags by folder (*Search tags...*). Click tags to add them; up to **20** tags can be selected (**Tags (n/20)**). **Templates** save and restore a complete view: tags, time range, retrieval mode and interval, line style, auto-refresh, chart type, panes and axis choices. Click **Save**, type a name (*New template name...*) or choose **Overwrite existing...**; click a template to load it. Templates are stored in the browser of the station, not on the server.

19.5.2 Chart types

Type	Use
Trend	All tags on one time axis
Stacked	Several panes, each with its own Y axis. Drag tags from the left panel into a pane (<i>Drop tags here</i>); + Add Pane ; Remove a pane
X-Y Scatter	One tag against another (X-Axis , Y-Axis) — e.g. pump flow vs. discharge pressure
Batch	Overlay of batch runs aligned on <i>Time from batch start</i> ; select a recipe (All Recipes...) and runs (Chapter 20)

19.5.3 Time range and navigation

Control	Function
1h 6h 12h 24h 7d 30d 3M 6M 12M Custom	Time window ending now; Custom uses the two date-time fields and Query
◀ / ▶ (outer)	Move one whole window earlier / later
Step: ◀ Min / Hour / Day ▶	Move by one minute, hour or day
Drag on the chart	Zoom into a region; Reset Zoom or double-click to return
Caption under the X axis	Duration of the visible window

The 3M, 6M and 12M ranges show dates without times on the axis. Use a non-raw retrieval mode for these ranges (Section 19.5.4).

19.5.4 Retrieval modes

Mode	Returns per interval (1 min, 5 min, 15 min, 1 hour, 1 day)
Raw	Every stored sample (no interval)
Interpolated	Linearly interpolated value at each interval boundary
Average	Mean of the samples in the interval
Minimum / Maximum	Lowest / highest sample
Delta	Last minus first value in the interval
Count	Number of stored samples
Range	Maximum minus minimum

Line style: **Smooth, Straight, Step** (use **Step** for setpoints and digital signals).

19.5.5 Axis, alarms, annotations

- **Y-axis** fixed range: tick the fixed-axis box and enter **Min/Max**.
- Click a pen in the legend to hide or show it (*Click to hide pen*).
- **Alarms** overlays alarm events from the alarm history on the chart (count shown on the button). Alarms of related tags (same base name, e.g. `TT_101_HH` for `TT_101`) are included.
- **Annotate**: switch on, click the chart, type the text (*Enter annotation text...*) and **Save**. Annotations are stored in the station's browser only; they are not shared and not part of the audit trail.

19.5.6 Statistics, raw data and playback

Below the chart a statistics table shows per tag **Min, Max, Average, Std Dev** and **Count** for the loaded range; **Raw Data** shows the samples (**Timestamp, Tag, Value, Quality**, paged with **Prev/Next**). **Auto** refreshes the chart every 10 s in live mode. The playback bar (**Replay, Pause, Live**, speed 0.5×–10×, scrubber) replays the loaded data. **Screen Replay** opens the screen replay view, which plays a synoptic screen with historical values (writes are blocked in replay).

19.5.7 Storage info

The **Storage Info** panel shows **Enabled Tags, Data Source, Storage Mode, Archive Size, Total Records, Compression** and **Master**. If a query returns no data, the server adds diagnostics (archive folder, whether the tag folder exists, file count, oldest and newest file, live quality, last recorded time) — use them when troubleshooting (Chapter 30).

19.6 Period comparison

Compare overlays one tag over two periods.

1. Click **Compare**.
2. Choose the **Tag**.

3. Enter **Period 1 Start/End** and **Period 2 Start/End**, or use a preset: **Today vs Yesterday**, **This Week vs Last Week**, **This Month vs Last Month** (the second period has the same length as the first, so partial periods compare fairly).
4. Click **Compare**.

The chart overlays both periods on a common relative time axis (interpolated, 1-minute interval). The table compares **Period 1**, **Period 2** and **Difference** for min, max, **Average**, **Std Dev** and **Count**, plus the **Correlation** of the two curves.

19.7 Correlation analysis

Select 2–10 tags and click the correlation button (*Show correlation analysis between selected tags*). **Correlation Analysis** shows a matrix of Pearson correlation coefficients computed on 1-minute averages over the current time range: **Green** > 0.7 (strong positive), **Red** < -0.7 (strong negative), **Gray** weak. A second list shows the historized tags that correlate most strongly with the first selected tag (*No correlated tags found* if none).

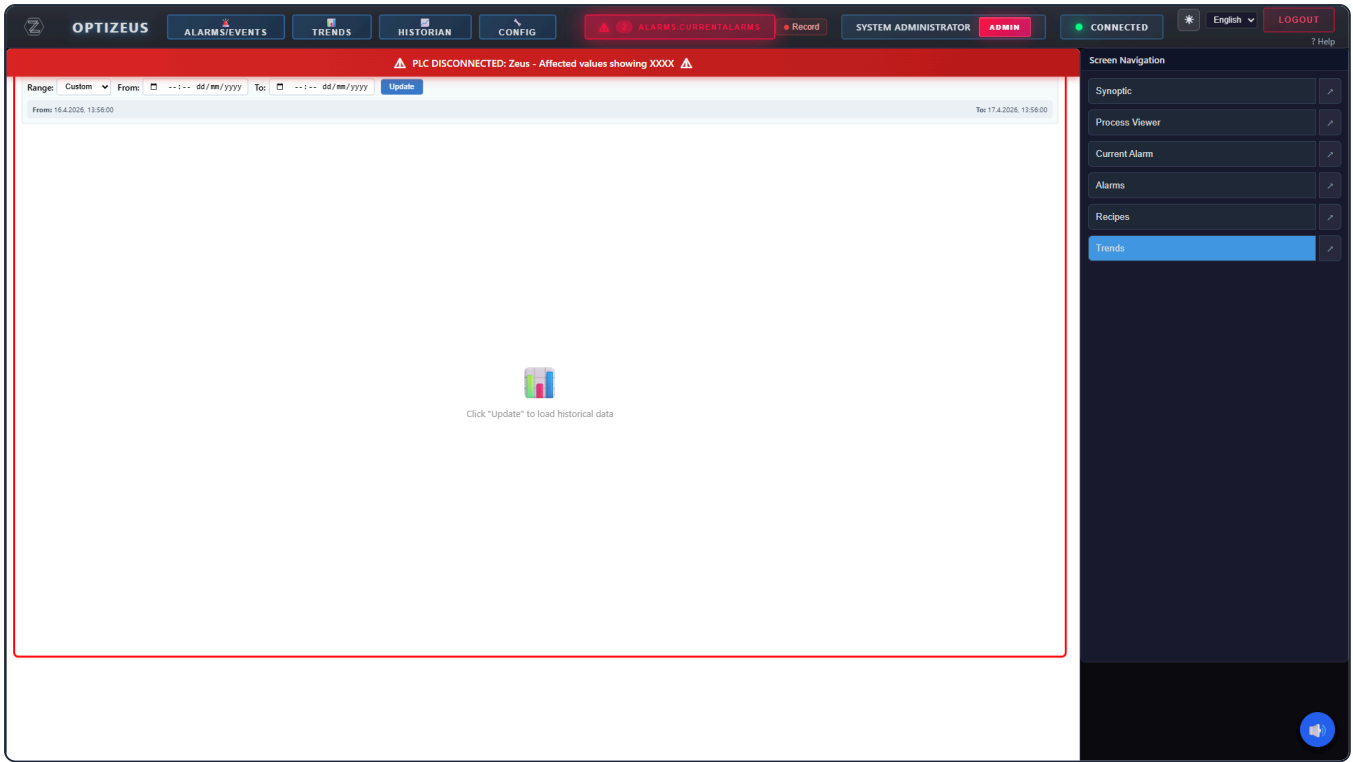
Note: The correlation is computed without time shift. Cause-and-effect relationships with a dead time (for example a valve and a downstream temperature) show a weaker coefficient than they really have; shorten or shift the time range to test such relationships. Correlation is not causation.

19.8 Export and printing

Button	Output
CSV	Server export of the selected tags and range (raw samples)
JSON	Same data in JSON through the data export API (Chapter 28)
PDF	Historian report with the chart and a statistics table; tick Data to include the data table
Export All Tags	CSV of the current value of every tag (a snapshot, not history)
Print	Browser print of the view

19.9 The Trends view and trend charts on screens

Trends (header button  **Trends**, title **Tag Trends**) is a live view for short-term observation. It does not read the historian: it plots the values the client has received since the page was opened, up to the last **60 samples per tag** (a new sample is added when a value changes), for at most 2000 tags.



Trend screen

Control	Options
Filter Mode:	Relative Time (Last 30 seconds ... Last 10 minutes) or Date Range
Sort:	Newest First, Oldest First
Chart Mode:	Individual Charts or Combined Chart
Tag Browser	Tree by folder (from / or . in the tag name); Expand All, Collapse All; Selected Tags, Clear All
Quick Select	Temperature, Pressure, Flow, Level — these buttons select fixed demonstration tag names (temperature1 , ...) and do nothing in a real project

Trend charts (in the Trends view and as chart objects on synoptic screens) support: click a legend entry to hide a pen, drag the brush under the chart to zoom, alarm markers from the alarm history (hover for details), and **Export to CSV**. Historical chart objects on screens read the historian for their configured duration (Chapter 12).

Tip: Use **Trends** for "what is happening now", and the **Historian** for anything older than a few minutes or for anything you need to keep, compare or export.

19.10 Performance notes

- **Choose the mode by range.** Raw mode over 30 days or more returns millions of points; use **Average** or **Interpolated** with 15 min–1 day intervals. The 3M/6M/12M ranges are designed for aggregated modes.
- **Cold data costs CPU.** Queries over days held only as .hda.gz decompress each day file. Expect long ranges over cold data to take seconds.

- **Keep the archive on a local SSD.** Network shares make every query slow.
- **Auto-refresh and reports.** Pages that refresh automatically re-run their query each cycle. Since version 2.6.1.10 visual reports do not start a new refresh while the previous one is still running, heavy batch statistics are refreshed at most every 15 s while a batch runs (once for a finished batch), and the server merges identical concurrent batch queries into one execution. Nevertheless, do not leave many stations with auto-refreshing long-range historian views or reports open at once.
- **Limit the historized tag count.** Record what you will actually analyse; do not enable **Select All** on a 30 000-tag project.
- **Monitor the recorder.** The server log reports the worker statistics every 30 s (`received` , `stored` , `compressed` , `buffered` tags); a large difference between received and stored is normal with compression.

19.11 Worked example: historizing Tank_101 and reviewing a shift

1. Config → **Data Storage** → **Historian**: tick **Master Enable - Historian Recording** and **File-Based Archive Storage (Industrial Mode)**, leave **Archive Storage Path** empty, set **Default Compression Deviation** to `0.01` , **Retention** `365` . **Save Configuration**.
2. In the tag dialog of `LT_101` tick **Enable Recording**, **Recording Method Both**, **Interval (ms)** `5000` , **Deadband** `0.01` . Do the same for `FT_101` (inlet flow, deadband `0.5`) and, with **On Change (Deadband)**, for `P201_Running` .
3. **Historian Tiers**: **Hot Retention** `30`, **Enable Warm Tier** on, **Warm Bucket** 1 minute, **Warm Retention** `365`. **Save Configuration**.
4. After a shift, open **Historian**, select `LT_101` , `FT_101` , `P201_Running` , chart type **Stacked**, drag `P201_Running` into a second pane, range **12h**, mode **Raw**, line style **Step** for the pump pane.
5. Switch on **Alarms** to see the `Tank_101_Level` alarm markers (Chapter 17).
6. Click **Save** under **Templates**, name it `Tank 101 shift review` .
7. Click **PDF** with **Data** unticked and attach the report to the shift log.
8. To compare with the previous day, click **Compare**, tag `LT_101` , preset **Today vs Yesterday**, **Compare**.

OPTIZEUS SCADA · OPERATION MANUAL

Part V — Production

20. Recipes and batch control (ISA-88)

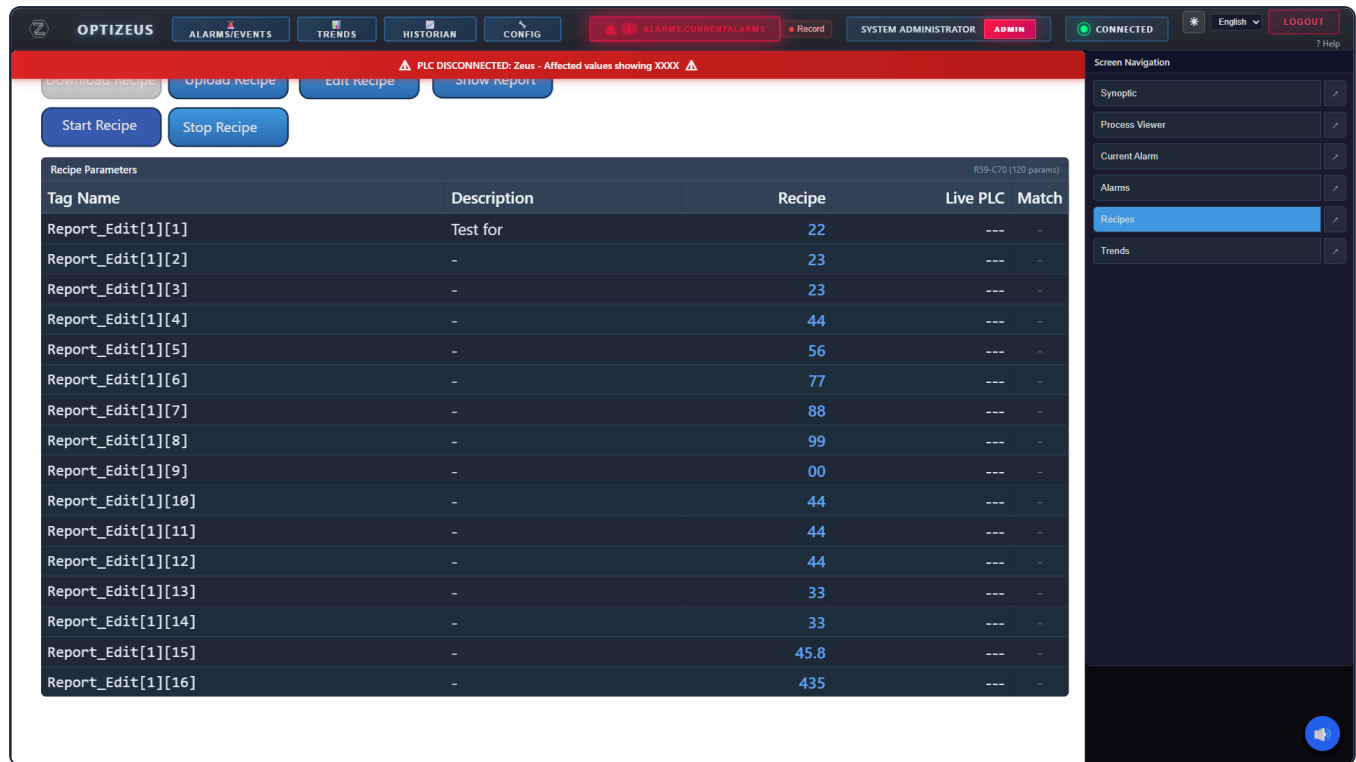
This chapter describes the two production-parameter tools in OptiZeus: the **Recipe Manager**, which stores named sets of tag/value pairs and writes them to a PLC in one action, and **Batch Control (S88)**, an ISA-88 style batch engine that runs Sequential Function Chart (SFC) recipes against phase logic in the PLC. Engineers need the whole chapter; supervisors need Sections 20.2, 20.4 and 20.9; operators need Sections 20.4.3, 20.9 and 20.10.

Note: The two tools are independent. A recipe from the Recipe Manager (a "parameter recipe") is **not** the same object as an SFC recipe in Batch Control, and neither one can be selected in the other. The batch reports described in Chapter 21 use a third mechanism, the *batch datalog*, which can take a snapshot of a parameter recipe (Section 20.6).

20.1 Which tool to use

You want to...	Use
Store the setpoints for Product A, B, C and write them to the PLC at changeover	Recipe Manager (Sections 20.2–20.5)
Let an operator pick a product on a screen and download it with one button	Recipe Manager + a Download Recipe click action on a synoptic object (Section 20.5)
Show recipe values next to the live PLC values on a screen	Recipe Table screen component (Section 20.5.3)
Record which recipe and which setpoints were used in a batch, for the batch report	Batch datalog + Attach Recipe to Batch click action (Section 20.6, Chapter 21)
Sequence phases (Fill, Heat, Mix, Drain) in the PLC from the server with hold/restart/abort, unit allocation and a phase log	Batch Control (S88) (Sections 20.7–20.11) — requires the Batch Control add-on licence

20.2 Recipe Manager overview



Recipe Manager

A parameter recipe has:

Element	Meaning
Recipe name	Unique name shown on the card. Must not contain a dot (.); the server rejects it with "Recipe name cannot contain dots (.) - use underscores (_) or hyphens (-) instead".
Description	Free text.
Category	Free text with auto-complete from the categories already used (for example Production, Test, Maintenance).
Target PLC	The PLC connection the recipe belongs to. The tag list in the editor is filtered to this PLC (plus tags with no PLC). A recipe without a target PLC cannot be downloaded or uploaded.
Status	Enabled (Released) or Disabled (Draft) . Draft recipes show a yellow DRAFT badge and are hidden from runtime download lists (Section 20.5) and, since 2.6.1.26, refused by the server on download.
Security Role (Required to Download)	No restriction or Role A ... Role Z . Shown as a blue Role X badge. Checked by the server on download since 2.6.1.26 (Section 20.2.2).
Parameters	An ordered list of rows: Tag , Value , Description . At least one is required.

There is no separate version number, approval workflow, min/max validation or comparison view for parameter recipes. Every save overwrites the recipe; the change is recorded in the event log (`recipe_created` , `recipe_updated` , `recipe_deleted` , `recipe_downloaded` , `recipe_uploaded` , `recipe_imported`). If you need a controlled release, use **Duplicate** to create a new draft (the copy starts disabled), edit it, and enable it when approved. Electronic approval with signature exists only for Batch Control SFC recipes (Section 20.8.5).

20.2.1 Where the Recipe Manager is

Location	Who
Config → Recipes tab	Administrators (the Config view is administrator-only by default)
A synoptic object with the click action Open Recipe Manager	Anyone who can open that screen; the server still enforces the permissions below

20.2.2 Permissions

Operation	Roles allowed by the server
List recipes, read one recipe, read live values, export to Excel	Any authenticated user
Create, update, delete, import from Excel	admin
Download to PLC	operator , admin (since 2.6.1.26)
Upload from PLC	admin

The server checks the following when a download is requested (since 2.6.1.26), whether it comes from the Recipe Manager, a synoptic **Download Recipe** button or the REST API:

Check	Refusal
Role operator or admin	<i>Insufficient privileges</i> (HTTP 403) for a viewer.
Recipe Status is Enabled	A disabled (draft) recipe is refused: <i>Recipe "<name>" is disabled (draft) and cannot be downloaded</i> (HTTP 409).
Security Role (Required to Download)	If the recipe has a letter, a user who is not admin must hold that security letter in their role (Chapter 24); otherwise <i>Recipe "<name>" requires security letter X</i> (HTTP 403) and the event <code>recipe_download_denied</code> is logged.

Note — changed in 2.6.1.26: Up to 2.6.1.25 only the admin role could download a recipe (an operator pressing a **Download Recipe** button received *Insufficient privileges*), and neither the recipe's **Status** nor its security letter was checked by the server. After upgrading, operators can download every enabled recipe that has no security letter; give recipes a letter where downloads must be restricted. **Upload from PLC** remains administrator-only.

20.3 Creating and editing a parameter recipe

To create a recipe:

1. Click **Config** → **Recipes**.
2. Click + **New Recipe**.
3. Under **Basic Information**, enter the **Recipe Name** (required), and optionally **Description** and **Category**.
4. Select the **Target PLC**.
5. Under **Release & Security**, set **Status** and **Security Role (Required to Download)**.
6. Under **Recipe Parameters**, add each parameter (Section 20.3.1).
7. Click **Save Recipe**.

The form fields:

Field	Meaning	Default
Recipe Name *	Name, no dots	empty (placeholder "Product A Settings")
Description	Free text	empty
Category	Free text, suggestions from existing categories	empty
Target PLC	PLC connection	Select PLC... (none)
Status	Checkbox: checked = Enabled (Released) , cleared = Disabled (Draft) . Hint: "Only enabled recipes can be downloaded to PLC at runtime"	Enabled
Security Role (Required to Download)	No restriction, Role A ... Role Z. Hint: "User must have this role or higher to download recipe". The server checks that the user's role holds this security letter; <code>admin</code> always passes.	No restriction

20.3.1 Adding parameters

The add-parameter row has:

Field	Meaning
Tag	Drop-down of configured tags, shown as <i>label (type)</i> . Filtered to the target PLC.
Index 1 (Row), Index 2 (Col), Index 3 (Depth)	Appear only for array tags (<code>Array[1D]</code> , <code>Array[2D]</code> , <code>Array[3D]</code>). The placeholder shows the valid range.
Value	The value to write. Free text; it is sent to the PLC as typed. With touch-screen mode on, the on-screen keyboard opens.
Description (optional)	Up to 600 characters.

Click **+ Add Parameter**. The parameter is appended to the table. For an array tag the stored tag name includes the index, for example `Zone_SP[3]` or `Profile[2][4]` . A tag (or array element) can appear only once in a recipe; a duplicate gives "*This tag element is already in the recipe*".

For array tags, **Add All Elements (N)** adds one row per element in a single step (after a confirmation), using the array dimensions and base index from the tag configuration. Values are left empty; fill them in the table.

In the parameter table you can edit **Value** and **Description** in place, or click **Remove** to delete a row. Row order is the order of writing during a download.

Tip: Keep recipe values in the engineering form the tag expects. The value is passed unchanged to the tag write routine, which applies the tag's own data-type conversion and scaling (Chapter 7).

20.3.2 Recipe card actions

Each recipe card shows the name, badges, description, **Parameters:** count, **PLC:** name, the first three parameters and "+N more". Buttons:

Button	Action
Download to PLC	Writes every parameter to the PLC (Section 20.4). Disabled when no PLC is assigned.

Button	Action
Upload from PLC	Replaces every parameter value in the recipe with the current value of its tag (Section 20.4.2).
Edit	Opens the form. Update Recipe saves.
Duplicate	Opens the form pre-filled with " <i>name</i> (Copy)", status Draft. Save as New Recipe creates it.
Export	Downloads <code>recipe_<name>.xlsx</code> (Section 20.3.3).
Delete	Deletes after the confirmation "Delete this recipe?".

20.3.3 Excel export and import

Function	Button	File layout
Export one recipe	Export on the card	Sheet Recipe Info (Field/Value rows: Recipe Name, Description, Category, Enabled, Export Date, Version) and sheet Parameters (columns #, Tag Name, Value, Description).
Export all recipes	Export All (Excel)	One sheet per recipe, named after the recipe (max. 31 characters), columns #, Tag Name, Value, Description. File <code>all_recipes_<date>.xlsx</code> .
Import	Import (Excel) (<code>.xlsx</code> , <code>.xls</code> , max. 10 MB)	If the workbook has a Recipe Info sheet, one recipe is created from it (name in B2, description B3, category B4) with parameters from the Parameters sheet. Otherwise every sheet with rows becomes a recipe named after the sheet. Parameter rows are read from column B (tag) and C (value), D (description), starting at row 2.

Imported recipes are always created as **new**, **enabled**, and **without a target PLC**. Edit each imported recipe and select the **Target PLC** before it can be downloaded. Import never updates an existing recipe.

Note: The **Enabled** row in an exported **Recipe Info** sheet always shows "Yes", and the **Version** row shows a fixed text, not the running version. Do not use the export as evidence of release status.

20.4 Download and upload

20.4.1 Download to PLC

WARNING: A download writes every parameter directly to the running PLC, in list order. The only range check is the tag's own clamping to its limits (Chapter 7), when it is switched on. Verify that the equipment is in a state where the new setpoints are safe (for example, idle between batches) before you download. A partially successful download leaves the PLC with a mix of old and new values.

To download a recipe from the Recipe Manager:

1. Make sure the recipe has a **Target PLC** and at least one parameter.
2. Click **Download to PLC** on the recipe card. The button shows **Downloading...**
3. Read the result message:
 - "Recipe downloaded successfully: *N* parameters written to PLC" — all writes succeeded.
 - "Partial download: *N* succeeded, *M* failed" — some writes failed. Each failed write is logged as a `tag_write_failed` event with its error, and (from a synoptic button) the failed tags are listed in the result dialog.

The server writes each parameter through the same code as a manual tag write (`POST /api/tags/<name>/write`), so every protocol that supports tag writes (Chapter 6) is supported — including EtherNet/IP (Logix5000 and MicroLogix/PCCC) — with the same scaling, Memory/Simulation tag handling and failure reporting. Security areas (Chapter 25) are not applied to the individual parameter writes. Each parameter write is logged as a `tag_write` event with the user who started the download and the reason *Recipe download <name>*; the download as a whole is logged as event `recipe_downloaded` with the success and failure counts.

Note — changed in 2.6.1.26: Up to 2.6.1.25 recipe downloads used a separate, older write routine that did not write to EtherNet/IP connections and recorded the parameter writes without the user.

20.4.2 Upload from PLC

Upload from PLC does not read the PLC again; it copies the *current value held by the server* for each parameter tag into the recipe and saves the recipe. Array elements are taken from the cached array value. Parameters whose tag has no current value keep their previous value. The message *"Recipe updated with N current PLC values"* counts all parameters, not only the changed ones.

Use upload to capture a "golden batch" set of setpoints: tune the process, then upload into a new (duplicated) recipe.

20.4.3 Downloading from a synoptic screen (operator procedure)

When the engineer has placed a button with the click action **Download Recipe** (Chapter 12):

1. Click the button. The recipe selection window lists only **enabled** recipes. You need the `operator` or `admin` role, and the recipe's security letter if it has one (Section 20.2.2).
2. Select the recipe.
3. Confirm the message *"Download recipe "<name>" to PLC? This will write N parameter values to the PLC."*
4. Read the result: *"Recipe "<name>" downloaded successfully! ✓ N parameters written to PLC"*, or a list of the failed tags with their errors.

After a successful download, every **Recipe Table** on the client switches to the downloaded recipe (the choice is remembered in the browser).

20.5 Recipes on screens

20.5.1 Click actions

In the Viewer Builder, the **Click Action** list of an object has these recipe-related actions (see Chapter 12 for the common action fields):

Click action	Property	Runtime behaviour
Download Recipe	Select Recipe — "Only enabled recipes are shown. At runtime, user will confirm before download."	Downloads the recipe after confirmation.
Upload Recipe (PLC → Recipe)	Select Recipe — "Reads current PLC values and saves them into the selected recipe. User will confirm before upload."	Copies current values into the recipe.
Open Recipe Manager	none	Opens the full Recipe Manager panel.

Click action	Property	Runtime behaviour
Attach Recipe to Batch	Batch Datalog, Recipe	Snapshots the recipe into the active batch of a batch datalog (Section 20.6).

20.5.2 Security

Give recipe buttons a security letter (Chapter 25) so that only authorised users see or can press them. The server additionally checks the recipe's own security letter on download, and requires the `admin` role for upload (Section 20.2.2).

20.5.3 Recipe Table component

The **Recipe Table** component (Viewer Builder component library) shows the parameters of one recipe with their recipe value and the live value side by side.

Property	Meaning	Default
Recipe	The recipe to display (list shows name, category and parameter count)	none
Title	Header text	Recipe Parameters
Font size, row height, header background/text, text colour	Appearance	12, normal, #334155 , #f1f5f9 , #e2e8f0
Show description	Adds the Description column	off
Show match indicator	Marks each row where the live value differs from the recipe value; the header blinks while there are mismatches	on

The live values are read from the server's tag cache (no PLC read). When an operator downloads a recipe from a synoptic button, the Recipe Table switches to that recipe.

20.6 Recording the recipe in a batch record

For batch reporting (Chapter 21) OptiZeus uses *batch datalogs* (Config → **Data Storage, Datalog Type = Batch**, Chapter 19). A batch datalog writes a header row per batch (`batch_number` , `username` , `start_datetime` , `end_datetime`) to `<table>_header` and data rows to `<table>_data` . The batch starts and stops on the **Batch Start/Stop Tag** (TRUE = running).

The **Attach Recipe to Batch** click action stores, on the header row of the batch that is currently running, the recipe name (`recipe_name`) and a JSON snapshot of all its parameters (`recipe_snapshot`). The visual report can then print the setpoints with `{{recipe.TAG}}` and the recipe name with `{{batchInfo.recipeName}}` (Chapter 21).

- The action requires a running batch; otherwise the server answers "*No active batch — recipe attach ignored*".
- The user must have the `operator` or `admin` role; a viewer is refused with *Insufficient privileges* (since 2.6.1.24). The same applies to the batch-datalog **snapshot** and **close batch** requests.
- The snapshot is taken from the recipe as stored, not from the PLC.
- The server adds the two columns to the header table automatically (SQL Server, MySQL and PostgreSQL-type data sources).

Tip: Put **Download Recipe** and **Attach Recipe to Batch** on the same screen and have the operator press them in that order after the batch has started, or start the batch from the PLC and attach from a script. The attach does not download anything.

20.7 Worked example: a 5-parameter mixing recipe

Plant: mixing tank Tank_101 with agitator, heating jacket and dosing pump on Line 1, controlled by a PLC connection named `PLC_Line1`. The PLC holds five setpoints, already configured as OptiZeus tags (Chapter 7):

Tag	Type	Meaning	Product A	Product B
Tank_101_Fill_SP	Float	Fill volume, L	1200	1500
Tank_101_Temp_SP	Float	Mix temperature, °C	65.0	72.5
Tank_101_Agit_Speed_SP	Int	Agitator speed, rpm	180	220
Tank_101_Mix_Time_SP	Int	Mix time, s	900	1200
Tank_101_Additive_SP	Float	Additive dose, kg	4.5	6.0

Step 1 — create Product A:

1. **Config** → **Recipes** → **+ New Recipe**.
2. **Recipe Name:** `Mix_ProductA` (no dots). **Category:** `Production`. **Target PLC:** `PLC_Line1`.
3. Leave **Status** checked (Enabled). Leave **Security Role** at **No restriction**.
4. For each of the five tags: select it in **Tag**, type the Product A value in **Value**, type a description (for example "Fill volume, L"), click **+ Add Parameter**.
5. Click **Save Recipe**. The card shows *Parameters: 5, PLC: PLC_Line1*.

Step 2 — create Product B from A:

1. On the `Mix_ProductA` card click **Duplicate**.
2. Change the name to `Mix_ProductB` and edit the five values in the table.
3. Click **Save as New Recipe**. The copy is created as a draft (**DRAFT** badge).
4. After the recipe has been reviewed, click **Edit**, check **Status**, click **Update Recipe**.

Step 3 — operator screen:

1. In the Viewer Builder open the Tank_101 screen.
2. Add a button "Load recipe", click action **Download Recipe**, and select `Mix_ProductA`. Add a second button for `Mix_ProductB`. Give both buttons a security letter.
3. Add a **Recipe Table** bound to `Mix_ProductA` with **Show match indicator** on.
4. Save and deploy (Chapter 5).

Step 4 — test at a safe moment (tank empty, agitator stopped): press "Load recipe", confirm, and check that the Recipe Table shows no mismatches. Check the event log for `recipe_downloaded ... 5 success, 0 failed`.

Step 5 — record it in the batch report: create a batch datalog `Mix101` on Tank_101 with **Batch Start/Stop Tag** `Tank_101_Batch_Running` (Chapter 19). Add a button with **Attach Recipe to Batch** (datalog `Mix101`, recipe `Mix_ProductA`). In the batch report (Chapter 21), a text element `Temperature SP: {{recipe.Tank_101_Temp_SP}}` °C prints 65.00.

20.8 Batch Control (S88) overview

Batch Control is a server-side batch engine modelled on ISA-88. The PLC runs the phase logic; the OptiZeus server sequences the phases through a *Phase Logic Interface* (PLI) of handshake tags, following an SFC recipe.

20.8.1 Licence and availability

Condition	Effect
The licence does not contain the add-on <code>batch-control1</code> ("Batch Control (S88)")	Config → Batch Control shows "Batch Control (S88) - Add-on License Required". The server refuses the phase-class list and Start New Batch with " <i>Batch Control (S88) requires a Batch Control add-on license.</i> "
Starting a batch in trial mode	Subject to the trial restrictions (Chapter 2).
Server role in a multi-server topology	The engine runs only on servers whose topology activates the batch engine service (Chapter 27). On other servers the batch pages do not work.

The add-on is sold on top of a licence tier (see Appendix C).

20.8.2 Where Batch Control is

Location	Notes
Config → Batch Control tab	Full editor (administrators).
Header button Batch Control	Opens the same view full-screen; add the button in Layout Builder → Header or Config → Display → Navigation Buttons (Chapter 14).

The view is titled **S88 Batch Control (PLI)** and has six tabs: **Execution**, **SFC Recipes**, **Phase Classes**, **Equipment**, **Materials**, **History**.

20.8.3 Permissions

Operation	Roles
Create, edit, delete phase classes, equipment, materials, SFC recipes	admin
Approve an SFC recipe	admin
Start, pause, resume, hold, restart, stop a batch; confirm a manual transition	admin , operator
Abort a batch	admin
View active batches, history and logs	Any authenticated user

20.8.4 Model

S88 concept	OptiZeus object	Where
Physical model: Process Cell > Unit > Equipment Module > Control Module	Equipment tree	Equipment tab
Phase (equipment phase in the PLC)	Phase class with a PLI tag suffix and parameters	Phase Classes tab

S88 concept	OptiZeus object	Where
Procedure / recipe	SFC recipe: steps, transitions	SFC Recipes tab
Recipe type	General, Site, Master, Control (label only; no functional difference)	Recipe properties
Materials	Material catalogue	Materials tab
Batch	Batch instance, running on one unit	Execution tab
Batch record	Batch instance + phase log	History tab

20.8.5 Recipe approval and electronic signature

SFC recipes carry the fields `approved`, `approved_by` and `approved_at`. Approval is available through the REST API only (`POST /api/batch/recipes/{id}/approve`, `admin` role, Chapter 28); there is no approve button in the Batch Control view in this version. When **electronic signatures** are required in the security settings (Chapter 25), the request must include the user's password; the server stores a signature with the meaning "I approve this recipe for production use" and writes an audit entry. A wrong password is refused and logged as `esig_failed`.

Important: In version 2.6.1.26 the engine does not require a recipe to be approved before a batch is started; only **Enabled** is checked. If your procedures require approval before use, enforce it organisationally (for example, keep unapproved recipes disabled).

20.9 Configuring Batch Control

Configure in this order: phase classes, then equipment, then SFC recipes.

20.9.1 Phase classes

A phase class describes one kind of phase that the PLC can run (Fill, Heat, Agitate, Drain...).

To create a phase class:

1. **Phase Classes** tab → **New Phase Class**.
2. Fill in the fields below. Click + **Add Parameter** and + **Add Report Param** for each parameter.
3. Click **Save**.

Field	Meaning
Name (required)	Display name, e.g. <code>Fill</code> .
PLI Tag Suffix (required)	The middle part of the PLI tag names, e.g. <code>FILL</code> .
Category	Free text, e.g. <code>Transfer</code> .
Description	Free text.
Parameters (Server → PLC: P1, P2, ...)	Rows of Name, type (REAL , INT , BOOL), Default, Units. The first row is written to <code>_P1</code> , the second to <code>_P2</code> , and so on.
Report Parameters (PLC → Server: RP1, RP2, ...)	Rows of Name, type (REAL , INT), Units. Read from <code>_RP1</code> , <code>_RP2</code> , ... while the phase runs.

20.9.2 The Phase Logic Interface (PLI)

For a step that uses a phase class on a unit, the engine builds the tag base `<UnitTagPrefix>_<PLISuffix>` (if the unit has no tag prefix, its name is used). Example: unit prefix `TK101`, suffix `FILL` → base `TK101_FILL`. The following OptiZeus tags must exist, with exactly these names, and be connected to the PLC phase logic:

Tag	Direction	Meaning
<code><base>_CMD</code>	Server → PLC	Command: 0 = None, 1 = Start, 2 = Stop, 3 = Abort, 4 = Hold, 5 = Restart, 6 = Reset
<code><base>_STATE</code>	PLC → Server	Phase state: 0 Idle, 1 Running, 2 Complete, 3 Paused, 4 Holding, 5 Held, 6 Restarting, 7 Stopping, 8 Stopped, 9 Aborting, 10 Aborted
<code><base>_OWNER</code>	Server → PLC	Numeric database ID of the batch that owns the phase
<code><base>_STEP</code>	PLC → Server	Current step number inside the phase (shown and logged)
<code><base>_MSG</code>	PLC → Server	Message text from the phase (shown and logged)
<code><base>_P1 ... _Pn</code>	Server → PLC	Phase parameters, written as numbers before Start
<code><base>_RP1 ... _RPn</code>	PLC → Server	Report parameters, collected while the phase runs

When a step is activated the engine writes, in this order: `_OWNER`, `_P1 ... _Pn` (step value, else the phase class default, else 0), then `_CMD` = 1. It then reads `_STATE` every second. When the batch ends (Complete, Stopped or Aborted) the engine writes `_CMD` = 6 (Reset) to the phases that are still active.

WARNING: In version 2.6.1.26 the batch engine's own write routine sends values to the PLC only for tags on **OPC UA, Modbus TCP, BACnet** and **MQTT** connections. (This routine was not changed in 2.6.1.26; parameter-recipe downloads, automation actions and pipelines now use the full operator write path, but batch PLI writes do not.) For tags on other drivers (for example Siemens S7 or Allen-Bradley/EtherNet/IP) the value is changed in OptiZeus only and **is not written to the PLC**, and no error is shown. Before commissioning, verify with the PLC programming tool that `_CMD`, `_OWNER` and `_Pn` actually arrive in the PLC. On Modbus TCP, parameter values are written to a single holding register as integers; scale REAL parameters accordingly in the PLC.

Note: If a PLI tag does not exist in OptiZeus, the write is skipped with only a server log message, and a missing `_STATE` tag reads as 0 (Idle) — the step will then never complete. Create all PLI tags before the first run.

20.9.3 Equipment

To create the equipment hierarchy:

1. **Equipment** tab → **New Equipment** (top level) or **+ Child** on an existing node.
2. Fill in the fields and click **Save**.

Field	Meaning
Name	e.g. REACTOR-01 , TK101 .
Type	Process Cell, Unit, Equipment Module, Control Module. Only Unit equipment can run a batch.
Tag Prefix	Prefix of the PLI tags, e.g. TK101 . If empty, the name is used.
Parent	Parent node, or -- None (Top Level) --.
Description	Free text.
Supported Phase Classes	(Units only) Checkboxes. Informational in this version: the engine does not reject a recipe that uses a phase class the unit does not support.

Each node shows its status. A unit is **AVAILABLE** until a batch starts on it; it is then **ALLOCATED** and cannot start a second batch. It returns to **AVAILABLE** when the batch completes, stops or aborts.

20.9.4 Materials

Materials tab → **New Material: Name, Type** (e.g. Raw Material, Additive), **Unit of Measure** (kg, L...), **Description, Lot Tracking Enabled**. The list shows Name, Type, UOM, Lot Tracking.

Note: The material catalogue is reference data only in this version. The SFC editor does not assign materials to steps and saves an empty bill of materials, so no material consumption is recorded.

20.9.5 SFC recipes

To create an SFC recipe:

1. **SFC Recipes** tab → **New SFC Recipe**. The editor opens with an initial step **Start**, a final step **End**, and an **Auto** transition between them.
2. Enter the recipe name and **Version** in the toolbar.
3. With nothing selected, set the **Recipe Properties: Description, Category, Recipe Type** (General, Site, Master, Control), **Enabled**.
4. Click **+ Step**, then click the canvas to place a step. Steps snap to a 20-pixel grid. Drag steps with **Select** active.
5. Click **+ Transition**, click the source step, then the target step.
6. Click a step to edit its **Step Properties**, or a transition to edit its **Transition Properties** (tables below).
7. Select an element and click **Delete** to remove it (deleting a step also deletes its transitions).
8. Click **Save Recipe**. Click **Back** to return to the list.

Step properties:

Field	Meaning
Name	Shown in the step box (15 characters visible).
Type	Initial, Step, Final. One initial step starts the batch; a final step without a phase class ends it.
Phase Class	The phase this step runs, or -- None --.
Unit Requirement	Text such as reactor . In this version all steps run on the unit selected when the batch starts.
Phase Parameters	One numeric field per phase-class parameter, pre-filled with the default.

Transition properties:

Condition	The transition fires when...
Auto (immediate)	the source step is finished (immediately for an initial step without a phase).
Phase Complete	the source phase reports <code>_STATE = 2</code> (Complete). This is the default for new transitions.
Tag Condition	the Expression is true, in the form <code>TAG_NAME <op> number</code> with <code>==</code> , <code>!=</code> , <code>>=</code> , <code><=</code> , <code>></code> , <code><</code> , for example <code>LEVEL >= 80</code> . If the tag name does not contain the unit's tag prefix, the prefix is added (<code>LEVEL</code> becomes <code>TK101_LEVEL</code>).
Manual Confirm	an operator confirms it (REST API only in this version: <code>POST /api/batch/{batchId}/confirm-transition</code> with the transition ID).

Important: Transitions are evaluated only when the source step's phase reports Complete (or immediately for steps without a phase). The editor draws simple chains: a step with **one** outgoing transition. The engine also supports parallel and alternative (selection) branches, but these must be defined in the recipe JSON through the REST API (`parallelBranches` , `selectionBranches`); a step with more than one outgoing transition and no branch definition never advances.

20.10 Running a batch (operator procedure)

To start a batch:

1. Open **Batch Control** → **Execution**.
2. Click **Start New Batch**.
3. Select the **Recipe** (only enabled recipes, shown with their version) and the **Unit** (units that are not AVAILABLE are greyed out).
4. Click **Start**. The message "*Batch BATCH-<number> started*" appears.

The batch ID has the form `BATCH-<timestamp>-<6 characters>`. The unit is allocated, the batch enters **RUNNING**, and the initial step becomes active.

Each active batch is a card with the batch ID, state badge, recipe, unit and a progress bar (completed non-initial steps ÷ all non-initial steps). The page refreshes every 2 seconds. Click a card to expand it:

- **SFC visualization:** active steps green (pulsing), completed steps blue, others grey.
- **Active Phases (PLI):** per active step, **State:** (PLI state), **Step:** and the PLC message.
- Command buttons according to the batch state.

Batch state	Buttons shown	Effect
RUNNING	Pause, Hold, Stop, Abort	Pause: writes Hold (4) to active phases, batch → PAUSED. Hold: writes Hold, batch → HELD. Stop: writes Stop (2), batch → STOPPED. Abort: writes Abort (3), batch → ABORTED.
PAUSED	Resume, Abort	Resume writes Restart (5), batch → RUNNING.
HELD	Restart, Abort	Restart writes Restart (5), batch → RUNNING.

States and allowed transitions of the batch:

From	To
IDLE	RUNNING
RUNNING	PAUSING, HOLDING, STOPPING, ABORTING, COMPLETE
PAUSING	PAUSED, ABORTING
PAUSED	RUNNING, STOPPING, ABORTING
HOLDING	HELD, ABORTING
HELD	RESTARTING, STOPPING, ABORTING
RESTARTING	RUNNING, ABORTING
STOPPING	STOPPED, ABORTING
STOPPED	ABORTING
ABORTING	ABORTED
COMPLETE, ABORTED	(final)

The PLC can also drive the batch: when an active phase reports Held (5), Stopped (8) or Aborted (10), the batch goes to HOLDING, STOPPING or ABORTING respectively.

WARNING: Batch commands act on the PLC phases immediately. **Abort** is available only to administrators. Hold and Abort from the server do not replace the PLC's own safety interlocks.

Batches that were active when the server stopped are reloaded at start-up and continue from their saved state.

20.11 Batch history and batch records

The **History** tab lists the last 50 finished batches: **Batch ID, Recipe, Unit, Status, Progress, Started, Ended**. Click **Refresh** to reload.

Button	Result
Log	Expands the row with the SFC picture of the batch and the Phase Log table: Step, Phase Class, Unit, PLI State, PLI Step, Message, Status, Start, End.
Report	Opens a <i>Batch Execution Report</i> in a new window: Batch Summary (ID, recipe, unit, status, progress, duration, start, end) and the phase execution log. Print / PDF prints it (use the browser's "Save as PDF" printer); Export CSV saves <code>batch-report-<time>.csv</code> .

Batch start and finish are also written to the event log (`batch_started` , `batch_finished` , the latter as a warning for STOPPED/ABORTED).

20.12 REST API summary

Method and path	Purpose						
GET /api/recipes , GET /api/recipes/{id}	Parameter recipes						
POST /api/recipes , PUT /api/recipes/{id} , DELETE /api/recipes/{id}	Maintain (admin)						
POST /api/recipes/{id}/download	Download (operator, admin; enabled recipes only; security letter checked)						
POST /api/recipes/{id}/upload	Upload (admin)						
GET /api/recipes/{id}/live-values	Current tag values for the recipe's parameters						
GET /api/recipes/{id}/export-excel , GET /api/recipes/export-all/excel , POST /api/recipes/import-excel	Excel						
POST /api/datalogs/{id}/attach-recipe	Snapshot a recipe into the active batch of a batch datalog (operator, admin)						
GET /api/batch/license-check	Is the add-on licensed?						
/api/batch/phase-classes , /api/batch/equipment , /api/batch/materials , /api/batch/recipes	CRUD						
POST /api/batch/recipes/{id}/approve	Approve (admin; password when e-signatures are on)						
POST /api/batch/start (recipeId , unitId)	Start a batch						
POST /api/batch/{batchId}/pause	resume	hold	restart	stop	abort	confirm-transition`	Commands

Method and path	Purpose
GET /api/batch/active , GET /api/batch/history , GET /api/batch/{batchId}/status , GET /api/batch/{batchId}/log	Status and records

Example — start a batch:

```
POST /api/batch/start HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{ "recipeId": 3, "unitId": 7 }
```

See Chapter 28 for authentication.

21. Reports

This chapter describes how to produce printed and exported reports in OptiZeus: template reports from the **Report Designer**, free-layout reports from the **Visual Report Builder**, batch reports built on batch datalogs, the **Shift Handover Report**, scheduled reports sent by e-mail, and export to PDF and Excel. Engineers who design reports need the whole chapter; supervisors and operators who only print or export reports need Sections 21.1, 21.5 and 21.7.

21.1 Report tools at a glance

Tool	Where	What it produces	Typical use
Report Designer ("Template Reports")	Config → Reports → Template Reports	Fixed-layout A4 PDF: title, date range, chart, statistics, data table, custom text pages	Quick tag-data, event, alarm, batch or SQL-query reports
Visual Report Builder	Config → Reports → Visual Report Builder	Free page layout (A4/Letter, portrait/landscape) with text, tables, charts, statistics, alarms, images, formulas; PDF and Excel export	Customer-format batch reports, daily production logs, shift reports
Show Visual Report click action	Synoptic screen object	Opens a saved visual report for the operator, optionally for the current batch	Operator prints the batch report at the end of a batch
Shift Handover Report	Header button Shift Report	On-screen summary + PDF: alarms, events, tag quality, PLC connections	Shift handover
Scheduled reports	REST API only (no configuration screen)	HTML shift summary, e-mailed	Daily automatic shift summary
Batch Execution Report	Batch Control → History → Report	Printable batch summary and phase log, CSV	S88 batch records (Chapter 20)
OEE reports	OEE → Reports tab	Averages and snapshot table	Chapter 22

All report tools read their data from *datalogs* (Chapter 19), the event log, the alarm history and — for named queries — from SQL data sources. A report can only show what has been logged: configure the datalog first.

21.1.1 Permissions

Operation	Roles
List and run template reports and visual reports; preview; export PDF/Excel	Any authenticated user who can open the view
Create, edit, delete report templates and visual report layouts	admin
Create, edit, delete, run scheduled reports	admin
Shift Handover Report	Any authenticated user

The **Config** view is shown to administrators by default, so in practice operators run reports through synoptic buttons (**Show Visual Report**) and the **Shift Report** header button.

21.2 Report Designer (template reports)

Config → **Reports** → **Template Reports** opens the **Report Designer**. Each saved template is a card showing its name, type (or datalog name), tag count, date range and the badges **Chart**, **Table**, **Stats**. Card buttons: **Preview**, **Generate PDF**, **Edit**, **Delete**.

To create a template:

1. Click **+ New Report Template**.
2. Enter **Report Name** (required) and **Description**.
3. Select the **Report Type** (Section 21.2.1) and fill in its section.
4. Set the **Date Range** (not used by batch reports that select batches).
5. Set **Report Options**, **PDF Page Layout** and optional **Custom Pages**.
6. Click **Preview** to check the data, then **Save Template** (**Update Template** when editing).

21.2.1 Report types

Report Type	Data	Specific fields
Datalog (Tag Data)	Rows of a standard datalog	Datalog; Tags to Include (checkboxes with select-all / clear)
Batch Report	Batches of a batch datalog	Batch Datalog; Batch Name Tag; Detail Tags; Detail Mode: By Time (show data points over time) or By Events (show events during batch)
Events	Event log	Date range only
Alarms	Alarm history	Source: Date Range or Batch (use batch time range) — then Batch Datalog and Batch
Named Query (SQL)	A saved named query (Chapter 28)	Saved Query ; one input per query parameter (:name (type))

21.2.2 Common options

Field	Meaning	Default
Range Type	Last Hour, Last 24 Hours, Last Week, Last Month (30 days), Custom Range (Start Date, End Date)	Last Hour
Include Chart	Datalog and batch reports	on
Chart Type	Line Chart, Area Chart	Line
Include Data Table	All types	on
Include Statistics	Min / max / average per tag (datalog and batch)	on
Chart on New Page / Statistics on New Page / Data Table on New Page	Start the section on a new PDF page	on / off / on
Custom Pages	Extra text pages: Page Title , content, position Before Data or After Data	none

The generated PDF is A4 portrait, created in the browser. It starts with the report name, "Generated: *date*" and "Date Range: *from – to*". For batch reports, each batch gets a summary block (batch name, start, end or

"Running...", user, logging type) followed by its chart, statistics and detail table.

Note: The preview shows the first 50 rows of the data table. The PDF contains all rows returned for the range; very long ranges produce very long PDFs. Use statistics or a coarser datalog interval for long periods.

21.3 Visual Report Builder

The Visual Report Builder is a page-layout editor. A *layout* is saved on the server and can be opened by administrators in the builder, and by operators through the **Show Visual Report** click action.

Config → **Reports** → **Visual Report Builder** lists the saved layouts with **Edit**, **Preview** and **Delete**. **+ New Visual Report** opens **Choose a Template**:

Template	Contents	Parameters
Blank Report	Empty canvas	—
Batch Report	Cover header (batch number, operator, start, end, duration, recipe, status), data table, trend chart, alarm table	batchNumber
Alarm Summary	Alarm history table (last 24 h, 200 rows), total alarm count field	—
Daily Production Log	Print date, data table, statistics, trend chart (last 24 h)	—
Shift Report	"Shift: {{param.shift}}", data table, alarm table, Operator Notes custom page	shift (default "Day")
Equipment Status	Data table with conditional formatting (>90 red, <10 blue)	—
SPC Report	"Tag: {{param.tagName}}", control chart, statistics, data table	tagName

Templates contain empty data sources: select the datalog and tags in every data element before use.

21.3.1 The editor

Area	Contents
Toolbar	← Back ; Portrait/Landscape ; A4/Letter ; Undo (Ctrl+Z), Redo (Ctrl+Y); align (L , C , R , T , M , B) and distribute (H , V) for multiple selection; Del , Dup , lock/unlock, Front , Back ; snap-to-grid with grid 5/10/15/20 mm; zoom – / +; Params ; + Batch Header ; Save ; Preview
Page tabs	One tab per page. Double-click to rename, arrows to reorder, × to delete, + Page to add
Sections	Page Header , Report Header (first page only), Report Footer (last page only), Page Footer . Tick a section to enable it, click Edit to draw its elements, set Height (mm) (10–100), Main / Back to Main to return
Element library	Categories below
Properties panel	Position & Size (X, Y, Width, Height in mm) plus type-specific properties

Leaving the builder with unsaved changes shows **Unsaved Changes** with **Discard Changes**, **Cancel**, **Save & Close**.

Element library:

Category	Elements
Text	Title, Subtitle, Paragraph, Label
Data	Data Table, Statistics, Alarm History
Charts	Line Chart, Bar Chart, Area Chart, Pie Chart
Fields	Special Field, Formula Field
Shapes	Rectangle, Line, Ellipse
Layout	Image / Logo, Divider, Page Break, Custom Page

21.3.2 Data elements

Data Table, Statistics and charts share the **Data Source** properties:

Property	Meaning
Data Mode	Date Range or Batch Number
Source Type (date range)	Datalog, Events, Alarms
Datalog / Tags	Datalog and the columns (tags) to show; All / None
Date Range	Last Hour, Last 24 Hours, Last Week, Last Month, Custom (Start, End)
Batch Datalog / Batch Number (batch mode)	The batch datalog and the batch. The batch number can be bound to a report parameter (Param → select a parameter; Unbind to remove)
Max Rows	Row limit for tables (default 50)

Additional properties:

Element	Properties
Data Table	Show Header , header colours, Striped Rows , font size; Grouping (Group By tag, Sort Asc/Dsc, header/footer, subtotals, + Add Group); Conditional Formatting rules (operator = , != , > , < , >= , <= , contains, is empty → text colour, background, bold; + Add Rule)
Statistics	Show Min, Max, Average, Count
Chart	Chart Type (Line, Bar, Area, Pie), Title , Show Legend , Show Grid , Manual Y-Axis Scale with Y Min , Y Max , Y Step (blank = automatic)
Alarm History	Alarm Filters : group, type (HH, H, L, LL, On, Off, Communication), severity (Critical, Warning, Info), area, status (Active, Cleared); Date Range including Batch (use batch time range) with Datalog and Batch # ; Max Rows (default 100); Visible Columns (Timestamp, Tag Name, Alarm Type, Value, Threshold, Status, Severity, Group, Area, Acknowledged); Table Style
Image / Logo	Upload Image (max 2MB) or Or enter URL ; Fit Contain/Cover/Fill
Special Field	Page Number , Total Pages , Print Date , Print Time , Current Date/Time , Report Name , Record Count
Formula Field	Expression , Decimals , Prefix , Suffix (Section 21.3.4)

Element	Properties
Custom Page	Page title and content text, sizes and colours

21.3.3 Chart legend

In line, bar and area charts, **Show Legend** draws the legend as its own row *below* the plot area: one coloured bar and the tag name per series, wrapping onto several lines when there are many tags. The plot shrinks to leave room for it, so the tag names are always visible in the preview and in the PDF, however small the chart box. (Before version 2.6.1.11 the legend was drawn inside the chart and could be cut off in small boxes.) Series colours follow a fixed palette of ten colours in tag order. Pie charts keep their built-in legend.

Tip: If a chart has many tags, make the chart box taller rather than wider: the legend needs a line per few tags, and the plot keeps the remaining height.

21.3.4 Dynamic text (tokens)

Text elements can contain tokens that are replaced when the report is previewed or printed. The **Insert Tag Value** helper in a text element's properties builds them: choose **Live (PLC)**, **From Batch** (then **Batch Datalog**) or **From Recipe** (then **Recipe (template)**), select **Tag** and **Attribute**, click **+ Insert Into Content**.

Token	Value
{{tag.NAME}}, {{tag.NAME.attr}}	Live tag value (with the tag's decimals); attr = value, unit, label, comment, min, max
{{batch.NAME}}, {{batch.NAME.attr}}	Aggregate of the tag over the batch rows; attr = last (default), first, min, max, avg, count
{{recipe.NAME}}, {{recipe.NAME.description}}, {{recipe._name}}	Setpoint from the recipe snapshot attached to the batch (Chapter 20.6), its description, the recipe name
{{batchInfo.X}}	Batch header: batchNumber, operator, startTime, endTime (" (running)" while open), startDate, startTimeOnly, endDate, endTimeOnly, duration (e.g. 2h 5m 10s), durationSec, recipeName, isRunning (Yes/No), status (RUNNING/COMPLETED)
{{param.X}}	Value of report parameter X

A token that cannot be resolved prints --. **+ Batch Header** inserts a ready-made cover block using the batchInfo tokens.

Formula Field expressions support SUM(tag), AVG(tag), MIN(tag), MAX(tag), COUNT(tag) over the data loaded by the report's data elements, combined with numbers, + - * / (). Example: AVG(Tank_101_Temp) * 1.8 + 32. An invalid expression prints #ERR.

21.3.5 Report parameters

Params opens **Report Parameters**. Each parameter has **Name**, **Label**, **Type** and **Default**. Types: **Text**, **Number**, **Date**, **Alarm Group**, **Alarm Type**, **Severity**, **Area**, **Status**, **Tag Name**, **Batch Number**. Parameters are asked for (or supplied by the synoptic button) when the report is shown, and are used in {{param.X}} tokens and in bound properties.

21.3.6 Preview and export

Preview renders all pages. For a report that uses batch data, **Select Batch for Preview** asks for the **Batch Datalog** and **Batch Number** first. The preview window has **Close**, **Export PDF** and **Export Excel**.

Export	How it is produced
Export PDF	Each report page is rendered as an image (JPEG) and placed on one PDF page of the chosen size and orientation, scaled to fit. Text in the PDF is therefore not selectable.
Export Excel	The server builds an <code>.xlsx</code> from the data tables, charts' data, statistics, alarm tables and texts; file <code><report name>_<date>.xlsx</code> .

21.4 Batch reports end to end

A batch report needs three things:

1. A **batch datalog** (Config → **Data Storage**, **Datalog Type** = **Batch**, Chapter 19) with: **Batch Logging Type** (**Interval-Based** or **Event-Driven**), **Batch Number Source** (**From Tag** or **Date & Time (Auto)** = `YYYYMMDD_HH:mm:ss`), **Batch Number Tag**, **Batch Start/Stop Tag** (TRUE = running), the logged tags, and an external SQL data source. It creates `<table>_header` (batch_number, username, start_datetime, end_datetime) and `<table>_data` (datetime, batch_number, tags).
2. Optionally the recipe snapshot (**Attach Recipe to Batch**, Chapter 20.6) and extra snapshots (**Log Tag Snapshot (Batch)**).
3. A report — a **Batch Report** template (21.2.1) or a visual report whose elements use **Batch Number** data mode.

To give the operator a batch report button:

1. In the Viewer Builder select the button, **Click Action** = **Show Visual Report**.
2. **Report Type: Batch**. Select the **Batch Datalog**.
3. If the datalog takes the batch number from a tag, select the **Batch Number Tag** ("Tag whose current value will be used as the batch number for the report"). For **Date & Time (Auto)** datalogs no tag is needed.
4. **Select Report Layout** and map the report parameters.
5. Save and deploy.

At runtime the report opens in a viewer with zoom (–, +, fit, 100 %), **Export PDF** and **Export Excel**.

21.5 Shift Handover Report

The header button **Shift Report** (add it in Layout Builder → Header or Config → **Display** → Navigation Buttons) opens the **Shift Handover Report**.

To produce a shift report:

1. Select the shift length: **4h Shift**, **8h Shift**, **12h Shift**, **24h** (the period ends now), or enter a **Custom range**: *from to* and click the custom-range button.
2. Click **Generate Report**.
3. Review the sections; click **Export PDF** to save `ShiftReport_<date>.pdf`.

Section	Contents
Shift Period	Start and end
Alarm Summary	Total Alarms, Unacknowledged , count by severity, Most Frequent Alarms (top 10 tags)
Event Summary	Total Events, Logins, Tag Writes, Config Changes
System Status	Total Tags, Good Quality, Bad Quality, Unknown , historian status
PLC Connections	Name, IP, Protocol of enabled connections
Recent Alarms (Last 20)	Time, Tag, Message, Ack

The PDF contains the alarm, top-alarm, event and system-status tables.

The report reads the local alarm history (the `alarms` table, Chapter 17) and the event log (`event_log` , Chapter 18) for the selected period. Local alarm rows store no severity or message text, so the report derives them:

Field	How it is derived
Severity	critical for alarm types <code>high_high</code> , <code>low_low</code> and communication alarms; warning for every other type.
Message	<code><alarm type> = <value> (limit <threshold>)</code> , for example <code>high = 87.5 (limit 85)</code> .
Logins	Number of <code>login_success</code> events (and SSO logins) in the period.
Tag Writes	Number of <code>tag_write</code> events in the period.

Note — fixed in 2.6.1.26: Up to 2.6.1.25 the shift report read a table (`alarm_history`) and event-log columns that no database contains, so **Generate Report** failed (HTTP 500) and no report was shown. Scheduled reports (21.6) and `GET /api/export/alarms` (Chapter 28) had the same fault and now work as well.

21.6 Scheduled (automatic) reports

The server contains a report scheduler that checks every 60 seconds for enabled scheduled reports and, at the configured time, generates a shift-summary report (the same content as Section 21.5, plus min/max/average for up to 30 historian-enabled numeric tags) as an HTML document and e-mails it as an attachment.

Important: There is no configuration screen for scheduled reports in version 2.6.1.26. They are managed through the REST API (Chapter 28), by an administrator. Scheduled reports cannot send Report Designer or Visual Report Builder layouts, and they do not produce PDF or Excel files. Scheduled reports use the same alarm and event data as Section 21.5 (fixed in 2.6.1.26; up to 2.6.1.25 they failed on a database error).

Field	Meaning	Default
<code>name</code>	Report name; used in the e-mail subject <code>[OptiZeus Report] <name> – <date></code> and in the file name	Shift Report
<code>type</code>	Only <code>shift</code> is implemented	shift
<code>schedule</code>	<code>daily</code> , <code>weekly</code> (Mondays), <code>monthly</code> (1st of the month)	daily

Field	Meaning	Default
time	HH:MM , server local time	06:00
shift_duration_hours	The report covers the last N hours	8
email_to	Comma-separated recipients	empty (no e-mail)
enabled	true/false	true

E-mail is sent through the e-mail settings of the alarm notification service (Chapter 17). If e-mail is not configured, the report is generated, `last_run` is updated and the server log shows "Email not configured". A report runs at most once per day.

Example — daily report at 06:00 covering the night shift:

```
POST /api/scheduled-reports HTTP/1.1
Authorization: Bearer <admin token>
Content-Type: application/json

{ "name": "Line 1 Night Shift", "type": "shift", "schedule": "daily",
  "time": "06:00", "shift_duration_hours": 8,
  "email_to": "shift.lead@example.com, production@example.com", "enabled": true }
```

Endpoint	Purpose
GET /api/scheduled-reports	List
POST /api/scheduled-reports , PUT /api/scheduled-reports/{id} , DELETE /api/scheduled-reports/{id}	Maintain (admin)
POST /api/scheduled-reports/{id}/run-now	Run immediately and e-mail (admin)
GET /api/scheduled-reports/{id}/preview	Report data as JSON
GET /api/scheduled-reports/{id}/preview-html	The HTML that would be e-mailed

Each run writes an event `REPORT` (or `REPORT_ERROR`).

21.7 Exporting and printing

From	PDF	Excel	CSV
Report Designer	Generate PDF	—	—
Visual Report Builder / Show Visual Report	Export PDF	Export Excel	—
Shift Handover Report	Export PDF	—	—
Batch Execution Report (Chapter 20)	Print / PDF (browser print)	—	Export CSV
Alarms, events, historian (Chapters 17–19)	Yes	—	Yes

All PDFs are generated in the browser, so they use the client's time zone and date format.

21.8 Worked example: a daily shift report

Goal: a one-page report for Line 1 that the shift leader prints at the end of each 8-hour shift, with a table and trend of Tank_101 and Pump_P201, statistics, alarms of the shift and an operator notes area.

Prerequisite: a standard datalog `Line1_Process` (Config → **Data Storage**) logging `Tank_101_Level`, `Tank_101_Temp`, `Pump_P201_Flow` and `Pump_P201_Run` every minute (Chapter 19).

1. **Config** → **Reports** → **Visual Report Builder** → + **New Visual Report** → **Shift Report**.
2. Rename the report "Line 1 Shift Report" (double-click the name).
3. Click **Params**: keep `shift` (Label "Shift", Type Text, Default "Day"); add `leader` (Label "Shift leader", Type Text). **Close**.
4. Select the title and change it to `Line 1 Shift Report`. Add a Label under it with `Shift: {{param.shift}}` – `Leader: {{param.leader}}`.
5. Select the data table. **Data Mode**: **Date Range**; **Source Type**: **Datalog**; **Datalog**: `Line1_Process`; tick the four tags; **Date Range**: **Custom** is not suitable for a recurring report — choose **Last 24 Hours** and reduce **Max Rows** to 50, or set **Last Hour** for a short extract. (For an exact 8-hour window use a named-query report, Section 21.2.1.)
6. Add a **Conditional Formatting** rule to the table: `value > 90` → red text, bold.
7. From **Charts** drag a **Line Chart** below the table: datalog `Line1_Process`, tags `Tank_101_Level` and `Pump_P201_Flow`, **Title** "Level and flow", **Show Legend** on. Make it about 60 mm high so the legend row has room.
8. Drag a **Statistics** element: same datalog and tags; Min, Max, Average.
9. Select the alarm table: **Date Range** **Last 24 Hours**, **Alarm Filters** Area = `Line 1`, Max Rows 30.
10. Enable **Page Footer**, click **Edit**, add a **Special Field Page Number** and a **Special Field Print Date**; click **Main**.
11. **Save**. Click **Preview**, check the data, then **Export PDF**.
12. For the operator: on the Line 1 overview screen add a button "Shift report", **Click Action** **Show Visual Report**, **Report Type** **Standard**, **Select Report Layout** "Line 1 Shift Report", map `shift` to a fixed value or tag. Save and deploy.

The operator presses the button at the end of the shift, types the parameters if asked, checks the preview and clicks **Export PDF** or prints it.

Tip: Automatic e-mailing of this layout is not available (Section 21.6). If an e-mailed summary is required, add a scheduled shift report in addition.

22. KPI, OEE and SPC dashboards

This chapter describes the performance-monitoring tools of OptiZeus: the **KPI Dashboard** (a free grid of widgets), **OEE & Downtime Tracking**, the three **SPC** screen components (control chart, histogram, capability analysis), and the eight operator dashboard components (gauges, displays, indicators) that can be placed on synoptic screens. It closes with a note on data logging. Engineers who set up the dashboards need the whole chapter; supervisors and operators need Sections 22.1.3, 22.2.4, 22.2.5 and 22.3.4.

Note: Your licence may list an "SPC / Statistical Process Control" add-on (Appendix C). In version 2.6.1.26 the SPC components described here are not blocked when the add-on is missing. Check your licence terms before using them in production.

22.1 KPI Dashboard

The KPI Dashboard is a full-page view with one or more dashboards, each a grid of resizable widgets bound to tags.

22.1.1 Opening and storage

Open it with the header button **KPI Dashboard** (add it in Layout Builder → Header or Config → **Display** → Navigation Buttons, Chapter 14). ← **Back** returns to the main dashboard.

Important: KPI dashboards are stored in the **browser** of the client (local storage key `kpi-dashboards`), not on the server. They are not shared between users of the same PC, are not copied to other client PCs, are not included in project backups (Chapter 26), and are lost when the browser data is cleared. Build the dashboard on each client PC that needs it, or use a synoptic screen with the components of Section 22.4 when the dashboard must be managed centrally.

22.1.2 Toolbar

Control	Function
Dashboard list	Switches between dashboards. The first dashboard is called "Main Dashboard".
Rename	Renames the current dashboard (OK / Cancel).
+ New Dashboard	Adds "Dashboard N".
Delete	Deletes the current dashboard (shown only when there is more than one; no confirmation).
Edit / Done Editing	Switches edit mode. In edit mode widgets can be dragged and resized, and each widget shows Edit and Delete .
+ Add Widget	(edit mode) Opens the widget dialog.

22.1.3 Adding a widget

To add a widget:

1. Click **Edit**, then **+ Add Widget**.
2. Click a **Widget Type** tile.

3. Select the **Tag** (the list shows the current value) and fill in the type-specific fields.
4. Set **Title**, **Subtitle**, **Decimal Places**, **Initial Size (1×1, 2×1, 2×2)** and **Color Theme** (Blue, Green, Red, Yellow, Purple, Cyan, Orange).
5. Save. Drag the edges of the widget to resize it, then click **Done Editing**.

The dialog has these general fields for value widgets: **Min**, **Max**, **Zones (JSON array: [{from, to, color}])**, **Warning Threshold**, **Alarm Threshold**, **Text Alignment**.

22.1.4 Widget types

Widget	Shows / does	Specific fields
Value Card	Current value with unit and quality	thresholds
Gauge	Round gauge	Min, Max, Zones
Sparkline	Small trend from the historian	Time Range (minutes): 15 min ... 8 hours
Status Indicator	On/off state	—
Bar Chart	Bar proportional to the value	Min, Max
Text Widget	Static text	text, alignment
Clock	Date and time	—
Alarm Count	Number of active alarms	—
PLC Status	Connection state of the PLC connections	—
Embed (iframe)	Another web page	Embed URL
Realtime Trend	Up to 8 tags as a live trend	Select Tags (up to 8), History Points, Interval (ms)
Current Alarms	List of active alarms	—
Linear Gauge	Horizontal/vertical bar gauge	Min, Max, Orientation
Tank Level	Tank graphic	Min, Max, Tank Color
LED Indicator	Coloured lamp	ON Color, OFF Color, ON Value
Write Button	Writes a fixed value after confirmation	Write Value, Button Label, Button Color
Toggle Switch	Writes true/false	ON Label, OFF Label
Slider Control	Writes a value from a slider	Min, Max, Step
Data Table	Several tags and their values	Tags (comma-separated)
Pie Chart	Share of several tags	Tags (comma-separated), Labels (comma-separated, optional)
XY Plot	One tag against another	X-Axis Tag, Y-Axis Tag, Max Points
Histogram	Distribution of the tag's recent values	Bins, Max Samples
Running Hours	Time the tag has been true	—
Setpoint Monitor	Process value against setpoint	PV Tag (Process Value), SP Tag (Setpoint), Min, Max

Widget	Shows / does	Specific fields
Circular Progress	Value as a ring	Min, Max, Arc Color
Image	Picture	Image URL, Object Fit

Widgets that plot "recent" values (Realtime Trend, XY Plot, Histogram, Running Hours) collect their samples in the browser while the dashboard is open; they start empty each time the page is opened. Running Hours is therefore a session counter, not a persistent hour meter — use a PLC counter tag with a Value Card for maintenance hours.

Write Button, **Toggle Switch** and **Slider Control** write through `POST /api/tags/<name>/write`, the same route as synoptic write objects: only `admin` and `operator` may write, security areas apply, and a write that does not reach the controller fails and leaves the old value (Chapter 16, Section 16.4.5). The **Write Button** asks for confirmation and shows *Success* or *Write Failed*. **Toggle Switch** and **Slider Control** show no message: the switch follows the live tag value, so a refused write shows as a switch that does not change; after moving a slider, check the displayed value. None of these widgets asks for a reason or login confirmation — use synoptic write objects (Chapter 16) where those are required.

Note — fixed in 2.6.1.25: Up to 2.6.1.24 the three write widgets sent their writes to a server address that does not exist, so no value reached the PLC, and the **Sparkline**, **Alarm Count** and **PLC Status** widgets requested their data without the login token and showed no data. The write widgets now write as described above, and the three display widgets send the login token and show their data.

22.2 OEE & Downtime Tracking

OEE & Downtime Tracking calculates Overall Equipment Effectiveness per machine from three tags and records downtime. Open it with the header button **OEE**. It has four tabs: **Dashboard**, **Equipment Config**, **Downtime Log**, **Reports**.

22.2.1 How OEE is calculated

Every 60 seconds the server takes a snapshot for each enabled machine:

Quantity	Calculation
Run minutes	Number of snapshots in the current shift in which the run tag was > 0 (one snapshot = one minute), plus the current minute
Planned minutes	Planned Production Hours/Shift × 60
Availability	run minutes ÷ planned minutes (max. 100 %)
Performance	(total parts × ideal cycle time) ÷ run seconds (max. 100 %)
Quality	good parts ÷ total parts (max. 100 %)
OEE	Availability × Performance × Quality

Shifts are fixed at **06:00–14:00**, **14:00–22:00** and **22:00–06:00** (server local time). The dashboard shows the *average* of the snapshots since the start of the current shift.

The part counters are used as absolute values: the counter tags must be reset by the PLC at the start of each shift (or be shift counters). A counter that keeps counting across shifts gives wrong Performance and Quality.

22.2.2 Configuring equipment

To add a machine:

1. **OEE → Equipment Config → Add Equipment.**
2. Fill in the fields and save.

Field	Meaning	Default
Equipment Name	Unique internal name, e.g. <code>Machine_01</code>	—
Display Label	Name on the dashboard, e.g. "Packaging Line 1"	—
Run Status Tag (1=Running, 0=Stopped)	Tag > 0 means running	—
Good Parts Counter Tag	Good parts in the shift	—
Total Parts Counter Tag	All parts in the shift	—
Ideal Cycle Time (seconds/part)	Design cycle time	60
Planned Production Hours/Shift	Planned time per shift	8
Enabled	Enabled / Disabled	Enabled

Delete removes the machine and all its OEE data (confirmation "Delete this equipment and all its OEE data?").

Creating, editing and deleting equipment requires the `admin` role.

22.2.3 Automatic downtime detection

When the run tag changes from running to stopped, the server opens a downtime event (category **Unplanned**, reason "Unclassified"). When it runs again, the event is closed and its duration calculated. The check runs with the 60-second snapshot, so stops shorter than a minute may not be recorded.

22.2.4 Dashboard tab (supervisor)

One card per machine: name, **Running/Stopped**, four values — **Availability**, **Performance**, **Quality**, **OEE** — with bars coloured green ($\geq 85\%$), amber ($\geq 60\%$) or red ($< 60\%$), **Run:** time, **Good:** ... / **Total:** ..., and the reason of an open downtime. The tab refreshes every 10 seconds.

22.2.5 Downtime Log tab (operator)

The table lists **Equipment**, **Start**, **End**, **Duration**, **Category**, **Reason**, **Description**, **Actions**, filtered by **All Equipment** or one machine. Operators classify automatic events by selecting **Category** and **Reason** in the row.

To record a downtime manually:

1. Click **Record Downtime**.
2. Select **Equipment**, **Category (Planned / Unplanned)**, **Reason**, enter **Description**, **Start Time** and **End Time (leave empty if ongoing)**.
3. Click **Save**.

Category	Reasons
Planned	Scheduled Maintenance, Changeover, Break/Meal, Planned Shutdown, Cleaning
Unplanned	Equipment Failure, Material Shortage, Operator Unavailable, Quality Issue, Power Failure, Other

Deleting a downtime event requires the `admin` role.

Note: Downtime classification is recorded for analysis; it does not change the Availability calculation, which is based only on the run tag.

22.2.6 Reports tab

Select the machine and the period (**Last 1 Hour**, **Last 8 Hours**, **Last 24 Hours**, **Last 7 Days**, **Last 30 Days**). The tab shows **Average OEE** with the number of data points and a table of snapshots: **Timestamp**, **Availability %**, **Performance %**, **Quality %**, **OEE %**, **Good Parts**, **Total Parts**. For printed OEE reports, log the OEE tags or use the REST API (`GET /api/oeo/metrics` , `GET /api/oeo/downtime-summary`) with a named query report (Chapter 21).

22.3 SPC components

The three SPC components are placed on synoptic screens from the Viewer Builder component library (Chapter 11): **SPC Control Chart**, **SPC Histogram**, **Capability Analysis**.

Important: SPC components calculate from the tag values the *browser* has received since the screen was opened, one sample per value change, at most **60 samples per tag**. The **Data Points** setting (default 50 or 100) cannot exceed this. They do not read the historian and do not use subgroups. A tag whose value does not change produces no new samples. Use them for a live indication of process stability; for formal capability studies export historian data (Chapter 19) and analyse it with a statistics tool.

22.3.1 SPC Control Chart

An individuals chart with centre line, control limits and run rules.

Property	Meaning	Default
Title	Header	Control Chart
Tag (Data Source)	Monitored tag	—
Data Points	Samples shown	50
Unit	Engineering unit, e.g. kg, psi, degC	from the tag
Show limits / warning limits / mean / statistics	Display options	on
UCL Override, LCL Override, Target Override	Fixed limits instead of calculated ones	Auto-calculated

Calculation: mean and sample standard deviation ($n-1$) of the displayed points; $UCL/LCL = \text{mean} \pm 3\sigma$; warning limits = $\text{mean} \pm 2\sigma$. Points beyond the control limits are drawn red and counted in a red badge; points between warning and control limits are amber. Rules evaluated:

Rule	Condition
1	One point beyond 3σ
2	9 consecutive points on the same side of the mean
3	6 consecutive points steadily increasing or decreasing

The footer shows **Mean**, **Std Dev**, **Count** and violations.

22.3.2 SPC Histogram

Property	Meaning	Default
Title, Tag (Data Source), Unit	As above	Histogram
Data Points	Samples	100 (max. 60 available)
Number of Bins	Histogram classes	10
Show normal curve / statistics / spec limits	Display options	on / on / off
USL (Upper Spec Limit), LSL (Lower Spec Limit), Target	Specification lines	—

22.3.3 Capability Analysis

Property	Meaning	Default
Title, Tag (Data Source), Unit, Data Points	As above	Capability Analysis, 100
USL (Upper Spec Limit), LSL (Lower Spec Limit)	Required ; USL must be greater than LSL	100, 0
Target (Optional)		midpoint
Display options	gauges, distribution, statistics	on

Indices:

Index	Formula
Cp	$(USL - LSL) \div 6\sigma$
Cpk	$\min((USL - \text{mean}) \div 3\sigma, (\text{mean} - LSL) \div 3\sigma)$
Pp, Ppk	Shown, but equal to Cp and Cpk in this version (one σ is used for both)
PPM	Expected parts per million outside the limits, from the normal distribution
% within spec	Measured share of samples between LSL and USL

Gauge marks are at 1.00, 1.33 and 1.67. With invalid limits or no data the component shows "No data or invalid spec limits".

22.3.4 Reading an SPC chart (operator)

- A red point or the red "out of control" badge means the last values are outside $\pm 3\sigma$ of the recent average: check the process and inform the supervisor.
- A rule 2 or rule 3 violation means a shift or trend: the process is drifting even if no alarm is active.

- The limits move with the data. After a process change, the chart needs a full window of new samples before the limits are meaningful again. Fixed limits (overrides) avoid this.

22.4 Operator dashboard components

These eight components (Viewer Builder component library, Chapter 11) display one tag each on a synoptic screen and are stored with the screen on the server. The KPI Dashboard reuses the analog gauge.

Component	Key properties (defaults)
OpAnalogGauge	Title, tag, unit, Min 0, Max 100, decimals 1, show value/ticks/labels, low/high thresholds, colour zones
OpLinearGauge	Min, Max, orientation horizontal/vertical, thresholds L, LL, H, HH, show scale and thresholds
OpDigitalDisplay	Decimals 2, 6 digits, show sign, blink on alarm
OpStatusIndicator	Label, shape (circle), size, ON colour green, OFF colour grey, blink when on, label position
OpBulletGraph	Min, Max, target, subtitle, orientation
OpSparkline	30 data points, show value, show min/max, thresholds, fill area
OpTankLevel	Unit %, Min 0, Max 100, scale, thresholds, vertical style, liquid colour, waves
OpMultiState	List of states (value, label, colour, icon), e.g. 0 = Off grey, 1 = On green

OpSparkline uses the same browser-side sample buffer as the SPC components (max. 60 samples).

22.5 Data logging for dashboards and reports

Dashboards show live values; reports, batch records and OEE analysis need logged data. Data logs are configured in **Config** → **Data Storage** (data sources, standard/batch/alarm-and-event datalogs, historian) — see Chapter 19. The source tree also contains an older stand-alone datalog page (`DataLogView`) that is not reachable from the menus in this version; use **Data Storage**.

22.6 Worked example: OEE for a filling machine

Machine: filler Line 1, PLC tags `Filler_L1_Running` (BOOL), `Filler_L1_Good` (DINT, reset by the PLC at 06:00, 14:00, 22:00), `Filler_L1_Total` (DINT, same reset). Nominal speed 40 bottles/min → ideal cycle time 1.5 s. Planned time 7.5 h per shift (30 min break).

1. **OEE** → **Equipment Config** → **Add Equipment**: Equipment Name `Filler_L1`, Display Label "Filler Line 1", Run Status Tag `Filler_L1_Running`, Good Parts `Filler_L1_Good`, Total Parts `Filler_L1_Total`, Ideal Cycle Time `1.5`, Planned Hours `7.5`, Enabled.
2. After two hours of the morning shift the machine has run 110 minutes and made 4 200 bottles, 4 116 good. The snapshot taken at that minute contains:
 - Availability = $110 \div 450 = 24.4\%$ (it rises as the shift progresses, because planned time is the whole shift; the dashboard shows the average of all snapshots of the shift, which is lower still)
 - Performance = $4\,200 \times 1.5\text{ s} \div 6\,600\text{ s} = 95.5\%$
 - Quality = $4\,116 \div 4\,200 = 98.0\%$

3. Operators open **Downtime Log**, classify the automatic "Unclassified" events (for example **Planned** → **Changeover**).
4. At the end of the week the supervisor opens **Reports**, selects **Last 7 Days** and reads the average OEE.

Tip: Because Availability is measured against the whole planned shift, the dashboard value during the shift is a running figure. Compare shifts using the value at the end of the shift or the **Reports** tab.

23. Operator tools: procedures, notepad, voice, AI assistant and anomaly detection

This chapter describes the tools that support operators and engineers in daily work: **Procedure Automation** (electronic standard operating procedures), the **Operator Notepad**, **Voice Commands**, the **AI Assistant**, **AI Anomaly Detection**, and the **Guided Tours**. Operators need Sections 23.1.4, 23.2, 23.3 and 23.6; supervisors and engineers need the whole chapter.

All views in this chapter except the AI Assistant and the anomaly-detection settings are opened from header buttons. Add the buttons **Procedures**, **Notepad** and **Voice** in Layout Builder → Header or Config → **Display** → Navigation Buttons (Chapter 14). Each view has a **Back** button that returns to the main dashboard.

23.1 Procedure Automation (electronic SOPs)

Procedure Automation stores step-by-step procedures (start-up, cleaning, changeover, lock-out...) and guides an operator through them, recording who completed each step and when.

23.1.1 Tabs and permissions

Tab	Contents
Procedure Library	All procedures as cards: name, category, status (draft, approved, archived), description, step count, "Approved by ..."
Active Execution	The procedure currently being executed on this client
Execution History	The last 100 executions: #, Procedure , Started By , Started , Completed , Status , Actions (Resume for running executions, View otherwise)

Operation	Roles
Create, edit, delete, approve a procedure	admin
Start an execution, complete a step, abort an execution	admin , operator
Skip a step	admin
View procedures and history	Any authenticated user

23.1.2 Creating a procedure

To create a procedure:

1. **Procedures** → + **New Procedure**.
2. Enter **Name** * (required), **Category** (free text, suggestions from existing categories; default "General") and **Description**.
3. Under **Steps (N)** click + **Add Step** for each step and fill in:

Field	Meaning
Step title	Short text, e.g. "Close drain valve V-102"
Instruction	Detailed instruction for the operator
Type	Manual, Verify Tag, Auto Write, Wait/Timer
Tag Name	Optional tag related to the step
Expected Value	Optional expected value, shown to the operator

4. Use ↑ / ↓ to reorder steps and **X** to delete a step.

5. Click **Create** (**Update** when editing). A new procedure has status **draft**.

Important: In version 2.6.1.26 the step type is descriptive. None of the types writes to the PLC, checks a value automatically, or runs a timer: **Auto Write** steps are only marked "Auto", and for steps with a tag the current value is *recorded* when the operator completes the step. The operator must perform and verify every action. Likewise, an **E-Sign** badge on a step (possible for procedures created through the REST API) is not enforced.

23.1.3 Approving a procedure

A procedure must be approved before it can be executed. On a draft card, click **Approve**. The card shows "Approved by *user*" and the status changes to approved; an audit entry `procedure_approved` is written. Approval does not ask for an electronic signature.

Note: Editing an approved procedure keeps it approved. If your quality system requires re-approval after changes, create a new procedure (or set the status back to draft through the REST API) and approve it again.

23.1.4 Executing a procedure (operator)

To run a procedure:

1. **Procedures** → **Procedure Library**.
2. On an approved card click **Start Execution**. The **Active Execution** tab opens with the first step active.
3. Read the step. Perform the action on the plant.
4. Optionally type a remark in **Step notes (optional)**...
5. Click **Complete Step**. The step turns green with a check mark, your user name and the time are recorded, and the next step becomes active. If the step has a tag, its current value is stored with the step.
6. Repeat until the last step. The execution is then marked completed and an event `procedure_completed` is logged.

Other actions:

Button	Effect
Skip	Asks "Reason for skipping:"; the step is marked skipped (grey dash) with the reason. Administrators only.
Abort	Asks "Reason for aborting:"; the execution ends with status aborted and an event <code>procedure_aborted</code> (warning).

The progress bar shows *completed* / *total* steps. An execution interrupted (browser closed) stays **running** and can be continued with **Resume** in **Execution History**.

WARNING: The procedure view records the operator's confirmation; it does not verify that the plant is in the stated condition. Keep interlocks and permits-to-work in place.

23.1.5 Worked example: pump changeover

Procedure "Changeover Pump_P201 → Pump_P202", category Operations:

#	Title	Type	Tag	Expected
1	Check standby pump P202 available	Verify Tag	Pump_P202_Ready	1
2	Start P202 from its faceplate	Manual	Pump_P202_Run	1
3	Confirm discharge pressure stable	Verify Tag	Header_PT301	4.0 bar ±0.2
4	Stop P201 from its faceplate	Manual	Pump_P201_Run	0
5	Wait 5 minutes, check flow	Wait/Timer	Pump_P202_Flow	> 30 m³/h

After approval, the operator starts it, performs each step on the pump faceplates (Chapter 16), and clicks **Complete Step**. The execution record shows the recorded value of each tag at the moment of confirmation — for example that Header_PT301 read 4.05 at step 3.

23.2 Operator Notepad

The **Operator Notepad** holds shift notes, handover notes, to-do items and sticky notes shared by all users of the system.

Tabs: **All Notes**, **Shift Notes**, **Sticky Notes**, **Resolved**. The first three show unresolved notes only. The **Search notes...** box filters by title and content. Pinned notes are listed first.

To write a note:

1. **Notepad** → **+ New Note**.
2. Fill in the fields and click **Create** (**Update** when editing).

Field	Values	Default
Title *	Required	—
Content	Free text	—
Type	Shift Note, Sticky Note, Handover, To-Do	Shift Note
Priority	High, Normal, Low	Normal
Shift	Free text, e.g. Morning, Night, A, B	—
Color	Card colour	yellow

The author and creation time are recorded automatically. Card buttons: **Edit**, **Pin/Unpin**, **Resolve** (moves the note to **Resolved** with "Resolved by *user* at *time*"), **Delete** (confirmation "Delete this note?").

Note: Any logged-in user can edit, resolve or delete any note. Notes are operational information, not controlled records; creation is logged as an event (`note_created`), but edits and deletions are not. Use the audit-trailed functions (Chapter 18) for records that must be kept.

Shift handover practice: at the end of the shift the outgoing operator creates a **Handover** note (Shift = the shift name) listing open issues, and resolves the notes that are finished. The incoming operator reads **All Notes**, then generates the Shift Handover Report (Chapter 21).

23.3 Voice Commands

Voice Commands lets an operator ask for values and open views by speaking or typing. The browser converts speech to text; the server interprets the text; the answer is shown in the **Conversation History** and read aloud.

23.3.1 Requirements

- A browser with the Web Speech API (for example a Chromium-based browser). If it is not available, the page shows "Speech recognition not supported in this browser" and you can still type commands.
- A microphone and permission to use it. Browsers normally allow the microphone only on `https://` pages or on `localhost` (Chapter 25 describes HTTPS).
- Recognition language is fixed to US English (`en-US`) in this version.

23.3.2 Using it

1. Open **Voice**.
2. Click the microphone ("Click to start listening"). When it shows "Listening... Speak now", say the command.
3. Or type it in the box "Or type a command... (e.g., 'What is Tank1_Level?')" and press Enter, or click one of the example buttons.
4. Read or listen to the answer. **Clear** empties the history.

23.3.3 Commands

Say or type	Result
"What is X", "What's X", "Value of X", "Read X", "Show X"	Value, unit and quality of the first tag whose name contains X (or is contained in X)
"How many alarms", "alarm count", "active alarms"	Number of active unacknowledged alarms and how many are critical
"System status", "How is the system", "overview"	Tag count, good-quality tags, active alarms, server uptime
"Acknowledge all alarms"	Answer "please use the Alarm Panel" and opens the alarm view — voice does not acknowledge
"Open / go to / navigate to view"	Opens: alarm(s), trend(s), config, historian, help, dashboard, kpi, batch, oee, notepad, soe, procedure(s)

Anything else returns "I didn't understand that command...".

Note: Voice commands only read values and navigate; they never write to the PLC. Tag matching is by name text, not by description: speak the tag name as it is configured (for example "value of Tank_101_Level"; the underscore is not spoken, so prefer typing for such names).

23.4 AI Assistant

The **AI Assistant** is a chat for engineers that helps create configuration (tags, alarms, screens, UDTs, super objects, datalogs) and documentation (GAMP 5 validation documents) using a large language model.

23.4.1 Where it is

Location	Use
Config → AI Assistant tab	Full chat and the provider settings
 ► button in the Config tab bar	Opens the chat as a side panel next to any other Config tab ( ◀ hides it)

The licence may include an "AI Assistant" add-on (Appendix C); in version 2.6.1.26 the assistant is not blocked when the add-on is missing.

23.4.2 Setting up a provider (administrator)

1. **Config → AI Assistant**, open the settings section **Multi-Provider AI Configuration**.
2. Tick **Enable AI Assistant**.
3. Select the **Active Provider**.
4. Under **Provider API Keys**, enter the key of the provider and select the model. Keys are masked after saving (only the last four characters are shown).
5. Click **Save Settings**.

Provider	Runs	Needs
Anthropic Claude, OpenAI, Google Gemini, Mistral AI, Groq, DeepSeek, Together AI	Cloud service of the provider	Internet access from the server and an API key from the provider
Ollama (Local - FREE)	On the OptiZeus server or another PC in the network (default localhost:11434)	Ollama installed and a model pulled (e.g. <code>ollama pull llama3.2</code>); no key. The settings show the Ollama status, installed models, recommended models and can pull a model.

WARNING: With a cloud provider, the questions and — if the context options are ticked — the names and descriptions of tags, screens, UDTs and alarms are sent over the internet to that provider. Check that this is allowed by your site's security policy (IEC 62443, customer contracts) before enabling a cloud provider. On isolated OT networks use Ollama. The API keys are stored in plain text in `ai-settings.json` in the project folder on the server; protect the server accordingly (Chapter 25).

23.4.3 Using the chat

1. Type a request, or click a quick action: **Create Tags, Create Alarm, Design Screen, Create UDT, Super Object, Datalog Config, GAMP5 Docs, From Screenshot** (upload a screenshot of an existing HMI screen; the assistant detects the objects).
2. Choose the context sent with the request: **Include Tags, Include Screens, Include UDTs, Include Alarms** (up to 100 existing tags are sent).
3. Read the answer. When it contains a configuration, the button ✓ **Apply Configuration** appears.
4. Review the proposal carefully, then click **Apply Configuration** to create the objects through the normal configuration functions (subject to your permissions and, if enabled, staging — Chapter 5).

The chat history is kept per browser. The trash button clears it.

Important: The AI Assistant proposes; it does not know your process. Every generated tag address, alarm limit and screen must be checked by a qualified engineer before deployment. Generated GAMP 5 documents are drafts that must be reviewed and approved under your quality system (Chapter 31).

23.5 AI Anomaly Detection

AI Anomaly Detection watches numeric tags continuously and raises an alarm when a value behaves unusually compared with its own recent history. It is statistical (no cloud service, no model training outside OptiZeus) and runs inside the OptiZeus server.

23.5.1 What it detects

Check	Condition
Z-score	The value deviates from the rolling mean by more than the threshold × standard deviation (default 3.0). It clears when the deviation falls below threshold – 0.5.
Rate of change	The change between samples is more than 3 × its usual variation (or a per-tag threshold).
Flatline	An analog value has not changed for the flatline time (default 300 s; 0 = off).

How it works:

- Every numeric, good-quality tag is analysed once every 10 seconds; boolean and string tags are ignored. Analysis starts after 30 samples (about 5 minutes).
- The rolling window holds up to 8 640 samples (24 hours at 10 s).
- Severity escalates with consecutive detections: **info**, **warning** from 3, **critical** from 10.
- After an anomaly, the same tag and check is suppressed for 10 minutes.
- Each anomaly writes an event `anomaly_detected`, a row in the anomaly history, and an **active alarm** of type `anomaly_zscore`, `anomaly_roc` or `anomaly_flatline` (source AnomalyDetection) that appears in the alarm list (Chapter 17).

23.5.2 Configuration (administrator)

Config → **Admin** → **AI Anomaly** opens **AI Anomaly Detection**. The badge shows ENABLED or DISABLED.

Field	Meaning	Default
Z-Score Threshold	Higher = less sensitive (3.0 $\approx$ 99.7 % confidence)	3.0
Flatline Timeout (sec)	0 = disabled	300
Learning Window (hours)	Hours of historian data used to learn normal behaviour	24

Buttons: **Enable & Save**, **Disable**, **Re-Learn Baselines** (reads historian data for the learning window to fill the rolling statistics immediately instead of waiting), **Refresh Status** (**Tracked Tags**, **Total Checks**, **Anomalies Found**, **Active Now**) and **Load History** (last 20 anomalies: Time, Tag, Type, Value, Z-Score, Severity).

Per-tag settings (enable/disable a tag, own thresholds, flatline time, suppression) and bulk enable/disable are available through the REST API (`/api/anomaly-detection/tags` , Chapter 28). Without a per-tag setting, every numeric tag is monitored with the global settings.

Tip: On a large project, enable detection first with a high threshold (for example 4.0), watch **Load History** for a day, then disable detection for tags that are noisy by nature (for example counters and setpoints) through the API before lowering the threshold.

Note: Flatline detection will flag any analog value that is legitimately constant (setpoints, stopped equipment). Set **Flatline Timeout** to 0 if this creates nuisance alarms.

23.5.3 The `anomaly_detector` folder is not part of the product

The repository and some installations contain a top-level folder `anomaly_detector` with Python programs (`main.py` , `dashboard.py` , `data_generator.py` , `ollama_explainer.py`). This is a separate stand-alone demonstration: it generates **simulated** process data, detects anomalies with Python libraries (pandas, scikit-learn), shows them in a web dashboard and can ask a local Ollama model for an explanation. It is not started by OptiZeus, does not read OptiZeus tags, and is not supported as part of the product. The anomaly detection described in Sections 23.5.1–23.5.2 is the built-in server function and needs no Python.

23.6 Guided Tours

Guided tours explain the user interface step by step with a spotlight on each element.

To start a tour:

1. Click **Help** in the header.
2. Open the section **Guided Tours**.
3. Click a tour card. OptiZeus opens the view the tour needs (for configuration tours, the right Config tab).
4. Use **Next** and **Back** to move through the steps. Click the dark overlay or **x** to close the tour.

Available tours:

Tour	Tour
Dashboard Overview	Data Storage & Historian
Tags & Monitoring	Reports & Visual Builder
User Defined Types (UDT)	Display & Layout Settings

Tour	Tour
Synoptic Screens (HMI)	AI Assistant
Super Objects	SVG Library
Alarms & Notifications	Data Import (from other HMI/SCADA projects)
Recipe Manager	Floor Display Mode
Process Viewer	User Management
PLC Communication	Notification Channels
Trends & History	Synoptic Builder Advanced
Batch Control (ISA-88)	Keyboard Shortcuts
Security & CFR 21 Part 11	Project Management
Staging & Deployment	OPC-UA Server

Tours for configuration views are useful only to users who can open the Config view (administrators).

23.7 REST API summary

Method and path	Purpose
GET/POST /api/procedures , GET/PUT/DELETE /api/procedures/{id} , POST /api/procedures/{id}/approve , GET /api/procedures/categories	Procedures
GET/POST /api/procedure-executions , GET /api/procedure-executions/{id} , POST .../{id}/complete-step , .../skip-step , .../abort	Executions
GET/POST /api/operator-notes , PUT/DELETE /api/operator-notes/{id}	Notepad (filters: type , shift , resolved)
POST /api/voice-command ({ "command": "..." })	Interpret a text command
GET /api/ai/providers , GET/POST /api/ai/settings , POST /api/ai/chat , GET /api/ai/ollama/status	AI Assistant
GET/PUT /api/anomaly-detection/config , GET /api/anomaly-detection/tags , PUT /api/anomaly-detection/tags/{tag} , POST /api/anomaly-detection/tags/bulk , GET /api/anomaly-detection/status , GET /api/anomaly-detection/history , POST /api/anomaly-detection/learn , POST /api/anomaly-detection/reset/{tag}	Anomaly detection

Example — exclude a counter from anomaly detection:

```
PUT /api/anomaly-detection/tags/Filler_L1_Total HTTP/1.1
Authorization: Bearer <admin token>
Content-Type: application/json

{ "enabled": false }
```

OPTIZEUS SCADA · OPERATION MANUAL

Part VI — Administration

24. Users, roles, MFA and directory integration

This chapter explains how OptiZeus identifies people: user accounts, the three built-in roles and exactly what each role may do, security letters, password rules, lockout, sessions and time-outs, two-factor authentication (TOTP), single sign-on through OpenID Connect, Active Directory / LDAP, and how an administrator forcibly disconnects a session. It is written for the administrator who sets up and maintains access to the system. Supervisors should read Section 24.3 to understand what their operators can and cannot do; operators need only Section 24.8 (logging in with a code) and Section 24.6.4 (changing a password).

All settings described here are in **Config** → **Admin**. The **Config** button in the header is shown to administrators.

24.1 Concepts

Term	Meaning
User	A named account (row in the <code>users</code> table of the active project database). Every login, write, acknowledgment and signature is attributed to a user.
Role	One of <code>admin</code> , <code>operator</code> , <code>viewer</code> . The role is carried inside the login token and is what the server checks before every protected request.
Security letters (A–Z)	Twenty-six flags per role. Screen objects and security areas can require a letter; a user whose role lacks it cannot use that object or area.
Security area	A rule that combines <i>who</i> (roles / letters) with <i>where</i> (client IP address) to protect groups of tags or screens (Section 24.4).
Session	A login. The server keeps one row per login in <code>user_sessions</code> and ends it after a period of inactivity.
Authentication type	<code>local</code> (password checked by OptiZeus), <code>ad</code> (password checked by a domain controller over LDAP) or <code>oidc</code> (user signs in at an external identity provider).

Note: Users, sessions and the password history are stored **per project**. Each project in `backend\projects\<name>\` has its own `scada.db`, so a user created in one project does not exist in another. Roles (`roles.json`), security settings (`security-settings.json`), security areas and the OIDC configuration are also files in the project folder. See Chapter 5 and Section 26.7.

24.2 The default accounts

When a project database is created and its `users` table is empty, OptiZeus creates three accounts: the built-in administrator account `admin`, one operator account and one viewer account. Their initial passwords are set in the product and are the same on every installation.

WARNING: Treat the default accounts as publicly known. Immediately after installation, and again after creating every new project: log in as the default administrator, set a new strong password for `admin` (Section 24.5.3), and either change the passwords of the default operator and viewer accounts or delete them.

Since version 2.6.1.24, when one of the factory accounts (`admin` , `operator` or `viewer`) logs in with its factory default password, the login screen opens the dialog **Change the Default Password** (*This account still uses its factory default password, which is published. You must set your own password before continuing.*) with **Current Password**, **New Password** and **Confirm New Password**. The dialog has no **Cancel** button; the session starts only after a successful change. Each such login is logged as the event `default_password_login` (category `security`). This does not apply on a server running in demo mode.

Important: The forced change is made by the login screen. A script that logs in to the REST API with a factory password still receives a session (the answer only flags the password as expired), and a factory account that is never used keeps its default password. Changing or deleting the default accounts therefore remains a commissioning step — put it in your installation checklist (the IQ protocol in Chapter 31 includes it).

The built-in `admin` account is protected by the server:

Protection	Server response if attempted
Cannot be deleted	Cannot delete admin user
Role cannot be changed from <code>admin</code>	Cannot change admin user role
Cannot be disabled	Cannot disable admin user
Never disabled for inactivity	(the daily inactivity check skips <code>admin</code>)

In addition, no administrator can delete his own account (`Cannot delete your own account`).

24.3 Roles and permissions

24.3.1 The three built-in roles

Role ID	Name	Description (as stored)	Default security letters	Kiosk mode
<code>admin</code>	Administrator	Full system access	A–Z all enabled	Off
<code>operator</code>	Operator	Can operate and acknowledge alarms	none	Off
<code>viewer</code>	Viewer	View only access	none	Off

These three roles are *system roles*: they cannot be deleted, and only their **security letters** and **Kiosk Mode** can be edited.

24.3.2 What each role may do

The table below is taken from the server's authorization checks in version 2.6.1.26. "Any" means any logged-in user, including a viewer.

Function	admin	operator	viewer
Log in, view screens, tag values, alarms, events, trends	Yes	Yes	Yes
Acknowledge, shelve, unshelve an alarm (REST API and live connection)	Yes	Yes	No
Group acknowledge / group shelve (alarm grouping)	Yes	Yes	No

Function	admin	operator	viewer
Write a tag value (POST /api/tags/:name/write)	Yes	Yes	No
Equipment control command over the live connection	Yes	Yes	No
Change alarm thresholds / sensor configuration over the live connection	Yes	No	No
Create, edit, delete synoptic screens and screen folders	Yes	Yes	No
Add, edit, import SVG library symbols	Yes	Yes	No
Create and edit schedules	Yes	Yes	No
Move tags between folders	Yes	Yes	No
Record and edit downtime events	Yes	Yes	No
Start a procedure, complete a step, abort a procedure	Yes	Yes	No
Start, pause, hold, resume, restart, stop a batch; confirm a transition	Yes	Yes	No
Download a recipe to the PLC (enabled recipes; the recipe's security letter is required when it has one)	Yes	Yes	No
Create, edit, delete, upload recipes	Yes	No	No
Connect / disconnect a PLC connection	Yes	Yes	No
Datalog batch actions: attach recipe, take snapshot, close batch	Yes	Yes	No
Create, edit, delete tags (POST/PUT/DELETE /api/tags... , PUT /api/tags/:name/historian)	Yes	No	No
Create, edit, delete datalogs	Yes	No	No
Historian configuration, purge, bulk enable, delete archive files	Yes	No	No
PLC connections (create, edit, delete, Test, Refresh Tags), alarm configuration, OPC UA server, reports configuration	Yes	No	No
Deactivate the licence	Yes	No	No
Users, roles, security settings, security areas, identity providers	Yes	No	No
Backup create / restore / download / delete; database maintenance	Yes	No	No
Audit trail, audit export, integrity check, e-signature history	Yes	No	No
Multi-server deployment, redundancy, multi-site, webhooks, ERP, Sparkplug, auto-update	Yes	No	No
Connected clients, force disconnect, service restart/stop, server shutdown	Yes	No	No
List backups and webhooks, read store-and-forward statistics	Yes	Yes	Yes

Note — changed in 2.6.1.26: Recipe **download to the PLC** (POST /api/recipes/:id/download) was administrator-only up to 2.6.1.25. It is now allowed for operator and admin ; a disabled (draft) recipe is refused, and when the recipe has a **Security Role** letter a non-admin user must hold that letter in their role (Chapter 20, Section 20.2.2).

Note — changed in 2.6.1.24: Earlier versions checked only for a valid login on tag create/edit/delete, datalog create/edit/delete, historian configuration and PLC connect/disconnect/test/refresh. From 2.6.1.24 the server enforces the roles in the table above for these requests, whether they come from the user interface or from a script; a refused request is answered *Insufficient privileges* (HTTP 403). An integration that creates tags or datalogs through the REST API (Chapter 28) must log in with an administrator account.

24.3.3 Custom roles

In **Config** → **Admin** → **Security** →  **General**, section **Role Management & Security Letters (A-Z)**, you can add roles in addition to the three system roles.

Field	Meaning
Role ID	Lower-case identifier. Characters other than <code>a-z</code> , <code>0-9</code> and <code>_</code> are replaced by <code>_</code> . Must be unique.
Role Name	Display name.
Description	Free text.
Security Letters (A-Z)	Click letters to enable them for the role.
Kiosk Mode	When enabled, users of this role run in locked full-screen kiosk mode.

The role editor also offers a list of permissions (**View**, **Operate**, **Acknowledge Alarms**, **Configure Tags**, **Configure Screens**, **Configure PLC**, **Manage Users**, **Manage Roles**, **Manage Groups**, **View Events**, **View Trends**, **Export Data**, **Full Access**).

Important — limitation in 2.6.1.26: Custom roles are stored and can be used in **security areas** (Section 24.4), but they cannot be assigned to a user. The user API accepts only `admin`, `operator` and `viewer` and answers `Invalid role` for anything else, even though the **Role** drop-down in the user form lists custom roles. The permission check-boxes of a role are saved but are not evaluated by the server; access is decided by the role ID (Section 24.3.2), security letters and security areas.

24.3.4 Security letters

Security letters let you grant access to individual screen objects without new roles. For example, give the operator role letter **M** and set "security letter M" on the maintenance-mode buttons of a screen: operators see and use them, viewers do not. Letters are set on objects in the Viewer Builder (Chapter 12) and on navigation buttons (Chapter 14). The administrator role always has all letters.

To change the letters of a role:

1. Open **Config** → **Admin** → **Security** →  **General**.
2. In **Role Management & Security Letters (A-Z)**, click **Edit** on the role.
3. Click the letters to toggle them. Enabled letters are shown in green.
4. Click the save button of the role form, then **Save Security Settings** at the bottom of the page.

Users receive their letters at login, so a change takes effect at the user's next login.

24.4 Security areas

A security area restricts **tag writes** and/or **screen access** to users that satisfy all of the area's rules. Open **Config** → **Admin** → **Security** →  **Security Areas**. The page shows **Your client IP** so you can test IP rules.

Field	Meaning	Default
Name *	Area name, e.g. <code>Control Room</code> .	—
Description	Optional.	empty
Allowed Roles (leave empty = any)	Roles admitted to the area.	any
Required Security Letter(s)	The user's role must have at least one of the selected letters.	none
Allowed IPs / CIDRs / Wildcards (one per line)	Client addresses admitted: exact (<code>10.0.0.5</code>), CIDR (<code>192.168.1.0/24</code>), wildcard (<code>172.16.*.*</code>), or <code>*</code> / any .	any
Tag Patterns (glob — applies when Enforce Writes)	Tag names protected, <code>*</code> as wildcard, e.g. <code>Setpoint_*</code> .	none
Screen Patterns (glob — applies when Enforce Screen Access)	Screen names protected, e.g. <code>Maintenance_*</code> .	none
Enabled	Area is evaluated.	on
Enforce Writes	Block writes to matching tags unless all rules pass.	off
Enforce Screen Access	Block opening matching screens unless all rules pass.	off

A blocked write is answered with HTTP 403 and a reason such as *Client IP '10.1.2.3' not allowed in area 'Control Room'*, and is logged as the event `area_write_denied`.

Example — setpoints only from the control room. Create an area **Control Room**, **Allowed Roles** Operator and Administrator, **Allowed IPs** `192.168.10.0/24`, **Tag Patterns** `Tank_101.SP_*` and `Pump_P201.Speed_SP`, and tick **Enforce Writes**. An operator logged in on an office PC (`192.168.20.x`) can view Tank_101 but cannot change its setpoints.

Note: The client IP is the address of the TCP connection. The `X-Forwarded-For` header is used only when the connection comes from the server itself (`127.0.0.1` or `::1`), that is, from a reverse proxy running on the OptiZeus server; the first address in the header is then taken. A header sent by any other client is ignored, so it cannot be used to pass an IP rule. A reverse proxy on a different computer therefore makes every client appear with the proxy's address — write the IP rules for that address, or run the proxy on the OptiZeus server. (Before 2.6.1.24 the header was accepted from any client.)

24.5 Managing user accounts

24.5.1 The Users page

Open **Config** → **Admin** → **Users**. The upper part is the form **Add New User** (or **Edit User: <name>**), the lower part **Existing Users** shows one card per user with its role badge, name, e-mail, **Status** (**Active** or **Disabled**) and the buttons **Edit** and **Delete** (not shown for `admin`).

Field	Meaning	Rules
Username *	Login name. Matching at login is case-insensitive.	Unique. Cannot be changed after creation (the field is disabled in edit mode).
Full Name	Shown in the header and in reports.	Optional.
Email	Used for notifications and to match single sign-on users (Section 24.9).	Optional.
Password * / New Password (leave blank to keep current)	Initial password or a reset password.	Must pass the complexity rule (Section 24.6.1).
Role *	Administrator, Operator or Viewer.	See Section 24.3.3 for custom roles.
Group	Group selection.	Not stored in 2.6.1.26 (groups are held in memory only).
User Enabled	Clear to block the account without deleting it.	admin cannot be disabled.

24.5.2 Procedure — create an operator

To create an account for a shift operator of Line 1:

1. Open **Config** → **Admin** → **Users**.
2. Enter **Username** `op.line1.anna`, **Full Name** `Anna Levi`, **Email** (optional).
3. Enter a **Password** that meets the complexity rule, for example 10 characters with upper case, lower case, a digit and a symbol.
4. Select **Role** `Operator`. Leave **User Enabled** ticked.
5. Click **Add User**. The card appears under **Existing Users**.
6. Give the password to the operator through a secure channel and ask her to change it at first login (Section 24.6.4).

The creation is written to the event log (`user_created`) and to the audit trail (action `create`, entity `user`).

24.5.3 Procedure — reset a password, disable or delete a user

1. Click **Edit** on the user's card. The form title becomes **Edit User: <name>** and an information line reminds you to click **Update User**.
2. To reset the password, type it in **New Password (leave blank to keep current)**. The new password must not be one of the user's last N passwords (Section 24.6.3).
3. To block the user, clear **User Enabled**.
4. Click **Update User** (or **Cancel**).

To delete: click **Delete** on the card and confirm *Are you sure you want to delete user: <name>?*. Deleting removes the account; past audit and event records keep the user name.

Tip: Prefer disabling to deleting for anyone who ever signed records. The account name stays reserved, so it cannot be recreated for a different person — a basic requirement of 21 CFR Part 11 §11.100(a) (Chapter 25).

24.5.4 Per-user policy fields

Each user row carries its own policy values. They are created with the defaults below and can be changed only through the REST API (`PUT /api/users/:username`); the **Users** form does not show them.

Column	Meaning	Default
<code>password_expiry_days</code>	Days after the last password change at which the next login requires a change. <code>0</code> = never.	180
<code>min_password_length</code>	Minimum length for this user.	8
<code>require_complex_password</code>	Upper, lower, digit and special character required.	yes
<code>password_history_count</code>	Number of previous passwords that cannot be reused.	3
<code>inactivity_disable_days</code>	Days without login after which the account is disabled automatically.	180
<code>session_timeout_minutes</code>	Stored, but not used for the server-side time-out (Section 24.7).	60
<code>auth_type</code>	<code>local</code> , <code>ad</code> or <code>oidc</code> .	<code>local</code>
<code>ad_domain</code>	Domain to use for this AD user, overrides the global domain.	empty

24.6 Password policy and logout

The global policy is edited in **Config** → **Admin** → **Security** →  **General** and saved with **Save Security Settings**. It is stored in `security-settings.json` in the project folder.

24.6.1 Password Policies

Field	Meaning	Default	Range
Minimum Password Length	Minimum characters.	8	6–32
Password Expire (days)	<i>Force password change after this many days (0 = never).</i>	180	0–365
Password History Count	<i>Cannot reuse last N passwords.</i>	3	0–24
Require Complex Password	<i>Uppercase, lowercase, number, special char.</i>	on	—

A complex password needs at least one upper-case letter, one lower-case letter, one digit and one of the characters `!@#$%^&*()_+=[{}];':"\|,.<>/?`. Error messages list each rule not met, for example *Password must contain at least one number*.

Important — how the global values are applied in 2.6.1.26. The server checks a new password against the **user's own** values (Section 24.5.4). The global **Minimum Password Length**, **Require Complex Password** and **Password History Count** are used only as a fallback when a user's own value is empty, and new accounts always receive their own values (8 / complex / 3). **Password Expire (days)** and **Inactivity Disable (days)** on this page are not copied to the users. In practice the effective policy is: minimum 8 characters, complex, last 3 passwords blocked, expiry after 180 days, disable after 180 days without login — unless you change the per-user values through the API.

24.6.2 Password expiry

At login the server compares the time since `password_changed_at` with the user's `password_expiry_days`. When the password has expired the login screen opens the dialog **Password Expired - Change Required** with **Current Password**, **New Password**, **Confirm New Password** and the list **Password Requirements**. After a successful change the user enters the application. Expiry is not applied to AD users.

24.6.3 Account Lockout

Field	Meaning	Default	Range
Inactivity Disable (days)	Disable user after N days of no login (0 = never). See the Important note above.	180	0–365
Max Login Attempts	Failed attempts allowed.	5	1–10
Lockout Duration (minutes)	Lock time after too many attempts.	30	1–1440
Session Timeout (minutes)	Inactivity time-out of the server session (Section 24.7).	60	5–480

How the lockout works:

- Attempts are counted **per client IP address**, not per user name, in a 15-minute window. Every call to the login endpoint counts, successful or not.
- When the count exceeds **Max Login Attempts** the address is locked for **Lockout Duration**. The login screen shows *Too many login attempts. Try again later.* (HTTP 429).
- A successful login clears the counter for that address.
- The counters are held in memory; restarting the service clears all lockouts.

Note: Because the counter is per address, several operators sharing one terminal share the counter, and a user who fails on one PC can still try from another. The daily inactivity check disables local accounts (except `admin`) whose last login is older than their `inactivity_disable_days`; the card then shows **Disabled** and the reason is stored (`Disabled due to N days of inactivity`). Re-enable the account with **User Enabled** in the edit form.

24.6.4 Changing your own password

The login screen has a **Change Password** link below the login button; the forced-expiry dialog is described in 24.6.2. From the login screen the dialog asks for **User Name** (filled in from the login form), **Current Password**, **New Password** and **Confirm New Password**, and changes the password of the account named in **User Name**. It is sent to `POST /api/change-password-login` and counts toward the login-attempt limit of the client IP like a login (24.6.3); an unknown or disabled user name is answered like a wrong password (*Current password is incorrect*). After a successful change, log in with the new password.

When you are logged in, **Change Password** changes the password of the logged-in user (`POST /api/change-password`); no user name is asked.

Every change and every failed attempt is written to the audit trail (`password_changed`, `password_change_failed`). AD users are refused with *Password is managed by Active Directory...*

Note: Before 2.6.1.24 the login-screen dialog did not ask for a user name and changed the password of the user whose session was still stored in that browser. The option **Allow Self-Service Password Change** in the Part 11 section of the security page is saved but not enforced in 2.6.1.26.

24.7 Sessions, time-outs and logout

A successful login creates a signed token (valid for at most 24 hours) and a session row. On every request the server checks the session:

Mechanism	Where set	Effect
Server inactivity time-out	Session Timeout (minutes) (global), default 60	If no request was received for longer than this, the session is deleted and the next request answers <i>Session expired due to inactivity</i> . The user is returned to the login screen.
Absolute token life	fixed	24 hours after login the token is no longer accepted even if the user was active.
Client auto-logout	Logout & Auto-Logout → Auto-Logout Timeout (minutes)	The browser or Electron client logs out after this many minutes without keyboard, mouse or touch activity. 0 = disabled (default).
Logout button	Show Logout Button	When cleared, users cannot log out manually (for kiosk terminals).

Note: A screen that is open and receiving live values keeps making requests, so an unattended operator screen normally does not reach the server time-out. If a workstation must log out when unattended, set **Auto-Logout Timeout**, which measures real user activity.

There is no limit on concurrent sessions per user.

24.8 Two-factor authentication (TOTP)

OptiZeus supports time-based one-time passwords (RFC 6238, 6 digits, 30-second step) compatible with Google Authenticator, Microsoft Authenticator, Authy and similar apps. A code is accepted if it matches the current step or one step before/after (± 30 s), so the server clock must be correct (use NTP).

24.8.1 Requiring MFA for everyone

1. Open **Config → Admin → Security →  General** and scroll to **CFR 21 Part 11 Compliance**.
2. Tick **Require Two-Factor Authentication (TOTP)** — *All users will be required to set up and use MFA on next login.*
3. If your own account has no MFA yet, the setup dialog opens immediately; complete it (24.8.2). Otherwise the page shows *✓ Your MFA is set up and active.*
4. Click **Save Security Settings**.

From then on, any user without MFA is shown the setup dialog right after login with the notice *Two-factor authentication is required by your administrator. You must complete setup to continue. Canceling will log you out.*

24.8.2 Setting up an authenticator (user)

The dialog **Setup Two-Factor Authentication** has two steps:

1. **Step 1: Scan QR Code** — open the authenticator app and scan the code. If you cannot scan, type the key shown under *Or enter this key manually*.
2. **Step 2: Verify Code** — enter the 6-digit code the app displays and click **Verify & Enable MFA**. *MFA Enabled Successfully!* confirms. The event is written to the audit trail (`mfa_enabled`).

24.8.3 Logging in with MFA

After a correct user name and password, the login screen asks for the **Two-Factor Authentication Code** (*Enter the code from your authenticator app*). Enter the 6 digits and click **Verify & Login** (**Back to Login** returns to the first step). A wrong code answers *Invalid MFA code* and is logged as a failed login.

24.8.4 Lost authenticator

An administrator can remove a user's MFA enrolment with the API call `POST /api/mfa/disable` with body `{"userId": <id>}`; the action is audited as `mfa_disabled`. There is no button for this in the **Users** page in 2.6.1.26. If MFA is required globally, the user is asked to enrol again at the next login.

```
POST /api/mfa/disable HTTP/1.1
Host: scada-srv:3004
Authorization: Bearer <administrator token>
Content-Type: application/json

{"userId": 7}
```

24.9 Single sign-on with OpenID Connect

Open **Config** → **Admin** → **Security** →  **Identity Providers**, section  **OpenID Connect (OIDC)**. OptiZeus acts as an OIDC relying party using the authorization-code flow and verifies the signed ID token against the provider's published keys.

Field	Meaning	Default
Enable SSO	Shows the SSO button on the login page.	off
Issuer URL *	e.g. <code>https://login.microsoftonline.com/{tenant}/v2.0</code> . The discovery document is read from <code>{issuer}/.well-known/openid-configuration</code> .	—
Client ID *	Application ID registered at the provider.	—
Client Secret	Leave as <code>*****</code> to keep the stored value.	—
Scopes		<code>openid</code> <code>profile</code> <code>email</code>
Redirect URI (configure this on the IdP side)	Leave blank to derive it from the request host; the default is <code><server>/api/auth/oidc/callback</code> .	auto
Sign-in Button Label	Text of the button on the login page.	<code>Sign in with SSO</code>

Field	Meaning	Default
Username Claim / Email Claim	Claims used to find the local user.	preferred_username / email
Auto-provision new users on first SSO login	Create an SSO account when no SSO account matches.	off
Default Role for New Users	Role of auto-provisioned users.	viewer

Buttons: **Save SSO Settings**, **Test Discovery** (shows the endpoints found, or *X Discovery failed: ...*). The page contains a quick-start list for Microsoft Entra ID.

At sign-in OptiZeus looks for an **SSO account** — an account created by SSO auto-provisioning (authentication type `oidc`) — whose **user name equals the username claim or whose e-mail equals the e-mail claim**. The OptiZeus role of that account applies.

- SSO never signs in to a password account (local or AD). If no SSO account matches but a password account has the same user name or e-mail, the login is refused: *The sign-in matches the existing password account '<name>'. Single sign-on does not log in to password accounts. Ask an administrator.* The event `oidc_login_refused` is logged.
- If nothing matches, an SSO account is created with the **Default Role for New Users** when auto-provisioning is on (event `oidc_user_provisioned`); otherwise the login is refused (*No local user matches '<name>' and auto-provisioning is disabled*).
- To give an SSO user more rights, let the user sign in once (so that the account is created), then change that account's **Role** in **Config** → **Admin** → **Users**.

Note — changed in 2.6.1.24: Earlier versions signed in to *any* local account whose user name or e-mail matched a claim, so an identity-provider account with an administrator's user name or e-mail signed in as that administrator. Existing users who signed in through SSO to a password account must now use their password, or have that account removed so that SSO creates its own.

Note: SSO logins do not ask for the OptiZeus TOTP code; the identity provider's MFA applies. Enforce MFA at the identity provider.

24.10 Active Directory / LDAP

OptiZeus can verify a user's password by binding to a domain controller over LDAP (`ldapjs`). The user must exist in OptiZeus with `auth_type = ad`; the OptiZeus role of that account applies. AD users do not change their password in OptiZeus and have no password expiry there.

Important — configuration in 2.6.1.26. The panel  **Active Directory / LDAP** on the **Identity Providers** tab (fields **Enable AD Authentication**, **AD Server (LDAP URL)**, **Domain**, **Base DN**, **Bind User (UPN)**, **Bind Password**, **User Group (for access)**, **Admin Group (for admin access)**) is saved in `security-settings.json`, but the login code reads its AD parameters from the **system settings** table (`ad_enabled`, `ad_server`, `ad_domain`, `ad_base_dn`, `ad_user_filter`, `ad_default_role`, ...). The group fields are not evaluated, and the panel's test button only checks the format of the parameters (it does not contact the server). Until this is unified, configure AD as follows.

To enable AD logins:

1. Set the system settings with an administrator token:

```
PUT /api/system-settings HTTP/1.1
Host: scada-srv:3004
Authorization: Bearer <administrator token>
Content-Type: application/json

{
  "ad_enabled": "true",
  "ad_server": "ldaps://dc01.plant.local:636",
  "ad_domain": "PLANT",
  "ad_base_dn": "DC=plant,DC=local",
  "ad_user_filter": "(sAMAccountName={{username}})"
}
```

2. Create each AD user with `auth_type` `ad` (no password needed):

```
POST /api/users HTTP/1.1
Authorization: Bearer <administrator token>
Content-Type: application/json

{"username": "jsmith", "role": "operator", "full_name": "John Smith", "auth_type": "ad"}
```

3. Test by logging in as `jsmith` with the domain password. OptiZeus binds as `PLANT\jsmith`. A wrong password answers *Invalid AD credentials*; an unreachable server answers *LDAP connection error:*

The same domain check is used wherever OptiZeus re-asks for a password: electronic signatures, alarm acknowledgment with signature, and the change-password dialog.

Tip: Use `ldaps://` (port 636) so that passwords do not cross the network in clear text.

24.11 Connected clients and force disconnect

Open **Config** → **Admin** → **Clients & Shutdown**. The page has the tabs **Connected Clients**, **Electron Client**, **Browser Client**, **Shutdown** and **Station Layouts**.

Connected Clients shows the registered stations (**Client Name**, **Client ID**, **IP Address**, **Touch Screen**, **Last Seen**, actions **Set Layout**, **Edit Layout**, **Clear**, **Remove**) and **Active Sessions (N)** with **Username**, **Role**, **Type**, **IP Address**, **Connected Since** and a **Disconnect** button per session. **Disconnect All Others** disconnects every session except yours.

To remove a user from the system immediately:

1. Open **Config** → **Admin** → **Clients & Shutdown** → **Connected Clients** and click **Refresh**.
2. Click **Disconnect** in the user's row. All live sessions of that user name are closed.
3. The user's screen shows *You have been disconnected by the administrator.* and returns to the login screen.
4. To stop the user from logging in again, clear **User Enabled** in **Admin** → **Users** (Section 24.5.3).

The **Shutdown** tab offers **Disconnect All Clients**, **Disconnect Single Client** (asks for the Client ID), **Force Shutdown Client** (Electron clients only: **Shutdown All Electron Clients**, **Shutdown Single Client**) and **Service Control** (Chapter 26). Each action is written to the event log (`client_disconnected_by_admin` , `all_clients_disconnected` , `client_shutdown_by_admin`).

Note: Force disconnect closes the live connection and makes the client log out. It does not end the session on the server; the session ends at the next logout or time-out. The list and the per-user disconnect cover clients connected over HTTP (port 3004); clients connected over HTTPS (port 3443) are reached by **Disconnect Single Client** with their Client ID.

24.12 Worked example — access plan for a small plant

Person	Account	Role	Letters	Notes
Plant engineer	eng.david	Administrator	A–Z	MFA; e-mail unique
Shift operators (6)	op.<name>	Operator	M (maintenance buttons)	One account each — never shared
Quality manager	qa.miriam	Viewer	—	Reads reports and trends
Control room panel PC	—	—	—	Auto-Logoff 15 min, Show Logout Button on

Security area **Setpoints**: Allowed Roles Operator, Administrator; Allowed IPs `192.168.10.0/24` ; Tag Patterns `*_SP` , `Recipe.*` ; Enforce Writes.

Global settings: Require Complex Password on, Max Login Attempts 5, Lockout Duration 30, Session Timeout 60, Require Two-Factor Authentication on. Default operator and viewer accounts deleted; `admin` password changed and stored in the plant safe.

25. Security, electronic signatures and 21 CFR Part 11

This chapter describes the technical security of an OptiZeus installation — how logins are protected, how HTTPS is enabled, what the server does against abuse, how records are made tamper-evident — and the functions that support regulated (GxP) operation: reasons for changes, re-authentication before critical actions, electronic signatures and the audit trail. It ends with a clause-by-clause mapping to 21 CFR Part 11. It is written for the administrator and for the validation / quality person who must decide whether a configured OptiZeus system meets their requirements. User accounts, roles and MFA are described in Chapter 24; the audit trail viewer in Chapter 18.

25.1 Security architecture at a glance

Layer	Mechanism in 2.6.1.26
Transport	HTTP on port 3004 (default); HTTPS + secure WebSocket on port 3443 when a certificate is installed (25.3).
Identification	Unique user names, bcrypt-hashed passwords (10 rounds), optional TOTP, optional LDAP or OpenID Connect (Chapter 24).
Session	JWT signed with HMAC (HS256) using a per-installation secret; server-side session row with inactivity time-out; 24 h maximum life.
Authorization	Role check on each protected REST route; role checks on live-connection commands; security letters; security areas with IP rules.
Abuse limits	Login rate limiter per client IP; general API limit of 1,500 requests per minute per IP.
Browser hardening	Security headers via Helmet, Content-Security-Policy, CORS limited to known origins.
Record integrity	Audit trail chained with HMAC-SHA256; electronic signatures bound with HMAC-SHA256.
Secrets at rest	<code>.jwt-secret</code> and <code>.hmac-secret</code> generated on first start (64 random bytes each).

25.2 Installation secrets

On first start the server creates two files in the backend folder:

File	Purpose	If lost
<code>.jwt-secret</code>	Signs login tokens.	A new secret is generated; all users must log in again. No data loss.
<code>.hmac-secret</code>	Keys the audit-trail hash chain and electronic signatures.	A new secret is generated; all existing audit records fail integrity verification and all existing signatures fail verification.

Important: Include `.hmac-secret` in your system-level backup (disk image or file backup of the installation folder) and restrict read access to it to administrators and the service account. It is deliberately **not** part of the OptiZeus configuration backup (Chapter 26) and must never be copied to another customer's installation.

Note: The files are created with owner-only permissions on Linux. On Windows they inherit the folder ACL; restrict the `backend` folder to Administrators and the service account (LocalSystem by default).

25.3 HTTPS and secure WebSocket

HTTPS is enabled automatically when both files exist in the backend folder at start-up:

- `ssl-key.pem` — private key (PEM)
- `ssl-cert.pem` — certificate, preferably with the chain (PEM)

The HTTPS server then listens on port **3443** (environment variable `HTTPS_PORT`) and serves the same application and a secure WebSocket (`wss://`) on the same port. The plain HTTP server on port 3004 keeps running.

To enable HTTPS:

1. Obtain a certificate whose name (CN / SAN) matches the host name that clients use, for example `scada-srv.plant.local`. Use your company CA where possible so browsers trust it without warnings.
2. Save the key as `ssl-key.pem` and the certificate as `ssl-cert.pem` in the backend folder of the running installation (on Electron installs: `<install>\electron\resources\backend\`, see Section 26.8).
3. Restart the OptiZeus service (Section 26.6).
4. Check the service log for `[SSL] HTTPS server created` and `OptiZeus HTTPS Server running on https://localhost:3443`.
5. On each client, open `https://scada-srv.plant.local:3443`.

The secure WebSocket verifies login tokens with the same installation secret (`.jwt-secret`) as the rest of the server, so a login made over HTTPS receives live data over `wss://` without any extra setting. Still check live values over HTTPS during commissioning before you close port 3004.

Note — fixed in 2.6.1.24: In 2.6.1.23 the secure WebSocket checked tokens with a different key (an environment variable with a built-in fallback) instead of `.jwt-secret`, so live data over `wss://` could fail to authenticate while REST calls worked. No environment variable is needed any more; a `JWT_SECRET` variable added to the service as a workaround is not read and can be removed.

Tip: When all clients use HTTPS, block port 3004 at the host firewall for everything except `localhost`. The installers do not create firewall rules; open 3443 (and 3004 if still used, 4840 for the OPC UA server) manually.

25.4 Hardening checklist

#	Action	Reference
1	Change the default <code>admin</code> password (the server forces this at the first login with the factory password); delete or re-password the default operator and viewer accounts.	24.2

#	Action	Reference
2	One named account per person; no shared accounts.	24.5
3	Require complex passwords, keep lockout at 5 attempts / 30 min.	24.6
4	Require TOTP for all users, at minimum for administrators.	24.8
5	Install a certificate and use HTTPS; restrict port 3004.	25.3
6	Allow inbound 3004/3443 only from the operator and engineering networks (Windows Defender Firewall rule, scope = remote subnets).	Appendix B
7	Put the server in the control network zone; allow the office network only through a DMZ or reverse proxy. Do not expose OptiZeus to the Internet.	—
8	Protect <code>.hmac-secret</code> , <code>.jwt-secret</code> , <code>ssl-key.pem</code> and licence files with file-system permissions.	25.2
9	Use security areas for setpoints and recipe tags.	24.4
10	Set Auto-Logoff Timeout on shared operator stations.	24.7
11	Configure an off-host backup of the backups folder.	26.3
12	Review the audit trail and run Verify Integrity periodically.	25.8
13	Keep Windows and Node.js patched; apply OptiZeus patches through the documented procedure.	26.8

25.4.1 What the server enforces automatically

Control	Value
Login attempts	Max Login Attempts (default 5) per client IP in 15 min, then locked for Lockout Duration (default 30 min). HTTP 429.
API request limit	1,500 requests per minute per client IP (bulk tag import excluded). HTTP 429.
Request size	JSON bodies up to 25 MB (50 MB for bulk tag import).
Security headers	Helmet defaults (X-Content-Type-Options, Referrer-Policy <code>strict-origin-when-cross-origin</code> , <code>frame-ancestors 'self'</code> , etc.).
Content-Security-Policy	<code>default-src 'self'; scripts 'self' 'unsafe-inline' 'unsafe-eval'; styles 'self' 'unsafe-inline'; images 'self' data: blob:; object-src 'none'; frame-ancestors 'self'.</code>
CORS	Cross-origin browser requests accepted only from <code>localhost / 127.0.0.1</code> on ports 3000, 3004 and 3443 and the vendor's web domain; same-origin requests always allowed.

Note: The CSP allows inline scripts and `eval`, which the screen scripting engine and some libraries require. Shared dashboards (public share links) remove the frame restriction so that they can be embedded in other pages.

25.5 Requiring a reason and re-authentication for writes

In **Config** → **Admin** → **Security** →  **General**, section **CFR 21 Part 11 Compliance**:

Option	Effect
Require Reason for Write Tag Actions	Before a value written from a screen (button, input, faceplate) is sent, the operator must choose a reason from the list or enter a custom one.
Reasons list (<i>Enter reasons, one per line</i>)	Default: Process adjustment , Maintenance , Calibration , Testing , Emergency correction .
Require Login Confirmation for Critical Actions	Before the write, a dialog asks for a user name and password. The write is recorded with <i>Confirmed by: <user></i> .
Require Electronic Signatures for Critical Actions	Enables electronic signatures (25.6).
Require Two-Factor Authentication (TOTP)	See 24.8.
Allow Self-Service Password Change	Stored; not enforced in 2.6.1.26.

Click **Save Security Settings**. Operator stations pick up the reason and confirmation settings at their next login or page reload.

Every successful write is recorded in the **event log** as `tag_write` (category `control`) with the message `Tag <name> value changed: <old> → <new> | Reason: <reason> | Confirmed by: <user>` and the fields tag, before, after, unit, reason and confirmedBy.

Important: The reason and login-confirmation dialogs are enforced by the operator client. A write sent directly to the REST API is accepted without a reason. Tag writes are recorded in the event log, which is **not** part of the HMAC-chained audit trail. If your URS requires writes to be in the tamper-evident trail, record that as a gap in the risk assessment or route writes through a datalog to an external database with its own controls.

Note: Login confirmation checks the entered credentials by performing a login; each confirmation therefore counts toward the login-attempt limit of that terminal.

25.6 Electronic signatures

25.6.1 Where signatures are required

When **Require Electronic Signatures for Critical Actions** is on, the server requires the signer's password and stores a signature for the following actions:

Action	Record type	Meaning stored (text bound into the signature)
Alarm acknowledgment (Alarms panel, REST)	<code>alarm</code>	<i>I acknowledge this alarm and confirm the corrective action</i>
Alarm acknowledgment (live connection)	<code>alarm</code>	<i>I confirm that I have reviewed and acknowledged this alarm</i>
Batch recipe approval (Batch Control)	<code>batch_recipe</code>	<i>I approve this recipe for production use</i>

Action	Record type	Meaning stored (text bound into the signature)
Any action sent by a custom integration to <code>POST /api/esignature/sign</code>	any	text supplied by the caller

With the option on, acknowledging an alarm opens a dialog that asks for the password (and a reason). A wrong password answers *Invalid password - acknowledgment denied* and writes `esig_failed` to the audit trail.

Important — scope in 2.6.1.26. Electronic signatures are implemented for alarm acknowledgment and batch recipe approval only. Recipe download, configuration changes, report approval and tag writes are **not** signed; for tag writes use reason + login confirmation (25.5). The general-purpose `ElectronicSignatureModal` dialog (**Electronic Signature Required**, fields **Password (re-authenticate to sign)** and **Reason / Comment**, buttons **Cancel / Sign & Confirm**) exists in the product but is not connected to any screen function in this version.

An acknowledgment over the live WebSocket connection follows the same rules as the REST path: only `operator` and `admin` may acknowledge (a viewer receives *Insufficient privileges to acknowledge alarms*), and when signatures are required the password is checked **before** the alarm is acknowledged. A missing password is refused with *Password required for electronic signature on alarm acknowledgment*, a wrong one with *Invalid password - acknowledgment denied* (and `esig_failed` in the audit trail); the alarm stays unacknowledged in both cases.

Note — fixed in 2.6.1.24: In earlier versions the live-connection acknowledgment was applied before the password was checked (a wrong password still acknowledged the alarm) and had no role check (a viewer could acknowledge). Both are corrected.

The client learns that signatures are required from the security settings loaded at login; this read is administrator-only, so on stations where a non-administrator logs in, the signature prompt depends on the value last stored in that browser. The server enforces the password on both acknowledgment paths regardless: if the prompt does not appear on a station, the acknowledgment is refused with the message above instead of being accepted without a signature.

25.6.2 What a signature contains

Each signature is a row in `electronic_signatures` :

Field	Content
<code>timestamp</code>	Server time (ISO 8601, UTC)
<code>username</code>	Signer (printed name comes from the user's Full Name)
<code>action</code> , <code>action_meaning</code>	What was signed and the meaning statement
<code>record_type</code> , <code>record_id</code>	The record signed
<code>record_hash</code>	SHA-256 of the record data at signing time
<code>signature</code>	HMAC-SHA256 over user name, timestamp, record hash and meaning, keyed with <code>.hmac-secret</code>

Field	Content
<code>reason</code> , <code>ip_address</code>	Reason entered and client address

Signatures are also forwarded to external databases configured for audit logging (Chapter 19).

25.6.3 Verifying signatures

Request	Purpose	Role
<code>GET /api/esignature/verify/:recordType/:recordId</code>	Lists the signatures of one record, each with <code>verified: true/false</code> (recomputed HMAC).	Any user
<code>GET /api/esignature/history?startDate=&endDate=&username=&recordType=&limit=</code>	Signature history (default limit 500).	admin

```
GET /api/esignature/verify/alarm/1843 HTTP/1.1
Host: scada-srv:3004
Authorization: Bearer <token>
```

A signature shows `verified: false` if any of its bound fields was changed in the database or if `.hmac-secret` was replaced.

25.7 The audit trail

The audit trail (`audit_trail` table) records security and configuration actions: logins and logouts (local and SSO), password changes and failures, MFA enable/disable, user creation, tag creation/deletion, alarm group acknowledgments, SMS/voice acknowledgments, e-signature applied/failed, recipe approval, backup download, settings changes (system settings, redundancy, Electron configuration), imports, automation actions, edge gateway start/stop, historian deletions, PLC program change log entries and failover. User updates and deletions, role and security-area changes, backup creation and restore, and tag writes are written to the **event log** (Chapter 18).

Each audit row stores user name, action, entity type, entity id and name, field name, old and new value, client IP, details, a sequence number, the hash of the previous row and its own hash (HMAC-SHA256 over all these fields). The first row of a chain uses the value `GENESIS` as previous hash.

To view, filter and export the audit trail use the **Audit Trail** view (Chapter 18). **Config** → **Admin** → **Security** →  **General** also contains an **Audit Trail Report** section (**Data Source** *Local Database (OptiZeus)* or a datalog, **Start Date**, **End Date**, **Entity Type**).

25.8 Verifying audit-trail integrity

Click **Verify Integrity** in the Audit Trail view, or call `GET /api/audit/verify-integrity` (admin). The result contains:

Field	Meaning
<code>valid</code>	<code>true</code> if every chained record links to the previous one and its hash recomputes.

Field	Meaning
<code>totalRecords</code>	Rows examined.
<code>verifiedRecords</code>	Rows whose hash was verified.
<code>unchainedRecords</code>	Rows without a hash (written by versions before chaining was introduced).
<code>sequenceGaps</code>	Jumps in the sequence number (deleted rows).
<code>brokenAt</code>	First failing row (<code>id</code> , <code>sequence</code> , and <code>reason</code> : <code>hash_mismatch</code> if the content was altered).

WARNING: Administrators can delete audit records: **Config → Admin → Database → Purge Historical Data → Audit Trail** and **Auto-Purge Internal Database → Audit Trail ... days** (Chapter 26). After any purge the first remaining record no longer links to its predecessor, so **Verify Integrity** reports the chain as broken at that record. In a regulated system: keep audit auto-purge at 0, forward the audit trail to an external database (Chapter 19) with its own retention, and document any purge through change control before running it.

25.9 Mapping to 21 CFR Part 11

The table maps the Part 11 clauses to the product functions and states what the regulated user must add by procedure. Compliance is a property of the validated system in its use environment, not of the software alone.

Clause	Requirement (summary)	OptiZeus function	User responsibility / gap
11.10(a)	Validation	GAMP 5 document set (Chapter 31).	Execute IQ/OQ/PQ on your configuration and version.
11.10(b)	Accurate and complete copies	Audit trail export (CSV/PDF), event and alarm export, report exports, REST export API (Chapter 28).	Define which records are GxP records.
11.10(c)	Record protection / retention	Local SQLite; external database logging for alarms, events, audit (Chapter 19); configuration backups (Chapter 26).	Retention policy; keep auto-purge off for GxP tables; off-host backups.
11.10(d)	Limit system access	Unique accounts, roles (enforced on the server, 24.3.2), security letters, security areas, lockout, time-outs, MFA, forced change of factory passwords.	Access procedure; periodic account review.
11.10(e)	Secure, computer-generated, time-stamped audit trail	HMAC-chained <code>audit_trail</code> with user, time, old/new value; integrity check.	Tag writes and several configuration changes are in the event log, not the chained trail (25.5, 25.7); purge is possible (25.8).
11.10(f)	Operational checks (sequence)	Batch control state machine and transition confirmation; procedures with step order (Chapters 20, 23).	Configure where sequence matters.
11.10(g)	Authority checks	Role checks, security letters/areas, re-authentication.	Assign roles by job function.

Clause	Requirement (summary)	OptiZeus function	User responsibility / gap
11.10(h)	Device checks	Client IP rules in security areas; registered stations (Client ID).	Define allowed terminals.
11.10(i)	Training	This manual.	Training records.
11.10(j)	Accountability policy	—	Written policy on electronic signatures.
11.10(k)	Documentation control	Versioned product documentation.	Control your copies.
11.50	Signature manifestation (name, time, meaning)	Signature stores user name, timestamp, meaning; shown in signature history.	Ensure reports that show signed records print these fields.
11.70	Signature/record linking	HMAC over user, time, record hash and meaning.	Protect <code>.hmac-secret</code> .
11.100(a)	Unique signatures, not reused	Unique user names; deletion recommended to be replaced by disabling (24.5.3).	Never reassign a user name.
11.100(b)	Identity verification before issuing	—	HR / identity procedure.
11.100(c)	Certification to the FDA	—	Letter to FDA.
11.200(a)	Two distinct components (ID + password)	Signature requires the signer's password in an active session; login confirmation requires ID + password.	For the first signing in a session policy, consider also requiring login confirmation.
11.300(a)-(b)	Password uniqueness and ageing	Unique names; expiry; history.	Review the per-user policy (24.6.1).
11.300(c)	Loss management	Admin reset of passwords and MFA (24.5.3, 24.8.4).	Procedure for lost credentials.
11.300(d)	Detect unauthorized use	Failed logins and failed signatures logged; lockout.	Review security events.
11.300(e)	Device testing (tokens)	TOTP apps.	Test authenticator devices.

25.10 Worked example — configuring a GMP mixing line

Requirement: operators may change Tank_101 temperature setpoints only with a reason and a second-person confirmation; alarm acknowledgments must be signed.

1. **Config → Admin → Security →  General → CFR 21 Part 11 Compliance:** tick **Require Reason for Write Tag Actions**, keep the default reasons and add `Deviation DEV-xxx`; tick **Require Login Confirmation for Critical Actions**; tick **Require Electronic Signatures for Critical Actions**; tick **Require Two-Factor Authentication (TOTP)**. Click **Save Security Settings**.
2. ** Security Areas:** area **Mixing Setpoints**, roles Operator + Administrator, IPs `10.20.1.0/24`, tag pattern `Tank_101.TT_SP*`, **Enforce Writes**.
3. **Config → Admin → Database → Auto-Purge Internal Database:** all three fields `0`.
4. Configure an event and audit datalog to the site SQL server (Chapter 19).

5. OQ: write `Tank_101.TT_SP` from the faceplate and verify the event text contains `Reason:` and `Confirmed by:` ; acknowledge an alarm with a wrong password and verify it is refused and `esig_failed` is in the audit trail; run **Verify Integrity** and record `valid: true` .

26. Backup, restore and maintenance

This chapter describes how to protect an OptiZeus project and keep the server healthy: what the built-in configuration backup contains (and what it does not), creating, scheduling, downloading and restoring backups, historian archive backups, database maintenance (purge, auto-purge, VACUUM), log files and their rotation, starting and stopping the service, and the procedure for installing a patch or upgrade. It closes with the most common maintenance mistake — working on the wrong database. It is written for the administrator responsible for the server.

26.1 What must be protected

An OptiZeus installation holds four kinds of data. The built-in backup covers only the first.

Data	Location	Covered by
Project configuration — tags, screens, super objects, UDTs, alarm configuration, PLC connections, recipes, reports, schedules, notifications, datalog definitions, users, system settings, OPC UA server settings, and the main JSON configuration files	Project database <code>scada.db</code> and JSON files in the project folder	Backup & Restore (26.2–26.5)
Runtime records — alarm history (<code>alarms</code>), event log, audit trail, electronic signatures, local historian table, sessions	Project database <code>scada.db</code>	Not in the configuration backup. Use a file backup of the stopped database (26.4.4) and/or external database logging (Chapter 19).
Historian archive — <code>.hda</code> files of the file-based historian	Historian archive folder	Historian Archive Backup (26.5)
Installation secrets and certificates — <code>.jwt-secret</code> , <code>.hmac-secret</code> , <code>ssl-key.pem</code> , <code>ssl-cert.pem</code> , licence files, <code>active-project.json</code>	Backend folder	System/file backup of the installation folder only.

Important: A configuration backup restores a working project on a new PC, but it does not bring back alarm history, events, the audit trail or signatures. For a regulated system, plan both: the configuration backup **and** a periodic file-level backup of the whole project folder taken while the service is stopped, or continuous forwarding of records to an external SQL server.

26.2 The Backup & Restore page

Open **Config** → **Admin** → **Backup & Restore**. The page has these sections:

Section	Purpose
Automatic Daily Backup	Schedule (26.3).
Create New Backup	Manual backup (26.4.1).
Upload Backup File	Import a <code>.zip</code> backup made on another server (26.4.3).
Available Backups	List with Info , Download , Restore , Delete per backup.

Section	Purpose
Historian Archive Backup	Backup of the <code>.hda</code> archive (26.5).

All backup functions except listing require the `admin` role.

26.2.1 Contents of a configuration backup

A backup is a ZIP file containing `backup-data.json` with the rows and column schema of these tables:

`synoptic_screens`, `charts`, `users`, `password_history`, `system_settings`, `tags_config`, `plc_connections`, `opcua_servers`, `opcua_tag_mappings`, `ethernetip_servers`, `ethernetip_tag_mappings`, `udt_definitions`, `udt_instances`, `super_objects`, `super_object_instances`, `alarm_config`, `data_sources`, `datalogs`, `gsm_modems`, `notification_users`, `notification_groups`, `notification_settings`, `report_templates`, `visual_report_layouts`, `recipes`, `schedules`, `opcua_server_settings`, `opcua_server_tags` (and `roles`, `groups`, `user_groups` if such tables exist).

and these project files when present: `super-objects.json`, `notification-settings.json`, `alarm-settings.json`, `notification-history.json`, `service-config.json`, `security-settings.json`, `layout-settings.json`, `roles.json`.

Not included: alarm history, event log, audit trail, electronic signatures, historian data, security areas (`security-areas.json`), the OIDC configuration, Sparkplug and ERP configurations, redundancy and deployment configuration, SVG library items stored in the global database, uploaded images and videos, secrets and certificates.

WARNING: A backup contains the user table with password hashes and the notification contacts, and `system_settings` may contain connection passwords. Store downloaded backups like any other credential file.

26.3 Scheduled (automatic) backups

Field	Meaning	Default
Enable auto backup	Turns the daily backup on.	off
Backup Time	Time of day (server local time, HH:MM).	02:00
Retention (days)	Automatic backups older than this are deleted from the project's <code>backups</code> folder after each automatic backup. <code>0</code> = keep all automatic backups.	30

To schedule a nightly backup:

1. Open **Config** → **Admin** → **Backup & Restore**.
2. In **Automatic Daily Backup**, tick **Enable auto backup**, set **Backup Time** `02:30` and **Retention (days)** `30`.
3. Click **Save Settings**.

The server checks once a minute. When the clock matches **Backup Time** and no automatic backup was made that day, it creates `auto-backup-<timestamp>.zip` with the description `Auto backup - <date>` and writes the event `auto_backup`. The date of the last automatic backup is shown on the page.

Note — limitations in 2.6.1.26: - If the service is not running during the scheduled minute, that day's backup is skipped (there is no catch-up). - With a retention of 0 nothing is deleted; watch the size of `backend\projects\<name>\backups` and delete old files manually. - Only files named `auto-backup-*.zip` are purged; manual backups are never deleted automatically.

Note — fixed in 2.6.1.25: Up to 2.6.1.24 a retention of 0 was treated as 30 days, and the clean-up looked in `backend\backups` instead of the project's `backups` folder, so old automatic backups of projects other than *Default* were never deleted. After the upgrade, the first automatic backup removes all automatic backups in the project folder that are older than the retention — copy any you want to keep elsewhere first.

Tip: The backups folder is on the same disk as the database. Copy it to another machine or a network share with a Windows scheduled task (e.g. `robocopy` nightly at 03:00). A backup that dies with the server disk is not a backup.

26.4 Manual backup and restore

26.4.1 Creating a backup

1. In **Create New Backup**, leave **Include database tables (synoptics, tags, users, etc.)** and **Include configuration files (super objects, notifications, alarms)** ticked.
2. Enter a **Backup Description (optional)**, e.g. `Before upgrade to 2.6.1.26`.
3. Click **Create Backup**. A progress indicator is shown; on success *Backup created successfully*.

The backup appears in **Available Backups** as `scada-backup-<date>T<time>.zip` with **Size** and **Created**. The event `backup_created` is written. Creation may take up to five minutes on very large projects.

Click **Info** to see **Filename**, **Created**, **Application**, **Version**, **Description**, the list of **Tables** with record counts, and **Configuration Files**. **Download** saves the ZIP to your PC (the download is recorded in the event log and the audit trail).

26.4.2 Restoring a backup

WARNING: A restore changes the running project immediately: tags, alarm configuration and PLC connections are reloaded and datalogs restarted. Restoring PLC connections or tag addresses that belong to another plant can cause the server to read from and write to the wrong controller. Restore only backups whose origin you know, preferably while the plant is in a safe state.

1. Create a fresh backup of the current state (26.4.1) so that you can go back.
2. Click **Restore** on the backup. The dialog **Restore Backup** opens with *Warning: Restoring a backup will modify your current data.* and loads the backup contents.
3. Select the categories to restore. Each category shows **Backup: n** and **Current: n** records; categories without data in the backup are hidden. Use **Select All** / **Deselect All**.

Category	Tables
Tags	<code>tags_config</code>

Category	Tables
Screens	<code>synoptic_screens</code>
Super Objects	<code>super_objects</code> , <code>super_object_instances</code>
Alarm Configuration	<code>alarm_config</code>
UDTs	<code>udt_definitions</code> , <code>udt_instances</code>
PLC Connections	<code>plc_connections</code> , OPC UA and EtherNet/IP servers and mappings
Reports	<code>report_templates</code> , <code>visual_report_layouts</code>
Recipes	<code>recipes</code>
Schedules	<code>schedules</code>
Notifications	GSM modems, notification users, groups and settings
Datalogging	<code>data_sources</code> , <code>datalogs</code>
Users & Security	users, password history, roles, groups
Charts	<code>charts</code>
System Settings	<code>system_settings</code>
OPC UA Server	OPC UA server settings and exposed tags
Configuration Files	the JSON files listed in 26.2.1

- If **Users & Security** is selected, decide on **Override Users & Passwords from Backup**. Cleared (default): *Only roles and groups will be restored. Current users and passwords will be kept.*
- Under **When items already exist:** choose **Replace existing items with backup version** (default) or **Skip existing items (only add new)**.
- Tick **Clear Existing Data Before Restore** only if the selected categories must contain exactly the backup content — *This will delete ALL current data in selected categories before restoring!*
- Click **Restore Selected** and confirm the summary (categories, counts, mode).
- The page reports *Restore completed! n tables and n config files restored. Reloading...* and reloads. All connected clients receive a notice and refresh.

If some tables fail, the message lists them (*Restore completed with errors: ...*). The event `backup_restored` records the file, tables and options.

Note: Changes to `security-settings.json` , `roles.json` and some other configuration files take full effect only after a service restart (26.6).

26.4.3 Moving a project to another server

- On the old server, create and **Download** a backup.
- Install OptiZeus on the new server and activate its licence (Chapter 2).
- Create the project on the new server (Chapter 5) and make it active.
- Upload Backup File:** click or drag the `.zip` (*Only .zip backup files are accepted*). The file is stored as `imported-<timestamp>-<original name>.zip`.

5. **Restore** it with all categories, mode **Replace**.
6. Copy items not in the backup by hand: security areas, OIDC, Sparkplug, ERP settings, uploaded images, certificates.
7. Check PLC connection addresses before enabling communication.

26.4.4 File-level backup of the database

To capture runtime records (alarm history, events, audit trail, signatures) as well:

1. Stop the OptiZeus service (26.6). Data acquisition stops during this time.
2. Copy the whole project folder `backend\projects\<name>\` (including `scada.db` and, if present, `scada.db-wal` and `scada.db-shm`) to the backup location.
3. Start the service.

Important: Do not copy `scada.db` alone while the service is running. The database runs in WAL mode; recent changes may still be in `scada.db-wal`, and a copy taken during a write can be inconsistent.

26.5 Historian archive backup

The section **Historian Archive Backup** (badge **HDA Files**) applies to the file-based historian (Chapter 19). It shows **Archive Status**, **Total Size**, **Total Records** and **Tags Archived**.

1. Optionally set **Start Date (optional)** and **End Date (optional)** to limit the range.
2. Click **Create Historian Backup**. With a date range the page notes *Partial backup: only archive files within the selected date range will be included*. A `historian-backup-<timestamp>.zip` appears under **Historian Backups** (**Refresh** updates the list).
3. Use **Download**, **Restore** or **Delete** on each entry.

Restore asks *Restore historian archive from "<file>"? This will add missing .hda files to the archive directory...;* files already present and up to date are not overwritten. Historian backups are stored in `backend\backups` and require the historian service to be active on this server.

26.6 Service control

OptiZeus runs as the Windows service **OptiZeus** (WinSW wrapper `backend\daemon\optizeus.exe`, Node.js, service account LocalSystem by default). The wrapper restarts the service automatically after a crash (after 5 s, 10 s, then 30 s).

Method	Start	Stop	Restart
Windows Services console	Start	Stop	Restart
Administrator command prompt	<code>net start OptiZeus</code>	<code>net stop OptiZeus</code>	stop, then start
Config → Admin → Clients & Shutdown → Shutdown → Service Control	—	Stop (confirmation: <i>STOP the OptiZeus service? ... You will need to start the service manually from Windows Services.</i>)	Restart (All clients will briefly disconnect and reconnect automatically. — back in about 10–15 s)

The panel also shows **Service Status: RUNNING, STOPPED, NOT INSTALLED, UNREACHABLE** or **CHECKING...**

WARNING: Stopping or restarting the service stops PLC communication, alarm detection, historian recording, datalogs and scheduled actions for the duration. Outputs written by OptiZeus keep their last value in the PLC. Coordinate with operations.

Note: Node.js does not reload changed program files. Any change to backend files (patch, certificate, some configuration files) needs a service restart. Frontend-only changes need only a browser refresh (Ctrl+F5).

26.7 Database maintenance

Open **Config** → **Admin** → **Database**. *Manage local SQLite database storage. Historical data should be stored in external databases via Datalog.*

26.7.1 Database Statistics

Shows **Database File Size**, **Tag History Records**, **Alarm Records**, **Event Log Records** and a configuration line (Tags, Alarms, Screens, Audit). **Refresh** re-reads.

26.7.2 Purge Historical Data

Select tables and click **Purge Selected Tables**. The selected tables are **emptied completely**.

Choice	Table
Tag History	tag_data
Alarms History	alarms
Event Log	event_log
Audit Trail	audit_trail
Deploy History	deploy_history
Pending Changes	pending_changes (staged, not yet deployed changes!)
User Sessions	user_sessions (logs out every user)

WARNING: Purging is permanent and cannot be undone. Purging **Audit Trail** destroys records required in regulated operation and breaks the integrity chain (Chapter 25). Purging **Pending Changes** discards staged configuration work. Take a file-level backup (26.4.4) first.

26.7.3 Auto-Purge Internal Database

Automatically delete old records from the internal SQLite database. Set to 0 to keep data indefinitely. Runs every hour.

Field	Deletes	Default
Alarm History ... days	alarms rows older than N days	0
Event Log ... days	event_log rows older than N days	0

Field	Deletes	Default
Audit Trail ... days	<code>audit_trail</code> rows older than N days	0

Click **Save Auto-Purge Settings** (*Changes take effect within 1 hour*). The status line reads *Auto-purge is disabled (all set to 0)* or lists the active rules.

Note: In 2.6.1.26 the fields are always displayed as 0 when the page opens, even when values are stored (display problem). Saving the page therefore overwrites the stored values with what is shown — re-enter all three values every time you save.

Recommended values for a non-regulated plant with external logging: Alarm History 365, Event Log 180, Audit Trail 0.

26.7.4 Reclaim Disk Space

Run **VACUUM** to reclaim disk space after purging data. Click **Run VACUUM**. The database file is rewritten without free pages and the result shows size before, after and reclaimed. **VACUUM** locks the database while it runs (seconds to minutes for a large file) and needs free disk space about equal to the database size; run it at a quiet time.

26.8 Log files and rotation

Log	Location	Rotation
Service output (console)	<code>backend\daemon\optizeus.out.log</code>	WinSW rotate mode: at 10 MB the file is renamed <code>optizeus.0.out.log</code> , older ones shift to <code>.1 7</code> ; eight old files kept (about 90 MB).
Service errors	<code>backend\daemon\optizeus.err.log</code>	Same scheme.
Wrapper log	<code>backend\daemon\optizeus.wrapper.log</code>	Grows slowly; delete when large.
Event log	<code>event_log</code> table (and external datalog if configured)	Auto-purge (26.7.3).

Every console line is prefixed with a local timestamp `[YYYY-MM-DD HH:MM:SS.mmm]`. The logs can be read in **Config** → **Admin** → **System Logs** (Chapter 29).

Note: On installations where the backend runs from `C:\OptiZeus\electron\resources\backend\`, the service logs are still written under `C:\OptiZeus\backend\daemon\`.

26.9 Installing a patch or upgrade

OptiZeus is updated with a **patch package** (`patch-v<version>.zip`) that contains `INSTALL-PATCH.bat`, the backend files and the compiled frontend. The patch keeps the project databases, projects folder, licence and settings.

26.9.1 Procedure

1. Read the release notes of the patch. Plan a window with operations (communication stops for 1–3 minutes).
2. Create a configuration backup and **Download** it (26.4.1). For regulated systems also take a file-level backup (26.4.4).
3. Record the current version: open `http://<server>:3004/api/update/current-version` in a browser (or see Chapter 29).
4. Close all Electron viewers on the server PC (the installer replaces `app.asar`, which is locked while the viewer runs).
5. Extract the ZIP to a local folder (not to the installation folder).
6. Right-click `INSTALL-PATCH.bat` → **Run as administrator**. If OptiZeus is not at `C:\OptiZeus`, enter the installation path when asked.
7. The installer performs these steps:

Step	Action
[1/8]	Stops the OptiZeus service.
[2/8]	Copies the current <code>server.js</code> , <code>license.js</code> , <code>drivers</code> , <code>services</code> to <code>backend\backup-pre-<version></code> .
[3/8]–[4/8]	Copies the new backend core and modules (<code>drivers</code> , <code>services</code> , <code>middleware</code> , <code>utils</code> , <code>workers</code> , <code>scripts</code> , <code>daemon</code>).
[5/8]	Installs <code>super-objects.json</code> only if none exists.
[6/8]	Copies the frontend to <code>backend\public</code> and <code>backend\frontend</code> .
[6.4]	Replaces the Electron viewer (<code>electron\resources\app.asar</code>) if present.
[6.45]	Also refreshes <code>electron\resources\backend</code> if the installation runs its backend from there.
[6.5]–[6.6]	Seeds the pipe and callout symbol libraries (idempotent).
[7/8]	Re-creates the Windows service if it is missing.
[8/8]	Starts the service and checks that it runs.

8. Wait about 30 seconds, then log in and verify: version (step 3 again), PLC connections green, live values, alarms, one trend.
9. Clear the browser cache on clients (Ctrl+F5) or restart Electron viewers.

Note — known issue: step [8/8] decides whether the service is running with `sc query OptiZeus | findstr "RUNNING"`. On Windows installations with a non-English display language this check can report *Service did not enter RUNNING state* although the service is running. Check the state in the **Services** console before taking action.

Note: The installers do not create Windows Firewall rules. After installing on a new server, open the ports you use (3004, 3443, 4840 for the OPC UA server) manually — see Appendix B.

26.9.2 Rolling back a patch

1. Stop the service.
2. Copy the files from `backend\backup-pre-<version>` back over `backend` (and over `electron\resources\backend` if used).

3. Frontend files are not backed up by the installer; re-run the previous version's patch package instead if the frontend must also be reverted.
4. Start the service and verify.

26.9.3 Online auto-update

Config → **Admin** → **Auto-Update** can download and install an update ZIP from an update server. See Section 27.9 for its limitations; the manual procedure above is recommended.

26.10 The many-databases trap

An OptiZeus server usually holds several SQLite databases:

File	What it is
backend\scada.db	The Default project (and the shared SVG symbol library for all projects).
backend\projects\ <name>\scada.db	One per project. The active project is named in backend\active-project.json .
electron\resources\backend\...	A second copy on Electron-hosted installations.
releases\... , backup folders	Old copies.

Before you inspect, copy, restore or repair a database, make sure it is the one the running service uses:

1. Look at the service log at start-up: [Project] Active: <name> (<folder>) .
2. Or open /api/diagnostics (Chapter 29), which returns project , projectDir and dbPath .
3. Compare screen names and tag counts with what the user interface shows.

Important: Profiling, repairing or restoring the wrong scada.db gives wrong conclusions and — worse — changes that "do nothing". The same applies to patches: on Electron-hosted installations the running code is electron\resources\backend\ , not backend\ (the patch installer handles both, manual file copies must too).

26.11 Recommended maintenance schedule

Interval	Task
Daily (automatic)	Configuration backup at 02:30; copy backups off the server.
Weekly	Check Config → Admin → Database statistics and disk space (Chapter 29); review failed logins in System Logs.
Monthly	Download a backup and test-restore it on a test PC; review users (disable leavers); run Verify Integrity on the audit trail.
Quarterly	File-level backup of the project folder with the service stopped; VACUUM after purges; apply Windows updates in a planned window.
Before every change	Backup with a description; follow change control; record the version.

27. Servers, multi-site, redundancy and store-and-forward

This chapter covers everything that involves more than one OptiZeus server or a server that must survive network loss: the multi-server deployment pipeline (test → production → backup), server redundancy and failover, the multi-site dashboard, store-and-forward buffering of external database writes, MQTT Sparkplug B publishing, offline mode of the browser client, and the online auto-update function. It is written for the system integrator who designs the server architecture and for the administrator who operates it.

Several of these functions are partly implemented in 2.6.1.26. Each section states exactly what works, so that you can design around the gaps. Where a function is not reliable, the recommended alternative is given.

27.1 Overview and licensing

Function	Where	Licence	Status in 2.6.1.26
Multi-server deployment (Test → Production → Backup, dedicated Historian/Batch servers)	Config → Admin → Multi-Server (top part)	Enterprise tier or multi-server feature	Configuration, health checks and service routing work; pushing a deployment to another server does not work (27.2.5).
Server redundancy (heartbeat, failover, client redirect)	Config → Admin → Multi-Server (lower part)	none	Heartbeat, partner status, manual failover and client redirect work; data sync and PLC take-over are not implemented (27.3).
Multi-site dashboard	Config → Admin → Multi-Site	none	Online/offline status works; alarm and tag counts are not shown (27.4).
Store & forward	Config → Data Storage →  Store & Forward	none	Working.
Sparkplug B publisher	Config → Admin →  Sparkplug B	none	Working (NBIRTH/NDATA/NDEATH).
Offline mode	automatic in the browser	none	Working on HTTPS or localhost only; shows cached data, refuses tag writes (27.8).
Auto-update	Config → Admin → Auto-Update	none	Not usable without manual API configuration (27.9).

See Appendix C for the licence tiers.

27.2 Multi-server deployment

27.2.1 Concept

In a regulated or large plant, engineering changes are made on a **Test** server and promoted to **Production** after testing; a **Backup** server mirrors production. In the largest topology the historian and the batch engine run on their own servers. Each server runs the same OptiZeus software; its **role** decides which services it runs.

Topology	Servers	Description (as shown)
Standalone	1	Single server runs all services
Production + Backup	2	Production + hot backup, all services on Production
Test + Prod + Backup	3	Development → Production → Backup pipeline
Full (5-Server)	5	Dedicated Historian + Batch Control servers

Services activated by role in the **Full (5-Server)** topology (all other topologies run all services on every server):

Role	Core SCADA	Historian	Batch engine	Alarm notification
Test Server (Development)	Yes	Yes	Yes	Yes
Production Server	Yes	No	No	Yes
Backup Server	Yes	No	No	Yes
Historian Server	No	Yes	No	No
Batch Control Server	No	No	Yes	No

The active flags of a server are returned by `GET /api/health/details` (`activeServices`). Historian API requests to a server whose historian service is off are refused; the clients route historian and batch requests to the dedicated servers using `GET /api/deployment/service-routing` .

27.2.2 The Multi-Server page

Open **Config** → **Admin** → **Multi-Server**. Without an Enterprise licence the page shows *Enterprise License Required — Multi-server deployment (Test → Production → Backup) is an Enterprise feature.*

Section	Content
Deployment Topology	Four topology cards; click to select.
Server Configuration	This Server Role: drop-down; one block per server (Test Server , Production Server , Backup Server , Historian Server , Batch Control Server) with Name ; URL : (e.g. <code>http://production-server:3004</code>), API Key ; and Generate . Edit / Save / Cancel .
Server Status	Health of each configured server (online / error / offline, version, uptime). Refresh .
Deployment Actions	Deploy to Production (on a Test or Standalone server when a production server is enabled), Sync Backup Server , Sync Historian Server , Sync Batch Server .
Deployment History	Date/Time , Type , Deployed By , Targets , Status . <i>No deployment history yet. when empty.</i>

27.2.3 API keys between servers

Servers authenticate to each other with API keys sent in the `X-API-Key` header. The receiving server accepts a key if it equals one of the keys stored in its own server configuration.

To pair a Test server with a Production server:

1. On the **Production** server, in the **Test Server** block, click **Generate** and copy the key.

2. On the **Test** server, in the **Production Server** block, enter the production **URL** and paste the same key into **API Key**.
3. Click **Save** on both servers.

Keys are shown as ******* after saving.

27.2.4 Deploy to Production (dialog)

Deploy to Production opens a dialog **Select Items to Deploy**: with **Screens, Tags, Alarms, PLC Connections, Recipes** (each with the number of items) and the warning *This will update the production server configuration. Backup server will also be updated automatically.*

27.2.5 Limitation — use backup/restore to promote changes

Important — not functional in 2.6.1.26. The receiving side of a deployment (`/api/deployment/receive`) fails: it tries to create a pre-deployment backup through an interface that does not exist, and its insert statements use column names that do not match the current `tags_config` and `synoptic_screens` tables. **Deploy to Production** and the **Sync ...** buttons therefore do not transfer configuration, even if the history shows an attempt.

Promote changes from Test to Production with the configuration backup instead:

1. On Test: **Config** → **Admin** → **Backup & Restore** → **Create Backup** (description e.g. `Release 12 - Tank farm screens`), then **Download**.
2. On Production: **Create Backup** (rollback point), then **Upload Backup File** and **Restore** only the categories you changed (e.g. **Screens, Super Objects**), mode **Replace existing items with backup version** (Chapter 26).
3. Verify on Production; repeat step 2 on the Backup server.

Within one server, use the staging workflow (Chapter 5) to review changes before they go live.

27.3 Server redundancy and failover

27.3.1 How it works in 2.6.1.26

Two OptiZeus servers are installed with the same project. Each is told the URL of its partner. Every **Heartbeat Interval** a server calls `GET /api/redundancy/heartbeat` on its partner (no login needed; answers `{"status":"alive","role":...}`).

Event	What OptiZeus does
Partner answers	Partner status online ; last heartbeat time stored.
Partner stops answering	Partner status offline ; critical event <code>redundancy_partner_offline</code> .
Standby has missed heartbeats for Failover After	Standby changes its own role to primary , logs <code>redundancy_auto_failover</code> (critical) and tells its own connected clients <i>This server is now the active primary (failover)</i> .
Partner comes back after a failover	Event <code>redundancy_partner_restored</code> .
Administrator clicks Force Failover	All clients connected to this server receive a failover message and are asked <i>Server failover initiated! ... Redirect to backup server?</i> — OK opens the partner URL.

Important — what redundancy does not do in 2.6.1.26: - No automatic data or configuration synchronisation. Sync Database Now / Sync Database to Partner only check that the partner answers. Keep the two projects identical with backup/restore (27.2.5), and log alarms, events and history to a shared external database (Chapter 19) so that records are not split between servers. - **No PLC take-over logic.** Both servers communicate with the PLCs all the time; the role is informational. This is acceptable for reading, but both servers can also **write** (weekly schedules — run by every server since 2.6.1.26 — automation actions, alarm pipelines, recipe downloads). Disable automatic writers on the standby, or make sure they are idempotent. - **No automatic client redirect on automatic failover.** Clients are redirected only by **Force Failover** on the server they are connected to. After an unplanned loss of the primary, operators open the standby URL themselves (put a shortcut on every station) or you use a virtual IP / DNS switch. - The heartbeat uses plain HTTP; a partner URL starting with `https://` is not monitored correctly.

27.3.2 Configuration

The **Multi-Server** page contains two redundancy panels that write the same server-side settings.

Server Redundancy & Auto-Failover (recommended):

Field	Meaning	Values
This Server Role	Primary (Active) or Standby (Hot Backup) .	—
Partner Server URL	e.g. <code>http://192.168.1.101:3004</code> .	—
Heartbeat Interval	Check period.	3, 5, 10 seconds
Failover After	Time without heartbeat before a standby takes over.	10, 15, 30, 60 seconds
Auto-Failback	Yes / No.	Stored; the server does not automatically fail back in 2.6.1.26.

Buttons: **Enable & Save** (*Server redundancy enabled! Heartbeat monitoring started.*), **Disable**. **Live Status** shows partner status and last heartbeat; **Actions** contains **Sync Database to Partner** and **Force Failover Now** (*FORCE FAILOVER? This will notify ALL connected clients to switch to the partner server.*).

Server Redundancy & Failover (older panel) has **Primary Server URL**, **Partner/Backup Server URL**, **Heartbeat Interval**, **Failover Timeout**, **Auto-Failback**.

WARNING: Saving the older panel always sets the role of *this* server to **primary**. Never use it on the standby server — it silently turns the standby into a second primary and disables automatic failover.

27.3.3 Procedure — two-server set-up

1. Install OptiZeus on `SCADA-A` (192.168.1.100) and `SCADA-B` (192.168.1.101); activate both licences.
2. Build the project on `SCADA-A`; copy it to `SCADA-B` with backup/restore.
3. On `SCADA-A`: **Server Redundancy & Auto-Failover** → **This Server Role** *Primary (Active)*, **Partner Server URL** `http://192.168.1.101:3004`, **Heartbeat Interval** 5 s, **Failover After** 15 s → **Enable & Save**.
4. On `SCADA-B`: role *Standby (Hot Backup)*, partner `http://192.168.1.100:3004`, same timing → **Enable & Save**.

5. Open the firewall on both servers for TCP 3004 from the partner.
6. Check **Live Status** on both: partner **online**.
7. Test: stop the service on SCADA-A . After about 15 s SCADA-B logs *Automatic failover activated* and becomes primary. Start SCADA-A again and note that SCADA-B stays primary until you reconfigure it.
8. Document the operator instruction for switching stations to SCADA-B .

27.4 Multi-site dashboard

Config → **Admin** → **Multi-Site** lists other OptiZeus servers (for example at other plants) and shows whether they are reachable.

Button / column	Meaning
+ Add Site	Asks <i>Site name:</i> and <i>Site URL:</i> (e.g. <code>http://plant2-scada:3004</code>).
Refresh Sites	Queries all enabled sites (results are cached for 10 s).
Site card	Name, URL, Online / Offline .
Open	Opens the site's URL in a new tab.
Remove	Deletes the site.

For each site the server calls the remote `/api/ready` and `/api/health` (no login needed) and marks it **online**, **degraded** (answers but not ready) or **offline**.

Note — limitations in 2.6.1.26: the alarm and tag counts on the cards are not filled (the page and the server use different field names), and the remote site's alarm and tag lists require a login token, which the **+ Add Site** dialog does not ask for. HTTPS certificates of remote sites are not validated. Use the dashboard as a reachability overview; to operate a remote site, click **Open** and log in there.

27.5 Store & forward

When a write to an **external** database fails — historian records, alarms, events or audit records sent to SQL Server, MySQL, PostgreSQL etc. through datalogs (Chapter 19) — the record is buffered locally and forwarded when the database is reachable again. Nothing is lost during network or database outages up to the buffer limit.

27.5.1 How the buffer works

Item	Value
Storage	Append-only files (<code>JSON Lines</code> , one record per line) in the archive folder; SQLite holds only the file list and statistics.
Default folder	<code>backend\store-forward-archive\</code>
File naming	<code><queueType>_<targetId>_<YYYYMMDD-HH>-<random>.jsonl</code> , e.g. <code>historian_5_20260927-14-3K9x.jsonl</code>
File rotation	Every hour, or at 10 MB.

Item	Value
Forwarding	Every 30 s the sealed (older) files are forwarded in chronological order, 500 records per attempt; a file is deleted when empty. The file of the current hour is forwarded only by Flush or once it is sealed.
Capacity	2 GB per queue (queue type + target); beyond that the oldest files are deleted and counted as Dropped (overflow) .

Note: The information text on the page (... saved to a local SQLite queue ... If a queue exceeds 50,000 records, the oldest records are dropped) describes an earlier design; the limits above are those of 2.6.1.26.

27.5.2 The Store & Forward page

Open **Config** → **Data Storage** →  **Store & Forward**.

Element	Meaning
Archive Folder	Current folder, with  Copy ,  Open (opens Explorer on the server PC; useful in the Electron client on the server) and  Edit → path field (e.g. D:\optizeus-buffer), Save Path , Cancel .
Statistics	Currently Buffered, Total Enqueued, Total Forwarded, Dropped (overflow), Failed Attempts.
Queue table	Type, Target Datalog, Records, Oldest Record, Last Retry, Retries, Action (flush / clear). Refresh .

Clearing a queue asks *Clear all <type> records for this queue? This cannot be undone.* and writes the event `store_forward_cleared`.

Tip: Put the archive folder on a different disk from the operating system if long outages are expected: one historian queue with 1,000 tags at 1 s can grow by several hundred MB per day.

27.6 MQTT Sparkplug B publishing

OptiZeus can publish its tags to an MQTT broker using the Sparkplug B specification, so that an MQTT-based IIoT platform or a second SCADA can consume them. (Reading data *from* MQTT devices is a PLC connection, see Chapter 6.)

Open **Config** → **Admin** →  **Sparkplug B**. The status badge shows **CONNECTED**, **DISCONNECTED** or **DISABLED**.

Field	Meaning	Default
Enable Sparkplug B Publishing	Master switch.	off
Broker URL *	mqtt://host:1883 or mqtt://host:8883 .	mqtt://localhost:1883
Username, Password	Broker credentials (password shown as <code>*****</code> after saving).	empty
Group ID *	Sparkplug group, e.g. MyFactory .	OptiZeus
Edge Node ID *	Edge node name, e.g. Gateway1 .	Node1
Device ID (optional)	If set, messages are published as DBIRTH/DDATA for this device instead of NBIRTH/NDATA.	empty

Field	Meaning	Default
Host ID (optional)	Primary host application ID to follow (STATE topic).	empty
Client ID (optional)	MQTT client ID; auto-generated if blank.	auto
Publish only changed tags (recommended)	NDATA contains only changed values.	on
Use JSON payload (for debugging — non-standard)	Plain JSON instead of protobuf.	off

Buttons:  **Save & Reconnect** and  **Restart**. The **Status** block shows **Connection Status**, counters **Published**, **Births**, **Deaths**, **Errors**, and **Topic namespace**: (`spBv1.0/<group>/<type>/<node>[/<device>]`), **Current seq**:, **bdSeq**:, **Payload format**: (*Protobuf (Sparkplug B standard)* or *JSON (debug)*), **Last birth**:, **Connected at**:.

Message sequence: NBIRTH with all tags on connect, NDATA with changed tags (sequence 0–255), NDEATH as MQTT will message on disconnect. The configuration is stored in `backend\sparkplug-config.json`.

Example. Publish all tags of Line 1 to a site broker: Broker URL `mqtt://broker.plant.local:8883`, Group ID `PlantA`, Edge Node ID `Line1_SCADA`, publish changed only. A consumer subscribes to `spBv1.0/PlantA/#`.

Note: All tags of the project are published. Use a broker ACL to restrict who may subscribe, and use `mqtt://` with credentials on any network beyond the control network.

27.7 Edge gateway

The server also contains an edge-gateway service (MQTT-based forwarding of tag data) controlled by `GET /api/edge-gateway/status` and start/stop requests that are audited. It has no dedicated page in the Admin menu of 2.6.1.26; use Sparkplug B (27.6) for standard publishing.

27.8 Offline mode (browser client)

The web client registers a service worker. When the server is unreachable:

- the application shell (page, logo) loads from the browser cache;
- the last responses of `/api/tags`, `/api/synoptic-screens` and `/api/alarms` (cached for up to 5 minutes) are shown instead of failing;
- **tag writes** (POST/PUT to `/api/tags/...`) are **refused**, never stored: the client receives HTTP 503 with "Not written: the OptiZeus server is not reachable. Nothing was sent to the plant.", and the screen shows **Write Failed!** with that message. Nothing is sent later.

When the connection returns, check the current values and repeat any write that is still needed. Values shown during an outage are stale; do not operate equipment while the connection indicator shows the connection lost.

Note — changed in 2.6.1.24: Earlier versions stored tag writes made during an outage in the browser (*Write queued for sync*) and sent them when the server was reachable again — possibly minutes later, into a process that had moved on. From 2.6.1.24 writes are never queued, and writes that an older version had queued in a browser are discarded unsent once the 2.6.1.24 client runs in that browser. To be certain after the upgrade, clear the browser site data on stations that were used during an outage before the upgrade.

Service workers only run on secure origins: HTTPS, or `http://localhost` . On clients that open `http://<server>:3004` over the network, offline mode is not active.

27.9 Auto-update

Config → **Admin** → **Auto-Update** (*Check for updates, download patches, and install them*):

Element	Function
Update Server Configuration	Field for the update server URL (e.g. <code>https://updates.example.com</code>) and Save URL .
Check for Updates	Asks the update server for its <code>/api/version</code> ; shows Update Available! with Latest version ;, Changelog ;, a Download URL field and Download Update ;, or <i>You are running the latest version</i> .
Install Update	After a download: extracts the ZIP over the backend folder and restarts the service (<i>Install update? The service will restart. Are you sure?</i>).
Update History	Load History: Date, Action, Details.

Important — limitations in 2.6.1.26: - **Save URL** does not store the URL (the setting does not exist yet in the settings table and the save only updates existing settings), so **Check for Updates** answers *No update server configured*. The URL can be created only by direct database maintenance. - The check compares against a fixed internal version string, not the running version, and uses plain HTTP. - **Install Update** copies files only into the backend folder the service runs from; it does not update `backend\public / frontend` separately, does not update `electron\resources\backend` , does not verify a signature of the package and does not keep a rollback copy. Use the patch installer procedure in Section 26.9 for all updates.

27.10 Choosing an architecture

Plant	Recommended
One line, one control room	Standalone; nightly backup copied off-server; external SQL logging optional.
Continuous process, downtime of the HMI unacceptable	Two servers with redundancy heartbeat (27.3), identical projects via backup/restore, shared external SQL database for alarms/events/history, store & forward on both, a documented manual switch of stations.
Regulated (GMP) with change control	Separate Test server for engineering; promote by backup/restore with change record; Production + Backup as above; audit trail forwarded to external database.
Several plants	One server per plant; Multi-Site page for reachability; Sparkplug B to a central broker for enterprise data.

28. Integration: REST API, webhooks, MQTT, ERP

This chapter explains how other systems exchange data with OptiZeus: the REST API (authentication, reading tags, writing values, exporting history, alarms and events for business-intelligence tools), webhooks that push alarms and events to chat and ticketing systems, MQTT (Sparkplug B publishing and MQTT as a data source), the SAP/ERP connector, and — by cross-reference — the built-in OPC UA server. It is written for integrators and IT engineers who connect OptiZeus to MES, ERP, BI and messaging systems.

28.1 Integration options at a glance

Need	Use	Direction	Section
Read live values or history from a script, BI tool or MES	REST API	pull	28.2–28.4
Write a setpoint from another system	REST API	push into OptiZeus	28.3.3
Notify Teams/Slack/ticketing when an alarm occurs	Webhooks	push out	28.5
Stream all tags to an IIoT platform	MQTT Sparkplug B	push out	28.6
Read devices that publish to MQTT	MQTT PLC connection	pull in	Chapter 6
Send production counts to ERP	ERP connector	push out	28.7
Let another SCADA/MES/OPC client browse and read tags	OPC UA server	client/server	Chapter 8
Log to an enterprise SQL database	Datalogs / external databases	push out	Chapter 19

28.2 REST API basics

Item	Value
Base URL	<code>http://<server>:3004/api</code> or <code>https://<server>:3443/api</code>
Format	JSON (UTF-8); some export endpoints return CSV
Authentication	Bearer token obtained from <code>POST /api/login</code>
Built-in endpoint list	<code>GET /api/docs</code> (no login; partial list, labelled with an older version number)
Rate limit	1,500 requests per minute per client IP → HTTP 429
Login limit	Max Login Attempts (default 5) logins per 15 minutes per client IP → HTTP 429, then locked for 30 min
Errors	JSON <code>{ "error": "<text>" }</code> with HTTP 400, 401, 403, 404, 429 or 500 (502 for a PLC write that did not reach the controller, 28.3.3). A database error is answered with HTTP 500; up to 2.6.1.24 some routes sent no answer at all in that case, so a client waited until its own time-out. Always set a client time-out (for example 30 s).

28.2.1 Service account for integrations

Create a dedicated OptiZeus user for each external system (Chapter 24), e.g. `svc.mes` with role `viewer` for read-only access, `operator` if it must write values, or `admin` only if it must create or change tags, datalogs or historian settings (these requests are administrator-only since 2.6.1.24, Chapter 24.3.2). Do not enable TOTP for service accounts (the API login would then require a code), and compensate with a long random password and a security area that limits the account's writes to certain tags and its client IP.

28.2.2 Logging in

```
POST /api/login HTTP/1.1
Host: scada-srv:3004
Content-Type: application/json

{"username": "svc.mes", "password": "<password>"}
```

Response (abridged):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": { "id": 12, "username": "svc.mes", "role": "viewer", "full_name": "MES interface" },
  "passwordExpired": false,
  "defaultPassword": false,
  "mfaSetupRequired": false
}
```

`"passwordExpired": true` means the password has expired or, with `"defaultPassword": true`, that a factory account still uses its default password (since 2.6.1.24). The token is issued anyway; change the password (`POST /api/change-password` with `currentPassword` and `newPassword`) instead of relying on it.

Send the token in every following request:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

28.2.3 Token lifetime — important for integrations

- A token is rejected after **Session Timeout (minutes)** without any request (default 60) — *Session expired due to inactivity* — and at the latest **24 hours** after login.
- On HTTP 401, log in again once and repeat the request.
- **Do not log in for every request.** Each login counts toward the login limit of the client IP; a script that logs in six times in 15 minutes locks out every user working from the same address.
- Call `POST /api/logout` when the integration stops.
- Passwords of service accounts expire like any other (default 180 days); set the account's `password_expiry_days` to 0 through the API if your policy allows, or plan the rotation.

28.2.4 Endpoints that need no login

Endpoint	Returns
<code>GET /api/health</code>	<code>{"status": "ok", "timestamp": ...}</code>
<code>GET /api/ready</code>	<code>{"ready": true, "tagsCount": ...}</code>

Endpoint	Returns
GET /api/update/current-version	{"version": "2.6.1.26", "buildDate": "2026-09-28"}
GET /api/docs	Endpoint catalogue

28.3 Live data

28.3.1 All tags with current values

```
GET /api/tags HTTP/1.1
Authorization: Bearer <token>
```

Returns one object keyed by tag name, each with `value`, `quality`, `timestamp`, `unit`, `label`, `folder`, `server_name`, `plc_address` and the tag configuration. `GET /api/tags/list` returns an array with the same content.

28.3.2 Export endpoint for BI tools

`GET /api/export/tags` returns a flat list suitable for Power BI, Excel or a data lake.

Parameter	Meaning	Default
<code>format</code>	<code>json</code> or <code>csv</code>	<code>json</code>
<code>folder</code>	Only tags in this folder and its sub-folders; empty string = tags without folder	<code>all</code>
<code>quality</code>	<code>good</code> , <code>bad</code> , ...	<code>all</code>
<code>page</code> , <code>pageSize</code>	Paging; <code>pageSize</code> max 10,000	1, 1000

The response header `X-Total-Count` gives the total number of matching tags.

```
GET /api/export/tags?folder=Tank_101&format=csv HTTP/1.1
Authorization: Bearer <token>
```

```
name,value,quality,unit,timestamp
Tank_101.LT,72.4,good,%,2026-09-27T08:15:02.114Z
Tank_101.TT,64.9,good,°C,2026-09-27T08:15:02.114Z
Tank_101.LSHH,false,good,,2026-09-27T08:15:02.114Z
```

28.3.3 Writing a value

```
POST /api/tags/Pump_P201.Speed_SP/write HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{"value": 1450, "reason": "MES order 4711 changeover", "confirmedBy": null}
```

Rule	Detail
Role	<code>operator</code> or <code>admin</code> (a <code>viewer</code> gets HTTP 403).
Licence	Requires an active licence or trial.
Security areas	Checked (Chapter 24); a blocked write returns 403 with the reason.
Array elements	Use the index in the name: <code>Recipe_Active[3]</code> , <code>Matrix[1,2]</code> (URL-encode the brackets).
Record	Written to the event log as <code>tag_write</code> with old value, new value, reason and user; a failed write as <code>tag_write_failed</code> .
Response	<code>{"success":true,"message":"Value written successfully","tag":...,"value":...}</code>
PLC not reached	HTTP 502 <code>{"success":false,"error":"Not written: the PLC connection \"<name>\" is not connected or rejected the write.", "tag":...}</code> . The tag keeps its old value. Since 2.6.1.24; earlier versions answered success with <i>Value updated locally</i> .
Memory / simulated tag	<code>{"success":true,"message":"Memory tag updated",...}</code> or <code>"Simulated tag updated (no PLC behind this tag)"</code> — the value exists only in OptiZeus.

WARNING: A REST write goes to the PLC exactly like an operator action, without the reason and confirmation dialogs of the screens. Give integration accounts write access only to the tags they need (security area with **Enforce Writes**), validate values in the calling system, and agree interlocks with the PLC programmer. Never let an external system write to safety-related tags.

28.3.4 Active alarms

`GET /api/alarms` returns the current alarms (active or unacknowledged). Acknowledging through the API (`POST /api/alarms/:id/acknowledge` with `{"reason": "..."}`) requires `operator` or `admin` , and a password in `esigPassword` when electronic signatures are enabled (Chapter 25).

28.4 Exporting history, alarms and events

28.4.1 Historian

`GET /api/export/historian`

Parameter	Meaning
<code>tags</code>	Comma-separated tag names (required)
<code>startDate</code> , <code>endDate</code>	ISO 8601 (required)
<code>interval</code>	Aggregation interval in ms; 0 or absent = raw values
<code>format</code>	<code>json</code> or <code>csv</code>
<code>page</code> , <code>pageSize</code>	<code>pageSize</code> max 50,000 (default 5,000)

The data source is the file-based historian archive if enabled, otherwise the historian external database, otherwise the local `historian_data` table. Rows are pivoted: one row per timestamp, one column per tag. With `interval` , values are averaged per bucket.

```
GET /api/export/historian?tags=Tank_101.LT,Tank_101.TT&startDate=2026-09-26T06:00:00Z&endDate=2026-09-26T14:00:00Z&interval=60000&format=json HTTP/1.1
Authorization: Bearer <token>
```

```
{
  "data": [
    { "timestamp": "2026-09-26T06:00:00.000Z", "Tank_101.LT": 70.12, "Tank_101.TT": 64.7 },
    { "timestamp": "2026-09-26T06:01:00.000Z", "Tank_101.LT": 70.35, "Tank_101.TT": 64.8 }
  ],
  "totalCount": 480,
  "page": 1,
  "pageSize": 5000
}
```

28.4.2 Events

GET /api/export/events?startDate=...&endDate=...&type=tag_write&format=csv — columns `id`, `timestamp`, `event_type`, `event_category`, `description`, `severity`, `source`. Without dates the last 24 hours are returned. `pageSize` max 10,000.

28.4.3 Alarms

GET /api/export/alarms is documented with the parameters `startDate`, `endDate`, `severity`, `format`, `page`, `pageSize`.

It returns the local alarm history (`alarms` table) for the period, newest first, as JSON { `data`, `totalCount`, `page`, `pageSize` } or, with `format=csv`, as a CSV file with the columns `timestamp`, `tagName`, `type`, `severity`, `message`, `acknowledged`, `acknowledgedBy`, `acknowledgedAt`. The header `X-Total-Count` gives the number of rows. Without dates the last 24 hours are returned. `severity` and `message` are derived as for the shift report (Chapter 21, Section 21.5): `critical` for `high_high`, `low_low` and communication alarms, otherwise `warning`; the message is `<type> = <value> (limit <threshold>)`. Filter with `severity=critical` or `severity=warning`.

Note — fixed in 2.6.1.26: Up to 2.6.1.25 this endpoint read a table (`alarm_history`) that no database contains and answered HTTP 500 *Failed to export alarm history*.

28.4.4 Example — daily shift export with PowerShell

```
$base = "http://scada-srv:3004/api"
$login = Invoke-RestMethod "$base/login" -Method Post -ContentType "application/json" `
        -Body '{"username":"svc.report","password":"<password>"}'
$h      = @{ Authorization = "Bearer $($login.token)" }
$start  = (Get-Date).Date.AddHours(-18).ToUniversalTime().ToString("o")
$end    = (Get-Date).Date.AddHours(6).ToUniversalTime().ToString("o")
Invoke-WebRequest "$base/export/historian?
tags=Line1.Count,Line1.Speed&startDate=$start&endDate=$end&interval=300000&format=csv" `
        -Headers $h -OutFile "D:\exports\line1_$((Get-Date).ToString('yyyyMMdd')).csv"
Invoke-RestMethod "$base/logout" -Method Post -Headers $h
```

Run it with Windows Task Scheduler at 06:05. One login per run keeps well inside the login limit.

28.5 Webhooks

A webhook sends an HTTP request to an external URL when something happens in OptiZeus. Typical targets: Microsoft Teams or Slack incoming webhooks, a ticketing system, an on-call service, a custom REST service.

28.5.1 Triggers

Trigger type	Fired when	Data available
<code>alarm</code>	Any alarm is raised	tag name, value, severity, message, timestamp
<code>alarm_critical</code>	An alarm with severity <code>critical</code> is raised (in addition to <code>alarm</code>)	same
<code>alarm_cleared</code>	An alarm-category event <code>alarm_cleared</code> , or a PLC reconnects	severity, message, event type, category, source
<code>event</code>	Any event is logged (except webhook failures)	severity, message, event type, category, source
<code>tag_change</code>	Accepted by the configuration, but not fired automatically in 2.6.1.26	—
<code>custom</code>	Only when an administrator calls <code>POST /api/webhooks/fire</code>	the data sent in that call

Caution: the `event` trigger fires for *every* event — logins, configuration changes, tag writes, communication messages. Always set a **filter** (28.5.3) on `event` webhooks, or the receiver is flooded.

28.5.2 Managing webhooks

Open **Config** → **Admin** → **Webhooks**. Click **Refresh** to load the list (**Name**, **URL**, **Trigger**, **Enabled**, **Last Status**, **Actions** with **Test** and **Del**).

To add a webhook:

1. Click **+ Add Webhook**.
2. Answer the prompts *Webhook name:*, *Webhook URL:* and *Trigger type* (*alarm*, *alarm_critical*, *alarm_cleared*, *event*, *custom*):.
3. The webhook is created enabled, method `POST`, 3 retries, 5 s time-out, with the generic payload.
4. Click **Test**. OptiZeus sends a test payload (tag `TEST_TAG`, value 100, severity critical) and shows the receiver's HTTP status in **Last Status**.

Filter, payload template, headers, method, retries and time-out are set through the API:

```

PUT /api/webhooks/3 HTTP/1.1
Authorization: Bearer <admin token>
Content-Type: application/json

{
  "name": "Teams - Line 1 critical",
  "url": "https://example.webhook.office.com/webhookb2/...",
  "method": "POST",
  "headers": {},
  "trigger_type": "alarm_critical",
  "trigger_filter": "Line1",
  "payload_template": "{\@type\":"MessageCard\","summary\":"{message}\","themeColor\":"{severityColor}\","text\":"🔥 {tagName}: {message}\"}",
  "enabled": true,
  "retry_count": 3,
  "timeout_ms": 5000
}

```

Field	Meaning	Default
name , url	Required.	—
method	HTTP method.	POST
headers	JSON object of extra headers (e.g. an API key for the receiver). Content-Type: application/json is always sent.	{}
trigger_type	See 28.5.1.	alarm
trigger_filter	Case-insensitive text; the webhook fires only if the tag name or the message contains it.	empty (all)
payload_template	Body with placeholders (28.5.3). Empty = generic template.	generic
enabled		true
retry_count	Attempts; waits 1 s, 2 s, 4 s ... between them.	3
timeout_ms	Per attempt.	5000

Built-in templates (GET /api/webhooks/templates):

Name	Template
generic	{"alarm":"{tagName}","value":"{value}","severity":"{severity}","message":"{message}","timestamp":"{timestamp}"}
slack	{"text":"🔥 {severity}: {tagName} - {message}"}
teams	MessageCard with summary , themeColor and text

28.5.3 Placeholders

{tagName}, {value}, {severity}, {message}, {timestamp}, {severityColor} (hex colour: critical FF0000, high FF4500, medium FFA500, low FFD700, info 00BFFF, other 808080), {eventType}, {category}, {source}, {triggerType}.

Note: Placeholders are inserted as plain text without JSON escaping. An alarm message containing a double quote (") or a backslash produces invalid JSON and the receiver rejects it. Keep alarm messages free of these characters for tags that feed webhooks.

A webhook that fails after all retries logs the warning event `webhook_failed` (it does not trigger other webhooks).

28.6 MQTT

- **Publishing (OptiZeus → broker):** MQTT Sparkplug B, see Section 27.6. All tags are published under `spBv1.0/<Group ID>/NBIRTH|NDATA|NDEATH/<Edge Node ID>[/<Device ID>]` with protobuf payload (or JSON for debugging).
- **Subscribing (broker → OptiZeus tags):** add a PLC connection with protocol MQTT (TLS optional), then map tags to topics. See Chapter 6. Tag writes to MQTT tags publish to the configured topic.

Example Sparkplug consumer check with the Mosquitto command-line client (JSON payload option on):

```
mosquitto_sub -h broker.plant.local -p 1883 -t "spBv1.0/PlantA/#" -v
```

28.7 SAP / ERP connector

Open **Config → Admin → ERP Connector**. The page title is **SAP/ERP Connector**. The connector sends JSON documents from OptiZeus to REST endpoints of an ERP or MES system.

28.7.1 Connection Settings

Field	Meaning
ERP Base URL	e.g. <code>https://erp.example.com/api</code> .
Authentication Type	API Key (sent as <code>X-API-Key</code>), Basic Auth (Username / Password) , Bearer Token , OAuth2 Client Credentials (Token URL, Client ID, Client Secret) .
Sync Interval (seconds, 0 = disabled)	Period of the automatic send. Minimum 60 s.
Enable Connector	Disabled / Enabled .

Buttons: **Load Config**, **Save Config** (*Configuration saved successfully.*), **Test Connection** (a GET to the base URL; *Connection successful! HTTP 200 OK* or *Connection failed: ...*). Secrets are shown masked after saving. The configuration is stored in `backend\erp-config.json` .

Note: **OAuth2 Client Credentials** can be selected and saved, but no token is requested in 2.6.1.26 — requests are sent without authorization. Use API key, basic or bearer authentication.

28.7.2 Data Mappings and Manual Data Send

Data Mappings assigns an endpoint path to each data type (defaults `/api/production` , `/api/oeo` , `/api/batch` , `/api/alarms` , `/api/quality`). **Manual Data Send** posts a JSON payload of the selected type — **Production**

Counts, OEE Metrics, Batch Results, Alarm Summary, Quality Data — immediately with **Send Now** (*Data sent successfully (HTTP 201)*).

For the scheduled sync, the payload is built by OptiZeus:

Data type	Scheduled payload in 2.6.1.26
Production counts	Up to 100 tags with good quality: name, value, unit
All other types	Only a timestamp and the type name (placeholder)

Important — limitations: the scheduled sync is started when the service starts; after changing the interval or enabling the connector, restart the service. Only production counts carry real data automatically. For OEE, batch or quality data, call the ERP from a script or scheduled report, or have the ERP pull from the REST API (28.4).

28.7.3 Sync Log

Load Log shows the last 200 entries: **Date**, **Action** (`test_connection` , `send` , `scheduled_sync`), **Status**, **Details**.

28.7.4 Example — production counts to MES every 15 minutes

1. **ERP Base URL** `https://mes.plant.local/api` ; **Authentication Type** **API Key**, key from the MES team.
2. **Sync Interval** `900` , **Enable Connector Enabled**; **Save Config**; **Test Connection**.
3. Restart the OptiZeus service (Chapter 26).
4. In the MES, accept POST `/api/production` with body `{"tags": [{"name": "Line1.Count", "value": 15320, "unit": "pcs"}, ...]}`.
5. After 15 minutes, **Load Log** shows `scheduled_sync` with status `ok`.

28.8 OPC UA server

OptiZeus can expose selected tags as an OPC UA server (default endpoint `opc.tcp://<server>:4840/UA/OptiZeus`) for MES, historians and other OPC UA clients. Configuration, security modes and exposed tags are described in Chapter 8. The OPC UA server is a licence add-on (Appendix C). Open TCP 4840 in the firewall manually; the installers do not create firewall rules.

29. System logs, performance and diagnostics

This chapter describes the tools for finding out what the OptiZeus server is doing: the **System Logs** page (event log and service console log), the **Performance** monitor, the version and health endpoints, the diagnostic endpoints for communication and the background worker threads, and the messages that the server writes at start-up and during operation. It is written for administrators and for support engineers who investigate a problem. Chapter 30 uses these tools to solve specific problems.

29.1 Where to look first

Question	Tool	Section
Which version is running?	GET <code>/api/update/current-version</code>	29.4
Is the server up and has it finished loading?	GET <code>/api/health</code> , GET <code>/api/ready</code>	29.5
Which project and database is active?	GET <code>/api/diagnostics</code> , start-up log	29.5, 29.7
What happened (logins, writes, configuration changes, communication loss)?	System Logs → Event Logs	29.2
Why did something fail inside the server?	System Logs → Console Logs	29.3
Is the server overloaded?	Performance	29.6
Is a PLC being read, how fast?	PLC Connections page (Chapter 6), console log, <code>/api/opcua/stats</code>	29.8
Is the historian recording?	<code>/api/historian/worker-status</code> , console log	29.8

29.2 System Logs — Event Logs

Open **Config** → **Admin** → **System Logs**. The page **System Logs** has two tabs: **Event Logs** and **Console Logs**.

Event Logs shows the event log of the active project (the same records as the Events view, Chapter 18). If an external event datalog is enabled (Chapter 19), the records are read from that database, and from the local database if the external one returns nothing.

Control	Options
Time Range	Last 1 hour , Last 6 hours , Last 12 hours , Last 24 hours (default), Last 48 hours , Last 7 days
Category	All Categories or a category present in the log (e.g. <code>security</code> , <code>system</code> , <code>config</code> , <code>control</code> , <code>alarm</code> , <code>plc</code> , <code>communication</code> , <code>maintenance</code> , <code>webhook</code>)
Severity	All Severities , Info , Warning , Error , Debug
Search	Text in the description or the event type (<i>Search logs...</i>)
Auto-refresh	Reloads every 30 seconds (on by default)

Control	Options
Clear Filters	Resets all filters
Export CSV	Saves the displayed page as CSV (Timestamp, Type, Category, Severity, Source, Description, Details)
Paging	100 records per page; First , Previous , Next , Last

Columns: **Timestamp**, **Type**, **Category**, **Severity**, **Source**, **Description**. Click a row to see its details (JSON), e.g. the before/after values of a tag write.

Note: In 2.6.1.26 the **Category** column of the exported CSV is empty for records read from the local database (the export uses a different field name). Use the Events view export (Chapter 18) if you need the category in a file.

29.2.1 Useful event types

Event type	Category	Meaning
<code>login_success</code> , <code>login_failed</code> , <code>logout</code>	security	Logins (failed logins include the reason: unknown user, disabled, AD, MFA).
<code>user_created</code> , <code>user_updated</code> , <code>user_deleted</code> , <code>user_disabled</code>	security	Account changes; <code>user_disabled</code> = automatic inactivity disable.
<code>security_settings_updated</code> , <code>role_*</code> , <code>area_*</code> , <code>area_write_denied</code>	security	Security configuration and blocked writes.
<code>unauthorized_action</code>	security	A user tried a command his role does not allow.
<code>client_disconnected_by_admin</code> , <code>all_clients_disconnected</code> , <code>server_shutdown</code> , <code>service_restarted</code> , <code>service_stopped</code>	security / system	Administrative actions.
<code>tag_write</code>	control	Value written, with reason and confirmation.
<code>tag_write_failed</code>	control	A write to a PLC tag did not reach the PLC (connection down or write rejected); nothing was changed.
<code>default_password_login</code>	security	A factory account logged in with its default password; the password change was forced (Chapter 24).
<code>backup_created</code> , <code>backup_restored</code> , <code>auto_backup</code> , <code>backup_download</code>	system / security	Backups.
<code>auto_purge</code>	maintenance	Records deleted by auto-purge.
<code>webhook_failed</code>	webhook	A webhook did not reach its receiver.
<code>redundancy_partner_offline</code> , <code>redundancy_auto_failover</code> , <code>redundancy_failover</code>	system	Redundancy (Chapter 27).

Event type	Category	Meaning
deployment_*	system	Multi-server deployment attempts.

29.3 System Logs — Console Logs

The **Console Logs** tab (administrators only) shows the end of the service's console output files.

Control	Options
Log Type	Output Log (<code>optizeus.out.log</code>) or Error Log (<code>optizeus.err.log</code>)
Lines	Last 100 , Last 200 (default), Last 500 , Last 1000
Filter	Text filter (<i>Filter logs...</i>)
Auto-refresh	Every 5 seconds (on by default)
Export Log	Saves the displayed lines

Lines are coloured by a simple rule: lines containing *error*, *exception* or *failed* are shown as errors, *warn* as warnings, *debug* as debug. Every line begins with the local time stamp of the server.

The files are read from `backend\daemon\` of the running backend and rotate at 10 MB (Chapter 26). Older rotated files (`optizeus.0.out.log` ... `optizeus.7.out.log`) can be opened with any text editor on the server.

Tip: To follow the console live on the server without the browser, open PowerShell and run `Get-Content C:\OptiZeus\backend\daemon\optizeus.out.log -Wait -Tail 50`.

29.4 Version information

Source	Example
GET <code>/api/update/current-version</code> (no login)	<code>{"version":"2.6.1.26","buildDate":"2026-09-28"}</code>
Config → Admin → Auto-Update → Current Version:	Not reliable in 2.6.1.26: after Check for Updates it shows a fixed internal string (<code>2.5.4.18</code>), not the running version.
Patch installer title	OptiZeus SCADA - Patch v2.6.1.26

Always quote this version (four parts: major.minor.patch.build) when reporting a problem.

Note: Other version strings exist in the product and are **not** the running version: the `/api/docs` catalogue says `2.5.3`, the update check compares against an internal fixed string, and project exports carry their own format version.

29.5 Health and diagnostic endpoints

Endpoint	Login	Returns	Use
GET /api/health	no	<code>{"status":"ok","timestamp":...}</code>	Is the web server alive? Used by the redundancy partner and multi-site.
GET /api/ready	no	<code>{"ready":true/false,"tagsCount":n}</code>	false while the tags are still being loaded after a start.
GET /api/health/details	yes	status healthy / starting , server role, topology, active services, uptime	Multi-server set-ups.
GET /api/diagnostics	yes	project name, project folder, database path and size, WAL size, tags in memory, connected PLC count, read cycle time, memory, Node.js version, host name, record counts	First check in any support case.
GET /api/system/performance	yes	CPU per core, memory, disks, network interfaces, WebSocket client counts, tag update cycle statistics, PLC connection statistics	Detailed performance (API only).
GET /api/opcua/stats	yes	per OPC UA connection: batch-read capability, subscription state, monitored items, age of cached values, reconnecting flag	OPC UA communication.
POST /api/opcua/reset-capability/:serverId	yes	clears the cached batch-read capability so the next read re-probes	After an OPC UA server upgrade.
GET /api/historian/worker-status?tags=a,b	yes	worker active or fallback mode, per-tag buffer state	Historian recording.
GET /api/license/status	no	licence state, tier, trial days	Licence (Chapter 2).

Example — identify the active project and database:

```
GET /api/diagnostics HTTP/1.1
Host: scada-srv:3004
Authorization: Bearer <token>
```

```
{
  "project": "Line1_2026",
  "projectDir": "C:\\OptiZeus\\backend\\projects\\Line1_2026",
  "dbPath": "C:\\OptiZeus\\backend\\projects\\Line1_2026\\scada.db",
  "dbSize": "184.3 MB",
  "walExists": true,
  "walSize": "4120 KB",
  "tagsInMemory": 1834,
  "plcConnected": 3,
  "uptime": 86412.5,
  "nodeVersion": "v24.12.0",
  "communication": { "connectedPlcs": 3, "tagsCount": 1834, "readCycleMs": 212 },
  "tables": { "tags": 1834, "recipes": 12, "udts": 9, "udtInstances": 64, "screens": 41,
  "plcConnections": 3 }
}
```

Note: `plcConnected` counts the S7, Modbus and OPC UA client objects; EtherNet/IP connections handled by the worker thread are not included. Check them on the PLC Connections page.

29.6 Performance monitor

Open **Config** → **Admin** → **Performance**. The page **Performance Monitor** reads `/api/diagnostics` and, with **Auto-refresh (5s)** ticked, updates every five seconds.

Panel	Values	What to watch
System	Uptime, Node.js Version, Platform, Hostname	Unexpected short uptime = the service restarted (look in the console log for the reason).
Memory Usage	RSS (Resident), Heap Used, Heap Total, External (MB)	Heap growing steadily over days indicates a leak; note the values daily. Known issue in 2.6.1.26: this panel divides values that are already in MB by 1024^2 and therefore shows values near 0 MB. Read the <code>memory</code> block of <code>/api/diagnostics</code> (values in MB) or the <code>node.exe</code> process in Windows Task Manager instead.
Database	DB File Size, WAL Size, Table Count	A WAL that stays at hundreds of MB points to a long-running reader or a blocked checkpoint; restart during a quiet period.
Communication	Connected PLCs, Tags Count, Read Cycle Time	Read cycle time close to or above the fastest scan rate means the scan rates cannot be met.

GET `/api/system/performance` returns more detail for scripted monitoring:

Field	Note
<code>cpu.avgUsagePercent</code> , <code>cpu.cores[]</code>	Calculated from the processor counters since the computer started — a long-term average, not the current load. Use Windows Task Manager or Performance Monitor for the current load.
<code>memory</code>	Total, used, free (MB, whole computer) and the Node.js heap.

Field	Note
<code>disk[]</code>	Drive, total, free, used %. Uses the <code>wmic</code> command; on Windows 11 24H2 and later <code>wmic</code> is removed and the field contains <i>Could not retrieve disk info</i> . Check disk space in Explorer instead.
<code>websockets</code>	Connected clients on HTTP and HTTPS.
<code>tagUpdateCycle</code>	Samples, average, minimum, maximum and last cycle time (ms).
<code>plcConnections</code>	Connection statistics from the communication manager.

29.6.1 Typical thresholds

Measurement	Good	Investigate
Free disk on the data drive	> 20 %	< 10 % — purge, move archive, extend disk
Heap used	stable	grows > 20 % per day
Read cycle time	< ½ of the fastest scan rate	> fastest scan rate
WAL size	< 50 MB	> 500 MB
WebSocket clients	= number of open stations	many more (stations with several tabs, reconnect loops)

29.7 Start-up messages

After a start the console log shows the start sequence. The key lines:

Message	Meaning
[Project] Active: <name> (<folder>)	Project and folder in use (the "many databases" check, Chapter 26).
[Project] WARNING: Directory not found: ..., falling back to Default	<code>active-project.json</code> points to a missing folder; the Default project is loaded instead.
[SECURITY] Generated new JWT secret key / ... HMAC secret key ...	First start, or the secret files were deleted (Chapter 25).
[SSL] HTTPS server created with certificates from ...	HTTPS active on 3443.
OptiZeus Server running on <code>http://localhost:3004</code>	Web server listening.
OptiZeus HTTPS Server running on <code>https://localhost:3443</code>	HTTPS listening.
License hardware ID cached in NNNN ms	Licence hardware probe completed (Chapter 30, slow login).
[Auto Backup] Scheduler started	Automatic backup check active.
[Audit] Hash chain initialized: seq=n	Audit trail chain continues from record n.
[Historian Worker] Ready, [Historian Worker] Archive path: ...	File historian worker running.

Message	Meaning
[EIP Worker] Connected to <PLC> (<id>)	EtherNet/IP worker connected.
[Redundancy] Heartbeat monitoring started for <url>	Redundancy active.

29.8 Background workers and communication diagnostics

Two worker threads run beside the main server thread so that slow devices do not block the user interface.

29.8.1 EtherNet/IP worker

Reads Allen-Bradley Logix and MicroLogix controllers.

Console message	Meaning
[EIP Worker] Connected to <name> (plc_<id>)	Connection established; PLC marked connected.
[EIP Worker] Scan complete for <name>: g good, b bad out of n tags (ms). Now using batch reads.	First scan finished; bad tags have addresses the controller rejected — check them.
[EIP Worker] plc_<id>: u/n tags in ms (method, b bad cached)	Periodic cycle statistics.
[EIP Worker] Disconnected from <name>: <error>	Connection lost; Marked n tags BAD follows and screens show XXXX .
[EIP Worker] Worker exited with code c	The worker stopped; it is restarted automatically after 5 s.

The worker reads its own list of addresses. Since 2.6.1.23 the list is re-sent automatically about 2 seconds after a tag's address or connection is edited; with older versions use **Refresh Tags** on the PLC connection or restart the service (Chapter 30).

29.8.2 Historian worker

Buffers and compresses values of historian-enabled tags and writes .hda archive files (every 5 s or 500 values).

Console message	Meaning
[Historian Worker] Starting historian worker thread, Ready	Running.
[Historian Worker] Flushed n records for t tags in ms	Normal writing.
[Historian Worker] Stats: received=..., stored=..., compressed=..., buffered=...	Periodic statistics; compressed values were dropped by the compression dead-band (normal).
[Historian Worker] Error: ...	Write problem (disk full, path not accessible).
[Historian Worker] Exited with code c	Restarted after 5 s.

GET /api/historian/worker-status answers {"workerActive": false, "fallbackMode": true} when the worker is not running; the main thread then records directly.

Note: Only values with **good** quality are sent to the historian worker. A tag whose PLC is disconnected produces a gap in the history, not a flat line — this is intended.

29.8.3 OPC UA connections

GET `/api/opcua/stats` lists, for each OPC UA server: the endpoint, `batchCapability` (whether multi-node reads work), whether a subscription is active, the number of monitored items and pending nodes, the age of the oldest cached value, the time since the last activity and whether a reconnect is in progress. A large `oldestCachedAgeMs` with `active: true` means the server stopped sending changes for some items.

29.9 Collecting data for a support case

1. Note the exact time of the problem and what the user saw (screenshot).
2. Record the version (29.4) and the active project (`/api/diagnostics`).
3. **Config** → **Admin** → **System Logs** → **Event Logs**: set the time range, **Export CSV**.
4. **Console Logs: Error Log, Last 1000, Export Log**; do the same for **Output Log**. If the problem is older than the displayed lines, copy the rotated files from `backend\daemon\`.
5. **Performance**: screenshot of all panels.
6. Describe the PLC connections involved (protocol, number of tags, scan rates).
7. Do **not** send `scada.db`, `.jwt-secret`, `.hmac-secret`, certificates or licence files. If the support engineer needs the configuration, send a configuration backup only after removing users and passwords from it, or agree a secure transfer.

30. Troubleshooting

This chapter lists the problems that are reported most often, in the form *symptom* → *cause* → *fix*. Each entry has been checked against the 2.6.1.26 code; where a problem was fixed in a specific build, the build is named so that you can tell whether an installation is affected. It is written for administrators and support engineers; operators should read Section 30.1 and call their administrator for anything else. The diagnostic tools referred to here are described in Chapter 29.

Before working through a table:

1. Note the exact time and the exact message or screenshot.
2. Check the version (`http://<server>:3004/api/update/current-version`).
3. Check which project is active (Chapter 29, `/api/diagnostics`) — many "impossible" findings come from looking at the wrong database (Section 26.10).
4. Look at **Config** → **Admin** → **System Logs** (event log and console log) for the same time.

30.1 Values show `xxxx`, `---` or tags are red

In synoptic screens a numeric display shows `xxxx` when its value cannot be trusted, and `---` when there is no value at all. In the Tag Monitor, tags with bad quality are shown in red.

Symptom	Cause	Fix
All values on the screen show <code>XXXX</code> , header shows <i>Communication Lost - Server disconnected</i>	The client lost its live connection to the server (network, server restart, server stopped). Displays switch to <code>XXXX</code> after 3 s without connection.	Check the network and the service (Chapter 26). The client reconnects automatically; values return within seconds after the server is reachable.
All values of one PLC show <code>XXXX</code> ; PLC connection red	The server lost the PLC (see 30.2). Tags of the PLC are set to bad quality.	Restore communication with the PLC.
Single tag shows <code>XXXX</code> ; others of the same PLC are fine	The PLC rejects the tag's address (wrong address, wrong data type, symbol does not exist, not in the optimized block / not accessible).	Check the address in Config → Tags and in the PLC program. For EtherNet/IP see the console line <code>Scan complete ... b bad</code> (Chapter 29).
A tag whose address was just edited stays red with the old value (EtherNet/IP)	Before 2.6.1.23 the EtherNet/IP worker kept polling the old address after a tag edit.	Upgrade to 2.6.1.23 (the worker is reconfigured automatically ~2 s after the edit). On older versions click Refresh Tags on the PLC connection or restart the service.
A new tag on an EtherNet/IP PLC is never read	Before 2.6.0.6 new tags were not added to the worker's list.	Upgrade, or Refresh Tags / restart.
Internal (memory) tag shows <code>XXXX</code> whenever a PLC drops	The tag still has a PLC connection or address from before it was switched to Memory .	Edit the tag, select Memory (this clears the PLC binding since 2.5.4.278) and save.
Value shows <code>---</code>	The tag exists but has never received a value (no value since start, or wrong tag binding).	Check the binding of the object (Chapter 12) and that the tag is being read.

Symptom	Cause	Fix
Value shows [Array]	The object is bound to a whole array tag.	Bind to an element, e.g. <code>Recipe_Active[3]</code> .
EtherNet/IP tags kept their last value after a PLC disconnect (looked live)	Fixed in 2.6.0.2; EtherNet/IP tags are now set to bad quality on disconnect, like OPC UA.	Upgrade.
A write shows Write Failed! <i>Not written: the PLC connection "<name>" is not connected or rejected the write.</i>	The PLC connection is down or not opened for writing (EtherNet/IP: ► Connect), the PLC refused the value (range, data type, write protection), or the protocol has no write driver. Since 2.6.1.24 such a write fails instead of being stored locally. The value was not changed.	Check the connection status and the event <code>tag_write_failed</code> (Chapter 29); correct the connection or address and write again (Chapter 6, 6.11.3).
A write shows <i>Not written: the OptiZeus server is not reachable. Nothing was sent to the plant.</i>	The client could not reach the server (network down, service stopped). Since 2.6.1.24 such writes are refused and never queued (Chapter 27).	Restore the connection to the server, check the current value, then write again if it is still needed.
Saving a tag or datalog, or Test / Refresh Tags on a PLC connection, answers <i>Insufficient privileges</i>	Since 2.6.1.24 these requests require the <code>admin</code> role; PLC connect/disconnect requires <code>operator</code> or <code>admin</code> (Chapter 24).	Log in with an administrator account; give integration accounts the role they need.

30.2 PLC shows disconnected

Symptom	Cause	Fix
Connection red from the start, test fails	IP address, port or network wrong; PLC firewall; PLC switched off.	<code>ping</code> the PLC from the server; test the port with <code>Test-NetConnection <ip> -Port <port></code> (S7 102, Modbus TCP 502, EtherNet/IP 44818, OPC UA 4840 or as configured). Correct the connection (Chapter 6).
S7: connection opens, reads fail	Wrong rack/slot, or the CPU does not allow external access (access protection, PUT/GET), or data blocks with optimized access.	Set rack/slot per CPU type; enable external access in the CPU properties; use non-optimized DBs or symbolic access via OPC UA (Chapter 6).
OPC UA: <i>Invalid OPC UA endpoint format</i>	The endpoint does not start with <code>opc.tcp://</code> .	Use e.g. <code>opc.tcp://192.168.1.10:4840</code> .
OPC UA: connected, but some values old	Subscription items not updating.	<code>GET /api/opcua/stats</code> (Chapter 29): check <code>oldestCachedAgeMs</code> , <code>pending</code> . After an OPC UA server upgrade call <code>POST /api/opcua/reset-capability/<serverId></code> , or disconnect/connect the connection.
Connection drops every few minutes	Network (switch, cable, Wi-Fi), PLC connection limit reached (too many clients), IP conflict.	Check the event log for the exact times; check the PLC's connection diagnostics; avoid duplicate IPs.

Symptom	Cause	Fix
EtherNet/IP: <i>Disconnected from <PLC>: ... repeatedly</i>	Controller or network interruptions. Since the 2.6.1.x builds a dropped socket no longer destabilises the server.	Fix the network; upgrade older installations.
All PLCs disconnect at the same moment	Server lost its network or the service restarted.	Check uptime in Performance and the console log around that time.
Communication alarm appears on reconnect as "cleared"	Normal: reconnect is logged as <code>plc_reconnected</code> and clears the communication alarm.	—

30.3 Login problems

Symptom	Cause	Fix
Login takes 13–15 s, first attempt fails and second works	Before 2.6.1.14 every licence status request re-probed the hardware with <code>wmic</code> ; Windows 11 24H2 removed <code>wmic</code> , so each probe fell back to slow PowerShell calls.	Upgrade to 2.6.1.14 or later: the hardware probe is cached for the life of the service and warmed at start-up (console: <code>License hardware ID cached in NNNN ms</code>). The first login within a few seconds after a restart can still be slower.
<i>Too many login attempts. Try again later.</i>	Login limit per client IP exceeded (default 5 in 15 min). Integrations that log in for every request also trigger it.	Wait for Lockout Duration (default 30 min) or restart the service (clears all lockouts). Fix the integration (Chapter 28).
<i>Invalid credentials</i> for a user who is sure of the password	Wrong project active (users are per project); Caps Lock; user exists only in another project.	Check the active project (Chapter 29). Reset the password (Chapter 24).
<i>Account is disabled. Contact administrator.</i>	Disabled manually or after 180 days without login.	Re-enable in Admin → Users .
<i>Invalid MFA code</i> although the code is fresh	Clock difference between server and phone > 30 s.	Synchronise the server time (NTP) and the phone time.
<i>Active Directory authentication is not enabled / LDAP connection error</i>	AD parameters not in the system settings (the Identity Providers panel does not configure the login in 2.6.1.26).	Configure as in Section 24.10.
Users are logged out after some time although they work	Session Timeout reached (no requests), or 24 h token life, or Auto-Logoff Timeout .	Adjust in Admin → Security (Chapter 24).
Everybody is logged out at once	User Sessions were purged, <code>.jwt-secret</code> was replaced, or the service was reinstalled.	Log in again.
Password expired dialog on every login	The password was changed on another project database, not the active one.	Change it again while logged in to the active project.

Symptom	Cause	Fix
Change the Default Password dialog after login	The account <code>admin</code> , <code>operator</code> or <code>viewer</code> still has its factory default password (since 2.6.1.24).	Set a new password in the dialog; the session continues only after the change.
SSO login refused: <i>The sign-in matches the existing password account ... Single sign-on does not log in to password accounts.</i>	The identity provider's user name or e-mail matches a local (password) or AD account. Since 2.6.1.24 SSO signs in only to accounts it created itself.	Sign in with the password account, or rename / remove the clashing local account so that SSO can create its own account (Chapter 24, 24.9).
Change Password on the login screen answers <i>Current password is incorrect</i> for a correct password	The User Name in the dialog is not the account whose password was entered (the field is filled from the login form).	Enter the right user name in the dialog.

30.4 Slow screens or configuration pages

Symptom	Cause	Fix
Config opens slowly, Viewer Builder list takes seconds	Before 2.6.1.16 the screen list transferred every screen's full content (over 1 MB on large projects).	Upgrade to 2.6.1.16 or later (list is ~5 KB).
Screens freeze on <i>Loading layout...</i> for ~12 s after a service restart	Before 2.6.1.0 the client always waited for the layout.	Upgrade; later versions render from the cached layout.
Operator actions answered <i>Too many requests</i>	More than 1,500 requests per minute from one IP (many stations behind one NAT address, or a runaway script).	Give stations their own addresses; fix the script.
High CPU on the server	Very fast scan rates on many tags, many open trend windows, heavy reports.	Check Performance read cycle time; relax scan rates (Chapter 6).

30.5 Reports and history

Symptom	Cause	Fix
Batch report of a long batch shows <i>No data</i> in charts and tables, short batches are fine	Before 2.6.1.10 the open report re-ran the full batch query every 3 s; on long batches the queries overlapped and exhausted the database connections.	Upgrade to 2.6.1.10 or later (polls cannot overlap; concurrent identical queries are merged). A failed element now shows <i>Failed to load data</i> with the reason instead of <i>No data</i> .
Report shows <i>Failed to load data</i>	The query itself failed (external database unreachable, time-out). The reason is shown under the message.	Check the data source (Chapter 19) and store & forward (Chapter 27).
Report or trend empty for a period	Datalog stopped; tag was bad quality (bad values are not recorded); time range/time zone wrong.	Check datalog status (Chapter 19), the event log for communication loss, and the selected range.

Symptom	Cause	Fix
Historian export via API returns HTTP 500 for alarms, or the Shift Report fails	Up to 2.6.1.25 <code>/api/export/alarms</code> , the Shift Report and scheduled reports read a table the schema does not contain.	Fixed in 2.6.1.26 — install the update (Chapter 2).

30.6 Service does not start or keeps restarting

Symptom	Cause	Fix
Service starts and stops immediately; Services shows <i>Stopped</i> or repeated restarts	Error during start-up.	Read <code>backend\daemon\optizeus.err.log</code> (end of file) and <code>optizeus.wrapper.log</code> .
Error log shows <code>EADDRINUSE ... :3004</code> (or <code>:3443</code>)	Another program uses the port (a second OptiZeus copy started by hand, a development server, IIS, another application). The server does not handle this error and exits; the wrapper restarts it after 5, 10 and 30 s.	Find the process: <code>`netstat -ano</code> findstr <code>:3004</code> , then <code>tasklist /FI "PID eq <pid>"</code> . Stop it, or change the port (30.7).
Cannot find module ...	Incomplete installation or patch (files missing, or patch copied to the wrong folder).	Re-run the patch installer as administrator; on Electron-hosted installations make sure <code>electron\resources\backend</code> was updated (Section 26.9).
Service does not exist in Services	Service was removed or never installed.	Run the patch installer (step 7/8 re-creates the service) or <code>SETUP.bat</code> .
[Project] WARNING: Directory not found ... falling back to Default	<code>active-project.json</code> points to a project folder that was moved or deleted.	Restore the folder or select the project again (Chapter 5). The Default project is running meanwhile — do not configure in it by mistake.
Error <code>SQLITE_BUSY</code> / database is locked at start	Another process holds the database (a second server instance, a database browser with an open transaction, antivirus scanning).	Close the other program; exclude the project folder from real-time antivirus scanning.
Patch installer says <i>Service did not enter RUNNING state</i> but OptiZeus works	Known issue: the installer checks the text <code>RUNNING</code> in the <code>sc query</code> output, which can misreport on Windows installations in other display languages.	Check in the Services console; if running, ignore the message.

Symptom	Cause	Fix
After a patch nothing changed	The running backend is <code>electron\resources\backend\</code> and a file was copied manually to <code>backend\</code> only; or the browser shows cached files; or the service was not restarted.	Copy to the running folder, restart the service, press Ctrl+F5 on clients (Section 26.9).

30.7 Port conflicts and firewall

Port	Used by	Change
3004/TCP	HTTP web server, REST API, WebSocket	Environment variable <code>PORT</code>
3443/TCP	HTTPS and secure WebSocket (only with certificate)	Environment variable <code>HTTPS_PORT</code>
4840/TCP	Built-in OPC UA server (if enabled)	OPC UA server settings (Chapter 8)

To change a port, add an `<env>` line to the service definition `backend\daemon\optizeus.xml`, e.g. `<env name="PORT" value="3005"/>`, then restart the service. Tell all clients the new address, and update the redundancy partner and multi-site entries.

Symptom	Cause	Fix
Server works on the server PC (<code>localhost</code>) but not from other PCs	Windows Firewall blocks the port. The installers do not create firewall rules.	Create inbound rules, e.g. <code>netsh advfirewall firewall add rule name="OptiZeus HTTP" dir=in action=allow protocol=TCP localport=3004 remoteip=192.168.10.0/24</code> (and 3443, 4840 as needed).
HTTPS page loads but live values do not update	Before 2.6.1.24: the secure WebSocket checked tokens with a different key (Section 25.3). From 2.6.1.24: port 3443 blocked for WebSocket traffic by a proxy or firewall, or the certificate not trusted by the browser.	Upgrade to 2.6.1.24. Otherwise open the browser console and check the <code>wss://</code> connection; log out and in again after a certificate change.
Browser warns about the certificate	Self-signed certificate or host name mismatch.	Use a certificate from the company CA with the correct host name.

30.8 Licence problems

The licence screens and messages are described in Chapter 2 and Appendix C.

Symptom / message	Cause	Fix
<i>Trial period has expired (7 days). Please activate a license.</i> — creating tags or screens, writing values and starting batches are refused (<code>TRIAL_EXPIRED</code>)	No valid licence after the 7-day trial.	Activate the licence file (Chapter 2).

Symptom / message	Cause	Fix
<i>License key is not valid for this machine. Hardware ID mismatch.</i>	Licence issued for another computer, or the hardware ID changed (disk, motherboard, CPU replaced; virtual machine moved).	Send the new hardware ID (License Activation page) to your supplier, or re-bind as described in Appendix C.6.3.
Hardware ID changed after a Windows upgrade to 24H2	Probe method changed when <code>wmic</code> disappeared (OptiZeus first tries <code>wmic</code> , then PowerShell).	Upgrade to 2.5.4.200 or later (PowerShell fallback) and contact the supplier if the ID differs.
<i>License has expired</i>	Time-limited licence.	Renew.
<i>This license has been deactivated and cannot be reused</i>	The licence was returned/deactivated on this machine.	Request a new licence.
<i>License signature is invalid or corrupted / License is corrupted or invalid</i>	File edited or damaged.	Re-copy the original licence file.
<i>Multi-server deployment requires Enterprise license</i>	Multi-Server page on a lower tier.	Enterprise tier or <code>multi-server</code> feature (Appendix C).
<i>Batch Control (S88) requires a Batch Control add-on license</i>	Add-on missing.	Order the add-on.
Limits reached (tags, PLC connections, screens, users, clients)	Tier limits.	See Appendix C; upgrade the tier.

30.9 Backup, maintenance and integration

Symptom	Cause	Fix
Backups folder keeps growing although Retention is set	Retention is <code>0</code> (keep all), or the server runs a version before 2.6.1.26, whose clean-up looked in <code>backend\backups</code> instead of the project folder. Manual backups are never purged.	Set a retention in days, or upgrade; delete old backups manually (Section 26.3).
No automatic backup on some days	The service was not running at Backup Time .	Choose a time when the server is always on; check the event log for <code>auto_backup</code> .
Auto-purge fields show 0 after reopening the page	Display issue (Section 26.7.3).	Re-enter all values before saving.
Verify Integrity reports the chain broken	Audit records were purged, or <code>.hmac-secret</code> was replaced, or the database was edited outside OptiZeus.	See Section 25.8; document as a deviation.
Webhook shows <i>Last Status</i> 400	Payload not valid JSON (quotes in the alarm message) or wrong template for the receiver.	Section 28.5.3.
Integration gets 401 after an hour	Session inactivity time-out.	Re-login on 401 (Section 28.2.3).

Symptom	Cause	Fix
An API request never answers	Up to 2.6.1.24 several routes sent no reply when the database returned an error. From 2.6.1.25 they answer HTTP 500 with the error text.	Upgrade; read the error text and the server console log (Chapter 29); set a client time-out.
Check for Updates says <i>No update server configured</i>	Known issue (Section 27.9).	Use the patch installer.

30.10 When to contact support

Contact your OptiZeus supplier when a problem persists after the checks above, when the service crashes repeatedly with the same error, or when data may have been lost. Prepare the information listed in Section 29.9.

31. Validation (GAMP 5)

This chapter explains how an OptiZeus system is validated for use in a GxP-regulated environment (pharmaceutical, biotech, medical device, food with regulatory oversight). It gives the GAMP 5 classification, lists the validation documents supplied with the product, outlines what the installation, operational and performance qualification (IQ/OQ/PQ) of *your* system must contain, and states which product functions support validation — and which gaps must be covered by procedure. It is written for validation engineers, quality assurance and the project engineer who prepares the qualification. Security and Part 11 functions are described in Chapter 25.

Important: Validation demonstrates that a **specific configured system** in its **specific environment** is fit for its intended use. The supplier documents described in 31.3 support that work; they do not replace it. Every change of version, configuration or infrastructure is subject to your change control and a documented impact assessment.

31.1 GAMP 5 classification

OptiZeus is classified as **GAMP 5 Software Category 4 — Configured Product**:

- it is a commercial product used without source-code modification;
- plant functionality is created by configuration (tags, alarms, screens, recipes, reports, users) through the user interface;
- standard interfaces are used for data acquisition (OPC UA, Modbus, Siemens S7, EtherNet/IP, MQTT).

Parts of a project can move towards **Category 5 (custom application)** and need more rigorous treatment:

Project element	Why	Approach
Scripts (Python event scripts, screen scripts, automation actions — Chapter 9)	Custom logic written by the integrator	Specify, review the code, test each script against its specification (Category 5 approach for that element).
Calculated tags with complex expressions	Custom calculation	Specify and test the expressions with boundary values.
Custom report layouts that compute results used for batch release	Custom calculation on GxP data	Verify calculations against independently computed values.
Integrations writing to the PLC (REST, ERP, webhooks)	Custom interface	Interface specification and test.

The underlying infrastructure (Windows, Node.js, SQLite, SQL Server if used) is Category 1.

31.2 Validation approach

A risk-based approach per GAMP 5 and ICH Q9:

1. **Planning** — Validation Plan / Protocol: scope, responsibilities, deliverables, acceptance criteria.
2. **Specification** — URS (your requirements), functional and configuration specification (what OptiZeus provides and how you configure it).
3. **Risk assessment** — functional risk assessment (FMEA) per GxP-relevant function; decides test depth.

4. **Supplier assessment** — use the supplier's documentation and test evidence (31.3) to reduce, not eliminate, your own testing.
5. **Verification** — IQ, OQ, PQ on the target system.
6. **Reporting** — Validation Summary Report, traceability matrix URS → risk → test.
7. **Operation** — change control, periodic review, backup/restore tests, access reviews, audit-trail review, incident management.

31.3 The supplied validation document set

The product documentation includes a GAMP 5 document set (Markdown and Word versions in the `docs` folder of the installation package):

Document	ID	Content	Use
GAMP5_URS — User Requirements Specification	URS-SCADA-001	86 numbered requirements (URS-001 ... URS-043b) (data acquisition, alarms, history, security, audit trail, reports...) with priority and GxP flag.	Template for your URS; delete what you do not use, add plant requirements.
GAMP5_FUNCTIONAL_SPEC	FS-SCADA-001	Functions by module (authentication, acquisition, alarms, history, screens, recipes, backup, audit).	Supplier FS; reference in your configuration specification.
GAMP5_DESIGN_SPEC	DS-SCADA-001	Architecture, database schema, security and data-integrity design.	Supplier DS.
GAMP5_RISK_ASSESSMENT	RA-SCADA-001	FMEA method, severity/probability/detectability scales, product-level risks and mitigations.	Starting point for your functional risk assessment.
GAMP5_VALIDATION_PROTOCOL	VAL-SCADA-001	Validation plan structure, classification, compliance matrix, IQ/OQ/PQ outline.	Template for your plan/protocol.
GAMP5_IQ	IQ-SCADA-001	37 installation checks (hardware, software, service, security configuration, drivers, documentation).	Template for your IQ.
GAMP5_OQ	OQ-SCADA-001	157 operational tests (authentication, sessions, RBAC, tags, alarms, notifications, audit trail, security, shelving, redundancy, historian, API...).	Template and supplier evidence.
GAMP5_PQ	PQ-SCADA-001	64 performance tests (response times, throughput, concurrent users, resources, stability, data integrity).	Template for your PQ.
GAMP5_VALIDATION_TEST_REPORT	VAL-SCADA-001-TR	Supplier test execution: 352 tests (IQ 23, OQ 157, PQ 64, backend unit 34, frontend unit 74).	Supplier test evidence.
GAMP5_VALIDATION_SUMMARY	VSR-SCADA-001	Supplier validation summary.	Supplier evidence.

Related documents: `CFR21_PART11_IMPLEMENTATION_GUIDE` , `SECURITY_AUDIT` , `QA_TEST_PLAN` , `QA_TEST_CASES` , and the release notes / changelog of each version.

Important — version of the supplier evidence. The supplier test report and summary were executed on version **2.5.4.106** (May 2026); the specifications carry document versions 2.0–2.2. The current product is **2.6.1.26**. Treat the supplier documents as templates and supporting evidence only, and execute your OQ on the version you install. Several product details have changed since those documents were written (for example the default HTTP port is 3004, not 3001 as in the IQ template; role and permission behaviour is as described in Chapter 24, not a five-role model). Correct the templates before use.

Note: The supplier OQ/PQ documents contain pre-filled results (`[X] PASS`) from the supplier's execution. Create blank copies for your own execution; never sign pre-filled results.

31.4 Installation Qualification (IQ) — outline

Objective: the system is installed as specified, on qualified infrastructure, with the correct software versions and configuration.

#	Check	Acceptance criterion	Evidence
IQ-1	Server hardware (CPU, RAM, disk, network adapters)	Meets the design specification	Screenshot of system information
IQ-2	Operating system version and patch level	As specified	<code>winver</code> , update history
IQ-3	Node.js version	As shipped with the installer	<code>node -v</code>
IQ-4	OptiZeus version	2.6.1.26 (or the version under validation)	<code>GET /api/update/current-version</code>
IQ-5	Installation folder and file integrity	Folder as specified; checksums of the release package recorded	Hash list of the ZIP
IQ-6	Windows service OptiZeus	Installed, start type Automatic, running	Services console
IQ-7	Ports	3004 (and 3443/4840 if used) listening; firewall rules as specified (installers do not create them)	<code>netstat</code> , firewall rule export
IQ-8	HTTPS certificate	Valid, correct host name, key protected	Certificate details
IQ-9	Licence	Activated, tier and add-ons as ordered	License Activation page
IQ-10	Active project	Correct project and database path	<code>/api/diagnostics</code>
IQ-11	Default accounts	<code>admin</code> password changed; default operator/viewer accounts removed or changed	Users page, event log

#	Check	Acceptance criterion	Evidence
IQ-12	Security settings	As in the configuration specification (password policy, lockout, session time-out, MFA, e-signatures, reasons)	Screenshot of Admin → Security
IQ-13	Time synchronisation	Server synchronised to the site NTP source	w32tm /query /status
IQ-14	Backup	Automatic backup configured; off-host copy configured	Backup page, task definition
IQ-15	External databases	Reachable; datalogs for alarms/events/audit configured if required	Datalog status
IQ-16	PLC connections	Addresses as per the I/O list	PLC Connections export
IQ-17	Secrets	.jwt-secret , .hmac-secret present, access restricted	File ACLs
IQ-18	Documentation	Manual, SOPs and configuration specification available	Document list

31.5 Operational Qualification (OQ) — outline

Objective: the configured functions work as specified across their operating ranges, including error cases. Prioritise by risk; the table lists the minimum for GxP-critical use.

Area	Test	Expected result
Authentication	Valid login; wrong password; disabled user; locked out after Max Login Attempts ; lockout expires	Messages as in Chapter 24; events logged
MFA	Enrol; login with valid/invalid code	Invalid code refused and logged
Password policy	Too short, not complex, reused password	Refused with the rule text
Session	Inactivity > Session Timeout	Next request requires login
Authorization	For each role, attempt every GxP function (write, acknowledge, recipe download, configuration) via the UI and via the REST API	Matches the role table of Chapter 24; document any function reachable by a lower role (see 31.7)
Security areas	Write from allowed and from disallowed IP/role	Allowed / refused with reason
Tag acquisition	Values of a sample of tags compared with the PLC / calibrated reference	Within tolerance; quality bad and XXXX on communication loss
Alarms	Each GxP alarm: activation at limit, delay, deadband, acknowledgment, return to normal, shelving	As configured; recorded with time and user
E-signatures	Acknowledge with correct / wrong password (REST and live-connection path); batch recipe approval	Wrong password refused and the alarm stays unacknowledged; signature record verifies (verified: true)
Reasons / confirmation	Write with reason and second-person confirmation	Event log contains reason and confirming user

Area	Test	Expected result
Failed writes	Write to a PLC tag with the PLC connection disconnected	<i>Write Failed!</i> shown; value unchanged; <code>tag_write_failed</code> in the event log
Audit trail	Create/modify/delete configuration items; login/logout; Verify Integrity	Records present with old/new values; <code>valid: true</code>
Historian / datalogs	Recording at configured rates; retrieval matches source; gap on communication loss	As specified
Reports	Content and calculations of each GxP report vs. independent calculation	Identical within rounding
Recipes / batch	Download, verify PLC values, batch record content	As specified
Backup / restore	Create, restore to a test system, compare	Configuration identical
Time	Record time stamps against NTP reference	Within ± 1 s
Failure	Stop service, PLC disconnect, database disconnect (store & forward)	Defined behaviour; no silent data loss

31.6 Performance Qualification (PQ) — outline

Objective: the system performs consistently in the real process under real load over a defined period.

Area	Example acceptance criterion
Screen update latency	Value change visible on the operator screen < 2 s (typical 0.5 s)
Alarm annunciation	Alarm visible < 2 s after the condition in the PLC
Historian completeness	≥ 99.9 % of expected samples over 7 days, gaps explained by recorded communication events
Concurrent operation	All stations connected, response times within criteria at peak (batch start, shift change)
Stability	30 days without unplanned restart; memory stable (Chapter 29)
Batch record	3 consecutive production batches with complete and correct records and signatures
Backup	Daily backups present for the whole PQ period; one restore test

31.7 What the product provides — and gaps to cover by procedure

Requirement area	Provided by OptiZeus 2.6.1.26	Gap / procedure needed
Unique user identification	Unique user names; one account per person supported	SOP: no shared accounts; disable, never reuse, names.
Access control	Roles checked by the server on configuration, write and acknowledgment requests (REST and live connection), security letters, security areas, IP rules; forced change of factory passwords	Since 2.6.1.24 tag, datalog and historian configuration and PLC connect/disconnect/test are role-checked and the live-connection acknowledgment follows the REST rules (Chapters 24, 25). Verify the role table in OQ via the UI and the REST API. SSO logins do not ask for the OptiZeus TOTP code; rely on the identity provider's MFA (Chapter 24).

Requirement area	Provided by OptiZeus 2.6.1.26	Gap / procedure needed
Custom roles	Can be defined	Cannot be assigned to users; use the three built-in roles.
Password and lockout policy	Complexity, history, expiry, lockout per IP, MFA	Global policy fields partly not applied to users (Section 24.6.1); verify effective values per user in OQ.
Audit trail	HMAC-chained, integrity check, export	Tag writes and several configuration changes are in the (non-chained) event log; audit purge possible by admins. Mitigate: forward events and audit to an external database, keep auto-purge 0, SOP forbidding purge without change control.
Electronic signatures	Alarm acknowledgment, batch recipe approval; bound by HMAC	Not available for recipe download, configuration changes, report approval. Use procedural signatures or login confirmation + reason for those.
Record retention	External database logging, historian archive, backups	Configuration backup excludes records; define file-level backups (Chapter 26).
Time stamps	Server clock, ISO/UTC in records	Qualify NTP.
Change control	Staging workflow, backup before change, event log	Formal change control is procedural. Multi-server "Deploy to Production" does not work in this version — promote by backup/restore with a change record (Chapter 27).
Business continuity	Redundancy heartbeat and manual failover, store & forward	No automatic data sync or PLC take-over; document the manual switch-over procedure.
Data integrity of exports	CSV/PDF exports, REST API	<code>/api/export/alarms</code> and the shift report derive alarm severity and message from the alarm type, value and limit (Chapter 21, Section 21.5); they work from 2.6.1.26.

31.8 Maintaining the validated state

Activity	Frequency	Reference
Change control for any configuration change (tags, alarms, screens, recipes, users' roles, security settings) with impact assessment and re-test	Each change	Chapters 5, 26
Version upgrades: read release notes, assess impact, test on a non-production system, execute regression OQ, update the configuration specification	Each patch	Section 26.9
Periodic review of the system (incidents, changes, deviations, audit trail, access)	Yearly	—
User access review (leavers, role changes, disabled accounts)	Quarterly	Chapter 24
Audit-trail review and Verify Integrity	Monthly or per batch release	Chapter 25
Backup restore test	At least yearly	Chapter 26
Time synchronisation check	Monthly	—

31.9 Worked example — validation deliverables for a small GMP line

Scope: one OptiZeus server, two operator stations, one PLC, 400 tags, 60 alarms, 5 recipes, a batch report.

Deliverable	Content	Based on
Validation Plan	Scope, Category 4 (+ Category 5 for 3 scripts), roles, acceptance	VAL-SCADA-001
URS	45 requirements selected from URS-SCADA-001 plus 12 plant-specific	URS-SCADA-001
Configuration Specification	Security settings, users/roles matrix, tag list, alarm list, recipes, report layout, backup settings	This manual
Functional Risk Assessment	38 functions, 9 high-risk (alarms on critical parameters, recipe download, batch report, audit trail)	RA-SCADA-001
IQ protocol and report	18 checks (31.4)	IQ-SCADA-001
OQ protocol and report	64 tests, full testing for high-risk functions, including the gap tests of 31.7	OQ-SCADA-001
PQ protocol and report	3 batches + 30-day stability	PQ-SCADA-001
Traceability Matrix	URS → risk → FS/CS → test	—
Validation Summary Report	Results, deviations, residual risks, release statement	VSR-SCADA-001
SOPs	Access management, backup/restore, change control, audit-trail review, incident handling, manual failover	—

OPTIZEUS SCADA · OPERATION MANUAL

Appendices

Appendix A. Keyboard shortcuts and touch operation

This appendix lists every keyboard shortcut that OptiZeus actually implements, the mouse modifiers used in the editors, and the on-screen keyboards and kiosk settings used on touch-panel operator stations. Engineers need sections A.1 to A.4 when they build screens. Supervisors and administrators who commission operator panels need sections A.5 to A.8.

Note: The built-in **Help** view and the guided tour mention some shortcuts that the software does not implement: Ctrl+S (save), Ctrl+H (help), Ctrl+Shift+A / C / H (jump to Alarms / Configuration / Historian), Ctrl+G (group) and Ctrl+F (search). The tables in this appendix list only the shortcuts that work. Save a screen with the **Save** button. Group objects with the toolbar group button.

A.1 Conventions

Notation	Meaning
Ctrl+Z	Hold Ctrl and press Z. On macOS clients, Cmd (⌘) works wherever the table says "Ctrl/Cmd".
Shift+Arrow	Hold Shift and press any arrow key.
Alt+click	Hold Alt and click with the mouse, or tap if the panel sends Alt.
"blocked in inputs"	The shortcut does nothing while the cursor is in a text box, number box or drop-down list, so that typing is not interpreted as a command.

Letter shortcuts are not case-sensitive. Caps Lock does not change them.

A.2 Synoptic screen editor (Viewer Builder)

These shortcuts apply in **Config** → **Viewer Builder** while a screen is open for editing.

A.2.1 Keyboard

Keys	Action	Notes
Ctrl/Cmd+Z	Undo	Up to 50 steps. Blocked in inputs.
Ctrl/Cmd+Y or Ctrl/Cmd+Shift+Z	Redo	Blocked in inputs.
Ctrl/Cmd+C	Copy the selected object(s)	If a selected object belongs to a group, the whole group is copied. The copy is also kept in the browser, so you can paste it on another screen.
Ctrl/Cmd+V	Paste	The first paste is offset +30 px right and down. Each further paste cascades another +30 px. Pasted groups get new group identities.
Ctrl/Cmd+D	Duplicate the selection	The duplicate is offset +20 px. This shortcut also fires while the cursor is in an input box.

Keys	Action	Notes
Ctrl/Cmd+A	Select all objects on the screen	Blocked in inputs.
Delete or Backspace	Delete the selected object(s)	Blocked in inputs. Use Ctrl+Z to restore.
Arrow keys	Move the selection 1 px	Ignores snap-to-grid. Each nudge is one undo step.
Shift+Arrow	Move the selection 10 px	

A.2.2 Mouse modifiers

Action	Result
Click a grouped object	Selects the whole group.
Ctrl/Cmd+click	Adds the object to, or removes it from, the multi-selection.
Alt+click	Cycles through objects that are stacked under the cursor, top to bottom. Use it to reach an object hidden behind another one.
Alt+Shift+click	Cycles through stacked objects in the opposite direction.
Ctrl-drag	Moves only the clicked objects without pulling in the other members of their group.
Double-click an object	Opens the Process Viewer editor, the state-text editor, or the faceplate, depending on the object type and its Click Action .

A.2.3 Tag Monitor in the editor

Keys	Action
Enter in an inline value cell	Writes the value to the tag. The audit trail records the reason as "Tag Monitor edit".
Escape , or clicking elsewhere	Cancels the edit.

WARNING: The Tag Monitor writes straight to the PLC. Pressing Enter in a value cell of a live project changes the plant. Test on a simulated or disconnected PLC first.

Zoom (0.25× to 3×, in 0.25 steps), grid display, snap-to-grid (default grid 20 px) and grouping are available only from toolbar buttons. There are no keyboard shortcuts for them.

A.3 Super object and faceplate builders

A.3.1 Super Object Builder

Keys	Action
Ctrl/Cmd+Z	Undo (up to 50 steps)

Keys	Action
Ctrl/Cmd+Y or Ctrl/Cmd+Shift+Z	Redo
Ctrl/Cmd+C	Copy objects together with their animations
Ctrl/Cmd+V	Paste (+30 px, cascading). Animations are re-linked to the new objects.
Ctrl/Cmd+D	Duplicate (+20 px)
Ctrl/Cmd+A	Select all
Delete / Backspace	Delete the selection
Arrow keys / Shift+Arrow	Move 1 px / 10 px
Ctrl/Cmd+click	Toggle an object in the multi-selection
Shift+click	Range-select, in object-list order

The builder shows a reminder line: "Shortcuts: Ctrl+C/V/D, Ctrl+Z/Y, Del, Arrow Keys". The builder zooms in 0.1 steps (0.25× to 3×), and snap-to-grid is on by default with a 10 px grid.

Important: In the Super Object Builder, shortcuts stay active while you type in the **Properties** panel. Pressing **Backspace** or **Delete** in a Properties field to erase text also deletes the selected object. Select the text and type over it, or use **Ctrl+Z** immediately if an object disappears.

A.3.2 Faceplate editor

The faceplate editor opens from the Super Object Builder or from the Viewer Builder. These shortcuts use the Ctrl key only (not Cmd) and are blocked in inputs.

Keys	Action
Ctrl+C / Ctrl+V	Copy / paste elements
Ctrl+Z / Ctrl+Y (or Ctrl+Shift+Z)	Undo / redo (up to 50 steps)
Ctrl+D	Duplicate
Ctrl+A	Select all
Delete / Backspace	Delete selected element(s)
Arrow keys	Move by one grid step (10 px by default) when snap is on, or 1 px when snap is off
Shift+Arrow	Move 10 px

Positions are never allowed below 0.

A.4 Other editors

A.4.1 Layout Builder (dashboard layouts)

Keys	Action	Notes
Ctrl/Cmd+Z	Undo	Blocked in inputs

Keys	Action	Notes
Ctrl/Cmd+Y or Ctrl/Cmd+Shift+Z	Redo	Blocked in inputs
Delete / Backspace	Delete the selected panel	Blocked in inputs
Ctrl/Cmd+D	Duplicate the selected panel	Also fires in inputs

There is no copy/paste and no arrow-key nudging in the Layout Builder.

A.4.2 Visual Report Builder

Keys	Action	Notes
Ctrl/Cmd+Z	Undo	Also works inside input boxes. Ctrl+Shift+Z also undoes; it does not redo.
Ctrl/Cmd+Y	Redo	
Ctrl/Cmd+C / Ctrl/Cmd+V	Copy / paste report elements	Not in text inputs
Delete / Backspace	Delete the selected element(s)	Not in text inputs
Escape	Clear the selection	
Ctrl/Cmd+click	Toggle an element in the multi-selection	

A.5 Runtime screens and dialogs

A.5.1 Dialogs

Dialog	Enter	Escape
Login	Logs in (form submit)	—
Login, server set-up screen	Test Connection (from the Server IP field)	—
Enter Value (write a tag from a synoptic screen)	Writes the value	Cancels
Confirm Your Identity (re-authentication before a protected action)	Submits (from the password field)	—
Change Password	Submits	Closes, unless the change is forced at login
Electronic signature	Sign , if a password has been entered	Cancels
MFA setup	Verify , when the code has 6 digits	Closes, unless MFA is mandatory
Tag faceplate	Writes the value in the set-point field	Closes the faceplate
AI Assistant	Sends the message (Shift+Enter inserts a new line)	—

Dialog	Enter	Escape
KPI dashboard rename	Confirms the new name	—
Historian template name	Saves the template	—
Create New Folder (Config → Tags)	Creates the folder	Cancels
Tag monitor inline edit (Config)	Writes the tag	Cancels
Alarm groups, alarm areas, predefined reasons (Config → Alarms)	Adds the entry typed in the box	—
New project (Config → Projects)	Creates the project	—

There is no Escape handler for synoptic pop-up windows. Close them with their close button, or with an object whose **Click Action** is **Close Popup**.

A.5.2 Trend charts

Where	Input	Result
History trend element on a synoptic screen	Drag across the chart	Zooms the time axis to the dragged range. The value axis fits the data with 10 % padding.
	Mouse wheel	Zooms about the centre (×1.2 out, ×0.8 in). Zooming out past the full data range resets the zoom.
	Double-click, or Reset Zoom	Resets the zoom
	1H 6H 12H 24H 7D 30D buttons	Selects the time window
Historian view	Drag across the chart	Zooms. Each pane has its own zoom.
	Double-click, or Reset Zoom	Resets the zoom
	Range slider below the chart	Scrolls and zooms the time window

The Historian view does not zoom on the mouse wheel.

A.5.3 Dashboard gauges

Action	Result
Click a tag gauge	Opens the tag faceplate. The click waits 250 ms so that a double-click can replace it.
Double-click a tag gauge (administrators only)	Opens the tag configuration

The gauge hint reads "Click: info | Double-click: configure".

A.5.4 Activity and auto-logout

Any key press, mouse movement, click, scroll or touch resets the auto-logout timer (see Chapter 24 for the timeout setting).

A.6 Touch-screen operation

OptiZeus has no multi-touch gestures of its own: pinch-zoom, pan, long-press and double-tap are not handled by the application. On a touch panel:

- A tap is a click.
- A double-tap works only where the browser turns it into a double-click, for example to reset a trend zoom.
- Pinch-to-zoom acts on the whole browser page, because the page does not disable browser zoom. In a locked-down kiosk, disable pinch zoom in the browser or the Electron client if operators must not change the page scale.
- The KPI dashboard slider writes its tag when you lift your finger, not while you drag.

A.6.1 Enabling touch-screen mode for a station

Touch-screen mode makes OptiZeus open an on-screen keyboard whenever the operator must type a value. It is set **per client station** on the server.

To enable touch-screen mode for an operator panel:

1. Start the panel once and log in, so that it registers as a client.
2. On an engineering station, open **Config** → **Admin** → **Clients & Shutdown**.
3. Select the **Connected Clients** sub-tab.
4. In the **Registered Clients** table, find the panel by **Client Name**, **Client ID** or **IP Address**.
5. Tick its **Touch Screen** check box. The message "Touch screen enabled for <name>. Client must refresh to apply changes." appears.
6. Refresh the browser on the panel, or restart the Electron client.

The server setting is sent to the client when it registers and again with every 60-second heartbeat. It overwrites any local setting.

Note: The **Electron Client** page (see A.8) also has a check box, **Enable Touch Screen Mode (shows on-screen keyboard for value entry)**. It changes only the station where you click it, and the next heartbeat overwrites it with the server setting. Use the **Touch Screen** column in **Registered Clients** instead.

On a device that reports a touch screen, the login page offers the on-screen keyboard even when touch-screen mode is off. There is also always an  **On-Screen Keyboard** button on the login page.

A.7 On-screen keyboards

OptiZeus has two different on-screen keyboards.

A.7.1 Touch keyboard (value entry and login)

This keyboard opens in touch-screen mode for:

- the login page;

- **Write Tag Value** click actions and faceplate set-value buttons on synoptic screens;
- the **Confirm Your Identity** dialog;
- recipe parameter values in the Recipe Manager.

It does **not** open for the Change Password dialog or the electronic-signature dialog. Connect a physical keyboard for those, or change passwords from an engineering station.

Layout	Used for	Keys
Numeric	Numeric tags and parameters	7 8 9 / 4 5 6 / 1 2 3 / 0 . -
Alphanumeric	STRING tags, user name, password	Digits 1-0, the letters q-p, a-l, z-m, . and -

Control keys:

Key	Action
⌵ Caps	Upper case on or off (alphanumeric layout only)
Space	Inserts a space
⌫ Back	Deletes the last character
Clear	Empties the field
✓ Done	Closes the keyboard and applies the value
✕ (next to the display)	On a synoptic screen: cancels (nothing is written). Elsewhere: closes the keyboard and keeps what was typed

The keyboard has no symbol page and no Hebrew or Arabic layout. Type names and passwords that need other characters on a physical keyboard.

How it behaves:

- **The first key replaces the value.** When the keyboard opens on an existing value, the first key you press replaces the whole value, as in a spreadsheet cell. After ⌫ **Back** or **Clear**, keys are appended.
- **Range display.** For numeric tags the header shows "Min: x Max: y" and the unit. A value outside the range turns red with "OUT OF RANGE", but the keyboard does not stop the write.
- **Done writes, ✕ cancels (synoptic screens).** When the keyboard was opened on a synoptic screen (for example by a Write Tag action), only ✓ **Done** or **Enter** writes the value on the display (if it differs from the current value). ✕ and tapping the dark area outside the keyboard cancel the entry: nothing is written. In all other fields (login, dialogs, forms) ✓ **Done**, ✕ and a tap outside all close the keyboard and keep what was typed. (Changed in 2.6.1.26; up to 2.6.1.25 ✕ and a tap outside also wrote the value on synoptic screens.)
- **Moving the keyboard.** The keyboard opens at the bottom centre of the screen (full width, at most 800 px). Drag it by its header or by the ≡ bar to uncover the field underneath. The position is not remembered.
- **Physical keyboard.** The keyboard's display field has focus, so a USB keyboard can type into it. **Enter** submits on the login page and closes the keyboard elsewhere.

WARNING: ✓ **Done** and **Enter** write immediately. Train operators to check the display before pressing ✓ **Done**, and to use ✕ to abandon an entry. For critical set-points, protect the object with a security letter or an electronic signature (Chapter 25).

A.7.2 Input Value keyboard (synoptic input fields)

An **Input Value** object on a synoptic screen can carry its own small keyboard. Configure it in the object's properties in the Viewer Builder or Super Object Builder:

Field	Options	Default
Enable Touch Keyboard	On / off	Off
Keyboard Position	Below, Above, Left, Right, Screen Center	Below
Keyboard Type	Numeric (0-9): 7 8 9 4 5 6 1 2 3 . 0 C Numpad (with +/-): adds + - * / Full Keyboard: digits, letters, . , - , DEL, C, SPACE	Numeric

This keyboard has **Cancel** and **Enter** buttons. **C** clears the field. The first key replaces the old value. On **Enter** the value is converted to a number. Text that is not a number is ignored. If the object has a minimum and maximum, a value outside the range is refused with "Value Out of Range!". This keyboard cannot be dragged.

A.8 Kiosk operation

Kiosk operation keeps an operator on the SCADA screen and away from the Windows desktop. OptiZeus has three separate mechanisms. Only the first two are enforced.

A.8.1 Kiosk role (browser and desktop clients)

This is the mechanism that works for any client, browser or Electron.

To make a role run in kiosk mode:

1. Open **Config** → **Admin** → **Security**.
2. In **Role Management & Security Letters (A-Z)**, edit the role, for example `operator`.
3. Switch on **Kiosk Mode**. The description reads "When enabled, users with this role will run in fullscreen locked mode and cannot exit or access desktop".
4. Save the role.

When a user with a kiosk role logs in:

- the page switches to full screen 0.5 s after login;
- while in full screen, these keys are blocked: **Escape**, **Alt+F4**, **Ctrl+W**, **Ctrl+Q**, **F11**, the Windows key and, as far as a browser allows, **Alt+Tab**;
- the right-click menu is disabled;
- closing or reloading the page asks "Are you sure you want to leave? You are in kiosk mode.";
- if full screen is lost anyway, a full-screen **KIOSK MODE** overlay covers the application with a button to re-enter full screen, and the application also retries full screen by itself;
- if the window loses focus, OptiZeus takes the focus back.

Note: A web browser can never fully block Alt+Tab, Ctrl+Alt+Del or the Windows key. For a sealed panel, combine the kiosk role with the Electron client lock described in A.8.2.

A.8.2 Electron client configuration

The Electron desktop client reads its settings from `electron-config.json` in the backend folder (see Appendix B). Edit them in **Config** → **Admin** → **Clients & Shutdown** → **Electron Client**. Changes take effect after the Electron application is restarted.

Section	Field	Effect	Default
Window Settings	Width (px), Height (px), Min Width (px), Min Height (px), quick presets (HD to 4K, Auto-Detect Current Display)	Window size	1600 × 1000, minimum 1024 × 768
	Start in Fullscreen	Full screen at start	Off
	Start Maximized (recommended if not using fullscreen)	Maximised at start	On
Kiosk Mode	Enable Kiosk Mode	Frameless, full-screen kiosk window; the menu is removed	Off
	Exit Password	See the note below	Empty
	Exit Key Combination	Global shortcut that exits kiosk mode	Ctrl+Shift+Q
Menu Bar	Show Menu Bar	Shows the standard menu. Its default accelerators then apply: Ctrl+R reload, Ctrl+Shift+I developer tools, Ctrl+O / Ctrl+= / Ctrl+- zoom, F11 full screen.	On
	Show DevTools Option	Adds Open DevTools to the tray menu	Off
System Tray	Enable System Tray Icon, Minimize to Tray on Close, Show Notification Balloons	Closing hides the window to the tray; double-click the tray icon to restore it	All on
Startup	Auto-start with Windows	Adds the Run registry value <code>OptiZeusSCADA</code> for the current user	Off
	Splash Screen Duration (ms)	0–10000	1500
	Default Screen	Stored but not used by the current client	Dashboard
Security	Allow Opening External Links		Off
	Disable Right-Click Context Menu		Off
	Disable Browser Keyboard Shortcuts (F5, Ctrl+R, etc.)	Blocks F5, Ctrl+R and Ctrl+Shift+R	Off
	Prevent Application Close (blocks close button, Alt+F4)	Blocks closing; also blocks Alt+F4, Windows-key combinations, Alt+Tab and Ctrl+Esc	Off
	Lock Desktop (blocks Win key, Alt+Tab, taskbar access)	See below	Off
Interface Settings	Enable Touch Screen Mode (shows on-screen keyboard for value entry)	Local only; see A.6.1	Off

Lock Desktop does the following:

- disables the Windows key for the current user through a policy value and a keyboard hook, and restarts Explorer;
- hides the taskbar;
- keeps the window always on top and out of the taskbar, and takes focus back if it is lost;
- swallows Win+D, Win+E, Win+R, Win+L, Win+Tab, Alt+Tab, Alt+Shift+Tab, Alt+Esc, Ctrl+Esc, Ctrl+Shift+Esc and Alt+F4;
- swallows F1, F2, F3, F4, F10, F11 and F12, alone and with Alt.

When the client exits normally, the Windows key and the taskbar are restored.

Leaving kiosk mode. Press the **Exit Key Combination** (default **Ctrl+Shift+Q**).

- If no exit password is set, the client quits at once.
- If an exit password is set, a dialog titled "Exit Kiosk Mode" appears with **Cancel** and **Exit**.

Important: In version 2.6.1.26 the exit dialog does not ask for or check the password. Anyone who knows the key combination can press **Exit**. Change the **Exit Key Combination** to something only engineers know, and do not rely on the exit password as a security control. The password is also stored in plain text in `electron-config.json`.

On every start the Electron client clears the stored login, so the operator must log in again. It also keeps the display from going to sleep, and it reloads the page automatically if the page crashes or hangs.

A.8.3 Browser Client page

Config → **Admin** → **Clients & Shutdown** → **Browser Client** has:

- a **Fullscreen Mode** button;
- **Lock Fullscreen**;
- **Enable Kiosk Mode**.

In version 2.6.1.26 the lock and kiosk switches on this page are only remembered by the browser and shown as status. They do not enforce anything. Use the kiosk **role** (A.8.1) to enforce kiosk behaviour.

A.8.4 Windows lock-down scripts

The installation folder contains helper scripts for a dedicated kiosk PC. Run them as an administrator, and only on a PC that is used only as an operator panel.

Script	Purpose
<code>START-SCADA-KIOSK.bat</code>	Sets current-user policies: no Windows key, Task Manager disabled, lock workstation disabled, change-password disabled, log-off removed, fast user switching hidden. Restarts Explorer, then starts the desktop client.
<code>STOP-SCADA-RESTORE.bat</code>	Removes those policies and restarts Explorer. It also ends every node.exe process on the PC, including the OptiZeus server. Restart the OptiZeus service afterwards (Appendix B).
<code>electron\apply-security-now.bat</code> / <code>electron\restore-security.bat</code>	Apply or remove the same policies without starting or stopping anything
<code>electron\test-winkey-disable.bat</code> / <code>electron\test-winkey-enable.bat</code>	Disable or enable only the Windows key

Script	Purpose
REMOVE-AUTOSTART.bat	Removes the older auto-start registry entries named "OptiZeus SCADA", "OptiZeus" and "OptiZeus SCADA Electron"

Important: In version 2.6.1.26: - START-SCADA-KIOSK.bat contains a fixed development path and starts the client with `node` instead of the Electron runtime. Check it before use, or start the client with the desktop shortcut. - REMOVE-AUTOSTART.bat does not remove the `OptiZeusSCADA` value that **Auto-start with Windows** creates. Switch that option off in the Electron Client page instead.

A.9 Quick reference card

Task	Keys
Undo / redo in any editor	Ctrl+Z / Ctrl+Y
Copy, paste, duplicate on a screen	Ctrl+C, Ctrl+V, Ctrl+D
Select all objects	Ctrl+A
Delete selection	Delete
Nudge 1 px / 10 px	Arrow / Shift+Arrow
Reach a hidden object	Alt+click (Alt+Shift+click backwards)
Add to selection	Ctrl+click
Confirm a dialog / write a value	Enter
Cancel a dialog	Escape
Reset trend zoom	Double-click the chart
Leave Electron kiosk mode	Ctrl+Shift+Q (or the configured combination)

Appendix B. Ports, services and files

This appendix lists the network ports OptiZeus uses, the Windows service and start-up entries it creates, and the folders and files of an installation: configuration files, logs, databases and certificates. System integrators need it to plan firewalls and back-ups. IT and OT administrators need it to maintain and troubleshoot a server.

The contents of secret files are never shown here. Treat every file marked "secret" in this appendix as confidential. Exclude those files from anything you send outside the site, such as support packages or copies of the project.

B.1 Listening ports

The OptiZeus server listens on the following ports.

Port	Protocol	Purpose	Notes
3004	HTTP and WebSocket (<code>ws://</code>)	The web interface, the REST API and the live-data connection of every client	The default. Change it with the environment variable <code>PORT</code> .
3443	HTTPS and secure WebSocket (<code>wss://</code>)	The same web interface and API over TLS	Started only if both <code>ssl-key.pem</code> and <code>ssl-cert.pem</code> exist in the backend folder. Change the port with <code>HTTPS_PORT</code> .
4840	OPC UA (<code>opc.tcp://</code>)	The built-in OPC UA server that publishes OptiZeus tags to other systems (Chapter 8). Endpoint path <code>/UA/OptiZeus</code> .	The port is configurable. It listens only after the OPC UA server is started from Config → OPC UA Server , or automatically after a service start when auto-start is configured (Chapter 8, since 2.6.1.24).

Ports 3004 and 3443 are two independent listeners for the same application. OptiZeus does not redirect HTTP to HTTPS by itself. A redirect happens only when a reverse proxy in front of OptiZeus reports that the original request was plain HTTP.

A browser client uses the same scheme for its live-data connection as for the page: `ws://` on 3004 for an `http://` page, `wss://` on 3443 for an `https://` page. If the live-data connection is not established within 5 seconds, the client tries the other one.

Note: Older documents refer to port **3002** as the "backend port". Version 2.6.1.26 does not use port 3002. Only 3004 and 3443 are used.

Other ports that appear on a development PC are not part of a production installation:

- 3003 is the development web server.
- 4000 belongs to a separate time-attendance application.

OptiZeus has no Modbus slave, EtherNet/IP adapter, DNP3 or BACnet server, MQTT broker, SNMP agent or syslog receiver. It does not listen on any other port. The EtherNet/IP and historian workers and the OPC alarm bridge are internal processes and do not open network ports.

B.1.1 Firewall

Important: The OptiZeus installers do not create any Windows Firewall rules. After installation, open the ports yourself on the server: TCP **3004** (HTTP), TCP **3443** (HTTPS) and, if the OPC UA server is used, TCP **4840**. Open only the ports you use, and allow them only from the operator and engineering networks. The uninstaller removes rules named "OptiZeus HTTP", "OptiZeus HTTPS" and "OptiZeus SCADA", so use these names when you create the rules.

To open the ports from an elevated command prompt:

```
netsh advfirewall firewall add rule name="OptiZeus HTTP" dir=in action=allow protocol=TCP
localport=3004
netsh advfirewall firewall add rule name="OptiZeus HTTPS" dir=in action=allow protocol=TCP
localport=3443
netsh advfirewall firewall add rule name="OptiZeus SCADA" dir=in action=allow protocol=TCP
localport=4840
```

Add `remoteip=<subnet>` to each rule to restrict it to the plant network, for example `remoteip=192.168.10.0/24`.

B.2 Outbound connections

OptiZeus opens connections to PLCs, databases and services. The table gives the default port that is used when the **Port** field of a connection is left empty. Firewalls between OptiZeus and these devices must allow the traffic.

Destination	Default port	Configured in
Siemens S7 (ISO-on-TCP)	TCP 102	PLC Connections (Chapter 6)
Modbus TCP	TCP 502	PLC Connections
EtherNet/IP, Allen-Bradley Logix, MicroLogix (PCCC)	TCP 44818	PLC Connections
OPC UA server (client connection)	TCP 4840	PLC Connections, Endpoint URL
MQTT broker	TCP 1883. With TLS the scheme is <code>mqtts</code> , but the port still defaults to 1883. Enter 8883 explicitly for a TLS broker.	PLC Connections, Sparkplug B, edge gateway
BACnet/IP	UDP 47808	PLC Connections
GE SRTP	TCP 18245	PLC Connections
Omron FINS	TCP 9600	PLC Connections
Mitsubishi MELSEC	5007 in the connection dialog	PLC Connections
Beckhoff ADS	TCP 48898, AMS port 851	PLC Connections
SMTP e-mail server	TCP 587	Alarm notification channels (Chapter 17)
SQL Server / MySQL / PostgreSQL / MongoDB / InfluxDB	1433 / 3306 / 5432 / 27017 / 8086	Data sources for datalogs and the historian (Chapter 19)

Destination	Default port	Configured in
GSM modem over Ethernet / Sierra Wireless / Teltonika, SMSEagle	5000 / 2332 / 80 or 443	GSM modem settings
Active Directory	389 (LDAP) or 636 (LDAPS), given in the server URL	Directory integration (Chapter 24)
AI providers, voice call service	TCP 443. A local Ollama AI server uses 11434.	AI Assistant, voice call-out
IP cameras	The camera's HTTP or HTTPS port. The MJPEG stream is relayed by OptiZeus to the clients; RTSP is not supported.	Cameras (Chapter 15)
Redundant, deployment and multi-site servers	The other server's web port (3004 or 3443)	Chapter 27

B.3 The Windows service

The full installation runs the OptiZeus server as a Windows service.

Property	Value
Service name	OptiZeus
Display name	OptiZeus
Description	OptiZeus Industrial Automation System
Wrapper	C:\OptiZeus\backend\daemon\optizeus.exe (a Windows service wrapper) with its configuration optizeus.xml in the same folder
Program	node.exe (the one found on the path at installation, otherwise C:\Program Files\nodejs\node.exe) running C:\OptiZeus\backend\server.js
Working folder	C:\OptiZeus\backend
Account	Local System
Start type	Automatic
Recovery	Restart after 5 s, 10 s and 30 s
Stop time-out	30 s
Environment	NODE_ENV=production

If the service wrapper cannot be installed, the set-up creates a scheduled task named **OptiZeus SCADA** instead. It runs at system start as SYSTEM and restarts up to three times at one-minute intervals.

To control the service:

- From Windows, use **Services** (`services.msc`), or `net stop OptiZeus` and `net start OptiZeus` in an elevated command prompt.
- From OptiZeus, open **Config** → **Admin** → **Clients & Shutdown**. The service-control area there has **Restart** and **Stop**. Restarting briefly disconnects all clients, which reconnect automatically. Stopping shuts the server

down completely; it must then be started again from Windows Services.

WARNING: Stopping the service stops all PLC communication, alarm handling and data recording. Do this only in a planned outage, and make sure the PLCs can run the plant safely without the SCADA.

Switching the active project (Chapter 5) also restarts the server: the process exits and the service wrapper starts it again.

B.3.1 Desktop client and a second server

The desktop client (Electron) is installed in `C:\OptiZeus\electron\` and contains its own copy of the server in `C:\OptiZeus\electron\resources\backend\`.

When the desktop client starts, it checks whether a server answers on `http://localhost:3004`:

- If one answers, the client uses it.
- If none answers, the client starts its **own** server from `electron\resources\backend\`.

That second server has its own active project, database and configuration files.

Important: On any given PC, find out which backend folder is actually running before you change files, apply a hot-fix or restore a database. Look at the command line of the `node.exe` process in Task Manager (**Details** tab, **Command line** column). A change made in the other folder has no effect. Patch installers update both folders.

B.3.2 Auto-start entries

Entry	Created by	Removed by
Service <code>OptiZeus</code>	Full set-up	Uninstaller
Scheduled task <code>OptiZeus SCADA (fallback)</code>	Full set-up, only if the service could not be installed	Uninstaller
Registry value <code>OptiZeusSCADA</code> under <code>HKCU\Software\Microsoft\Windows\CurrentVersion\Run</code>	Desktop client, when Auto-start with Windows is on	Switching the option off
Desktop shortcuts OptiZeus SCADA (browser launcher), OptiZeus SCADA (App) (desktop client) and Uninstall OptiZeus SCADA	Full set-up	Uninstaller

The set-up also adds the server's self-signed certificate to the Windows trusted root store of the server PC, so that `https://localhost:3443` opens without a warning there. Other client PCs show a certificate warning until the certificate is imported on them or replaced by a certificate from the site's own certificate authority (Chapter 25).

B.4 Folder structure

The full set-up always installs to `C:\OptiZeus`. The installation folder cannot be changed.

C:\OptiZeus\	
├─ license.dat, agreement.dat, grace-period.dat,	Licence files (Appendix C)
├─ license-return.dat, revoked-licenses.dat	
├─ install.log	Set-up log
├─ UNINSTALL.bat, icon.ico	
├─ client\optizeus-launcher.html	Browser launcher (tries HTTPS, then HTTP)
├─ electron\	Desktop client
├─ └─ OptiZeus SCADA.exe	
├─ └─ resources\	
├─ └─ └─ backend\	Server copy used by the desktop client
├─ └─ └─ frontend\	
└─ backend\	Server folder of the Windows service
├─ server.js, license.js, package.json	
├─ node_modules\	Libraries
├─ daemon\	Service wrapper, its configuration and logs
├─ public\	Web interface (compiled)
├─ services\ drivers\ middleware\ utils\ workers\	Program modules
├─ projects\<Project name>\	One folder per project (except Default)
├─ uploads\	Uploaded logos and images (/uploads)
├─ videos\	Tutorial videos (/videos)
├─ updates\	Downloaded update packages
├─ backups\	Automatic and historian backups
├─ store-forward-archive\	Store-and-forward queue
├─ whatsapp-session\	Linked WhatsApp session (secret)
├─ backup-pre-<version>\	Code saved by the patch installer
├─ scada.db (+ scada.db-wal, scada.db-shm)	Default project database and SVG library
├─ runtime\	Default project run-time state
└─ *.json	Server-wide and Default-project settings

B.4.1 Project folders

The project named **Default** lives directly in `backend\`. Every other project has its own folder `backend\projects\<Project name>\`. The file `backend\active-project.json` records which project is active.

A project folder contains:

Item	Content
<code>scada.db</code> , <code>scada.db-wal</code> , <code>scada.db-shm</code>	The project database (B.6)
<code>*.json</code>	Project settings (B.5)
<code>runtime\</code>	Run-time state (B.4.2)
<code>backups\</code>	Project backups (<code>scada-backup-*.zip</code>)
<code>historian-archives\<tag>\</code>	File-based historian archives, one file per tag per day, if file archive storage is enabled (B.6.2)
<code>store-forward\<datalog>\</code>	Datalog records waiting for an unreachable external database

Note: Back up the whole project folder, not only `scada.db`. With WAL journaling, recent changes can still be in `scada.db-wal`. Use **Config** → **Admin** → **Backup & Restore** (Chapter 26), which copies a consistent set, rather than copying the files while the service runs.

B.4.2 The runtime folder

`runtime\` holds state that the server writes while it is used, as opposed to engineered configuration:

File	Content
<code>client-settings.json</code>	Registered client stations: client ID, name, touch-screen mode, last seen
<code>notification-history.json</code>	Recent notifications (last 1,000)
<code>alarm-notification-log.json</code>	The alarm notification send log, when no external database is used (last 1,000)
<code>security-results.json</code> , <code>test-results.json</code>	Results of built-in security and self-tests

Deleting `runtime\` resets this accumulated state, for example all registered clients, without touching the engineered project. Stop the service first.

B.5 Configuration files

"Server" files are in `backend\` and apply to the whole server. "Project" files are in the active project folder, which is `backend\` itself for the Default project. Edit these files through the user interface wherever possible. Stop the service before editing a file by hand, and keep a copy.

File	Level	Written by	Content
<code>active-project.json</code>	Server	Projects (switch project)	Name and folder of the active project
<code>performance-config.json</code>	Server	By hand	<code>scanIntervalMs</code> , <code>broadcastIntervalMs</code> (defaults 1,000 ms, minimum 100 ms). It does not exist by default.
<code>redundancy-config.json</code>	Server	Redundancy settings	Primary and backup server, health-check interval, fail-over time-out, fail-back
<code>server-deployment-config.json</code>	Server	Deployment	Server role and topology (test, production, backup, historian, batch servers), deployment and sync options. Contains API keys (secret).
<code>service-config.json</code>	Server	Service settings	Service name, auto start, restart on crash, log rotation
<code>erp-config.json</code>	Server	ERP connector	ERP synchronisation and data mappings (secret if it holds credentials)
<code>sparkplug-config.json</code>	Server	Sparkplug B	Broker, group ID, credentials (secret)
<code>store-forward-config.json</code>	Server	Store-and-forward	Archive folder

File	Level	Written by	Content
deployment-sync-log.json	Server	Deployment	Deployment log
alarm-settings.json	Project	Alarms → Global Settings	Acknowledge reason rules and predefined reasons
alarm-notification-config.json	Project	Alarms → Notification	Channels (SMTP, WhatsApp, Telegram, GSM modems, voice), users, groups, areas, templates. Passwords and tokens are stored encrypted (secret).
notification-settings.json	Project	Legacy notification settings	Older notification configuration (secret)
security-settings.json	Project	Admin → Security	Password policy, lock-out, session time-out, electronic signatures, MFA, directory settings (secret: directory bind password)
roles.json	Project	Admin → Security	Roles, permissions, security letters, kiosk flag
layout-settings.json	Project	Display	Main screen layout and panels
electron-config.json	Project; the desktop client reads the copy in electron\resources\backend\	Admin → Clients & Shutdown → Electron Client	Window, kiosk, menu, tray, start-up and security options (Appendix A). The kiosk exit password is stored here in plain text.
ai-settings.json	Project	AI Assistant	AI providers, models and API keys (secret)
super-objects.json , super-object-instances.json	Project	Super Object Builder	Super-object and faceplate definitions
udts.json	Project	UDT / Super Tags	UDT definitions
areas.json	Project	Admin → Security → Areas	Plant areas
named-queries.json	Project	Admin → Named Queries	Saved SQL queries
historian-tiers.json	Project	Admin → Historian Tiers	Hot and warm retention (defaults 30 and 365 days)
oidc-config.json	Project	Admin → Security → Identity	Single sign-on settings, including the client secret (secret)

Note: The **Electron Client** page saves `electron-config.json` in the project folder, but the desktop client reads the copy in `electron\resources\backend\`. If a change made on the page has no effect on a desktop client, copy the file to that folder and restart the client.

B.5.1 Secret files and certificates

File	Location	Purpose
<code>.jwt-secret</code>	<code>backend\</code>	Signs log-in tokens and sessions, and is the base of the key that encrypts passwords in configuration files. Generated at first start.
<code>.hmac-secret</code>	<code>backend\</code>	Protects the integrity of electronic records (21 CFR Part 11)
<code>ssl-key.pem</code> , <code>ssl-cert.pem</code>	<code>backend\</code>	TLS key and certificate for port 3443. The set-up creates a self-signed certificate for <code>localhost</code> , valid for 10 years. Replace both files with the site's certificate and restart the service.
<code>whatsapp-session\</code> , <code>.webjbs_auth</code> , <code>.webjbs_cache</code>	<code>backend\</code> or the project folder	The linked WhatsApp session. Anyone who has these can send messages as the linked phone.
<code>license.dat</code> and companion files	Folder above <code>backend\</code>	Licence (Appendix C)

WARNING: Anyone who obtains `.jwt-secret` can forge log-in tokens for any user, including administrators. Restrict access to the `backend` folder to administrators and the service account. Never copy it into e-mails, tickets or support packages. If it has been exposed, stop the service, delete the file (a new one is generated at start) and restart. All users then have to log in again, and passwords stored encrypted in configuration files must be re-entered.

B.6 Databases

B.6.1 Project database

Each project has one SQLite database, `scada.db`, opened in WAL mode with a 5 s busy time-out. The main groups of tables are:

Area	Tables
Tags	<code>tags_config</code> , <code>tag_data</code> , <code>udt_definitions</code> , <code>udt_instances</code>
Connections	<code>plc_connections</code> , <code>opcua_servers</code> , <code>ethernetip_servers</code> , <code>opc_ae_servers</code> , <code>data_sources</code> , <code>gsm_modems</code>
Screens	<code>synoptic_screens</code> , <code>screen_folders</code> , <code>super_objects</code> , <code>super_object_instances</code> , <code>user_layouts</code> , <code>station_layouts</code> , <code>shared_dashboards</code>
Alarms	<code>alarm_config</code> , <code>alarms</code> , <code>alarm_history</code> , <code>sequence_of_events</code> , <code>alarm_pipelines</code> , <code>alarm_pipeline_executions</code> , <code>alarm_translations</code> , <code>notification_groups</code> , <code>notification_users</code>
Events and audit	<code>event_log</code> , <code>audit_trail</code> , <code>electronic_signatures</code> , <code>deploy_history</code> , <code>pending_changes</code>

Area	Tables
Users	<code>users</code> , <code>password_history</code> , <code>user_sessions</code>
Historian and datalogs	<code>historian_data</code> , <code>datalogs</code> , <code>store_forward_files</code>
Production	<code>recipes</code> , <code>batch_recipes</code> , <code>batch_instances</code> , <code>batch_phase_log</code> , <code>batch_phase_classes</code> , <code>batch_equipment</code> , <code>batch_materials</code> , <code>oee_equipment</code> , <code>oee_snapshots</code> , <code>downtime_events</code> , <code>procedures</code> , <code>procedure_steps</code> , <code>procedure_executions</code> , <code>procedure_step_log</code>
Reports	<code>report_templates</code> , <code>visual_report_layouts</code> , <code>scheduled_reports</code> , <code>charts</code>
Integration	<code>opcua_server_settings</code> , <code>opcua_server_tags</code> , <code>webhooks</code> , <code>remote_sites</code> , <code>ip_cameras</code>
Other	<code>system_settings</code> , <code>schedules</code> , <code>scripts</code> , <code>automation_actions</code> , <code>operator_notes</code> , <code>anomaly_detection_config</code> , <code>anomaly_history</code> , <code>languages</code> , <code>translations</code> , <code>project_version</code>

The SVG symbol library (table `svg_library`) is always stored in `backend\scada.db`, whichever project is active, so that all projects share one library.

WARNING: Do not open or edit `scada.db` with a database tool while the service is running. Doing so can corrupt the database or lose recent changes. Use the application, the REST API or a backup copy.

B.6.2 Historian storage

The historian stores data in one of three ways, selected in **Config** → **Data Storage** → **Historian Configuration** (Chapter 19):

Mode	Where	Notes
Local database (default)	Table <code>historian_data</code> in the project's <code>scada.db</code>	Simple. Suitable for moderate tag counts and retention.
File-Based Archive Storage (Industrial Mode)	<code>historian-archives\<tag>\<tag>_YYYYMMDD.hda</code> in the project folder, or the Archive Storage Path	One binary file per tag per day, with swinging-door compression. Older files are compressed (<code>.hda.gz</code>), and down-sampled "warm" files are <code>.whda</code> . This is the only mode that supports all retrieval modes (Delta, Count, Range and others).
External database	The table prefix given in Table Name Prefix on the selected data source	Writes that fail are buffered on disk in <code>store-forward\</code> and sent when the database is back.

B.6.3 Other files

The files `database.sqlite` and a `scada.db` at the top level of a development tree are not used by the server.

B.7 Logs

Log	Location	Content and rotation
Server console log	backend\daemon\optizeus.out.log	All server messages, time-stamped. The service wrapper rotates it at about 10 MB and keeps 8 files (optizeus.0.out.log ... optizeus.7.out.log).
Server error log	backend\daemon\optizeus.err.log	Errors and stack traces. Rotated in the same way.
Wrapper log	backend\daemon\optizeus.wrapper.log	Service start, stop and restart
Event log, audit trail, alarm history, electronic signatures	Tables in <code>scada.db</code>	Retention is set in Config → Admin → Database → Auto-Purge Internal Database . 0 means keep for ever.
Notification send log	runtime\alarm-notification-log.json , or the external database if one is configured	Last 1,000 entries
Set-up logs	C:\OptiZeus\install.log , %TEMP%\optizeus-setup.log	Installation

Show the console log inside OptiZeus with **Config** → **Admin** → **System Logs** (Chapter 29). The folder `backend\logs\` is normally empty.

Note: The server's "page not found" message says the logs are in `C:\OptiZeus\logs` . The actual logs are in `backend\daemon\` .

B.8 Checklist for a new server

1. Install with the full set-up, and confirm that the service **OptiZeus** is running.
2. Create the firewall rules for TCP 3004 and 3443, and TCP 4840 if the OPC UA server is used (B.1.1).
3. Replace the self-signed certificate with the site certificate if clients connect by HTTPS (B.5.1).
4. Activate the licence (Appendix C).
5. Log in with the default administrator account. OptiZeus asks for a new password at once (**Change the Default Password**, Chapter 24). Then change the passwords of the default operator and viewer accounts, or disable them: their change is forced only when someone logs in to them.
6. Set the historian, event, alarm and audit retention periods (Appendix C).
7. Turn on automatic backups and store them on another disk or server (Chapter 26).
8. Restrict NTFS access to `C:\OptiZeus\backend` to administrators and SYSTEM.
9. Record the Hardware ID, ports, service account and backup location in the site documentation.

Appendix C. Licence tiers, add-ons and limits

This appendix lists the OptiZeus licence editions and add-on modules, and states which limits the software actually enforces in version 2.6.1.26. It also gathers the technical limits found in the product: rate limits, session and password defaults, upload sizes, scan rates, retention defaults and query sizes. Integrators need it to size a project and choose an edition. Administrators need it to activate, move and renew a licence and to diagnose "limit reached" messages.

For the activation procedure step by step, see Chapter 2. For security settings, see Chapters 24 and 25.

C.1 Licence editions

A licence has one edition (tier). Every edition includes the same software; the edition sets the numeric limits and the feature flags recorded in the licence.

	Starter	Basic	Professional	Professional Max	Enterprise
Tags	100	1,000	5,000	10,000	20,000
Concurrent clients	1	2	5	5	10
PLC connections	1	3	8	8	20
External data sources	1	3	10	10	12
Synoptic screens	5	10	50	50	Unlimited
Users	2	5	20	20	Unlimited
Feature flags	dashboard, alarms, events, synoptic	+ trends, datalog	+ reports, recipes	+ notifications-ext	+ notifications-ext, sql-server, api, custom-drivers, redundancy, multi-server

Professional Max is Professional with a higher tag count and the extended-notification flag (SMS, WhatsApp and Telegram). E-mail notification is intended to be available in every paid edition.

A licence is issued for a period (normally 365 days from issue) and is bound to one computer by its Hardware ID (C.4).

C.1.1 What is actually enforced

The numeric limits and feature flags are recorded in every licence, but in version 2.6.1.26 only the following are checked by the server:

Limit or flag	Enforced?	What happens at the limit
Concurrent clients	Yes	See C.1.2.

Limit or flag	Enforced?	What happens at the limit
PLC connections	Yes, when a connection is created	The new connection is refused: "PLC connection limit reached (x/y)..." (code <code>PLC_LIMIT_EXCEEDED</code>).
Tags	Yes, when tags are created one by one or in bulk	The new tag is refused (code <code>TAG_LIMIT_EXCEEDED</code>). See the note on staging below.
Multi-server deployment	Yes	All Deployment functions for pushing projects to other servers need the <code>multi-server</code> flag or the Enterprise edition. Otherwise: "Multi-server deployment requires Enterprise license".
Batch Control add-on	Yes	See C.2.
External data sources	No	Counted and reported only.
Screens, users	No	Counted and reported only.
trends, datalog, reports, recipes, notifications-ext, sql-server, api, custom-drivers, redundancy	No	These functions work in every edition. Order the edition that includes them so that the installation stays compliant when enforcement is added.

Note: When project staging is on (the default, Chapter 5), new tags are first written to the staging area and reach the live project when you **Deploy**. The tag-count check is made on direct creation only, so a staged import is not stopped at the limit. Keep the tag count within the licensed number. The **License** status page shows the current count against the limit.

C.1.2 Client counting

Every browser tab or desktop client that has logged in holds one client seat for as long as its live-data connection stays open. A second tab on the same PC takes a second seat. When all seats are taken, the next login is refused with:

Client limit reached (N clients). Please upgrade your license or disconnect other clients.

The event `client_limit_exceeded` is written to the event log. A seat is freed when the tab is closed or when the connection stops answering. The server checks each connection every 30 seconds.

To see how many seats are in use, open **Config** → **Admin** → **Clients & Shutdown**. The **Connected Clients** list shows the sessions, and an administrator can disconnect one.

C.2 Add-on modules

Add-ons are sold on top of an edition and appear in the licence feature list.

Add-on	Name	Status in 2.6.1.26
<code>batch-control</code>	Batch Control (S88)	Enforced. Without it, Config → Batch Control shows "Batch Control (S88) - Add-on License Required" and the server refuses to list phase classes or start a batch (code <code>BATCH_LICENSE_REQUIRED</code>). The batch <i>datalog</i> (Chapter 20) does not need this add-on.

Add-on	Name	Status in 2.6.1.26
spc	SPC / Statistical Process Control	Recorded in the licence. The SPC screens are not locked.
ai-assistant	AI Assistant	Recorded in the licence. Not locked. The AI assistant also needs its own AI provider settings (Chapter 23).
opcua-server	OPC UA Server	Recorded in the licence. The OPC UA server pages are not locked (Chapter 8).

C.3 Trial mode

A new installation without a licence runs a **7-day trial**, which starts the first time the server runs. The top bar shows "Trial: Nd HH:MM:SS remaining", and a link **Activate License** opens the activation page.

During the trial the server uses the **Starter** limits: 1 client, 1 PLC connection, 100 tags.

Note: Two screens show other trial figures. The licence status reports the trial as "Trial" with Basic limits, and the licence dialog shows a fixed "50 tags, 5 screens, 3 users". The Starter limits above are the ones the server applies.

When the trial has expired, the top bar shows "Expired - License required". These actions are then refused with "Trial period has expired (7 days). Please activate a license.":

- creating a PLC connection;
- creating or changing a synoptic screen;
- creating a tag;
- writing a tag value through the REST API;
- starting a batch.

Everything else keeps running: PLC polling, alarms, the historian, datalogs and operator commands from screens. An expired trial never stops the plant, but the project cannot be changed until a licence is activated.

Trial state is kept in a protected file (C.5) and is tied to the Hardware ID. Deleting or damaging the file ends the trial.

C.3.1 Licence expiry

When a licence passes its expiry date, it is no longer valid and the server falls back to the Starter limits and trial behaviour. In version 2.6.1.26 an expired licence gets a new 7-day trial period, which gives the site time to install the renewal. Plan renewals before the expiry date.

C.4 Hardware ID

The Hardware ID has the form `xxxx-xxxx-xxxx-xxxx-xxxx` (five groups of four hexadecimal characters). It is a hash of:

- the serial number of the disk that holds the operating system;

- the motherboard serial number;
- the processor identifier;
- the computer name.

An older form of the ID, which also included the lowest physical network-adaptor address, is still accepted.

The hardware is read once when the server starts. Replacing the system disk or the motherboard, or renaming the computer, changes the Hardware ID after the next restart, and the licence then reports a hardware mismatch.

Important: Do not rename an OptiZeus server after the licence is activated. If you must, use **Re-bind license to this machine** (C.6.3) or ask the vendor for a new licence.

On a virtual machine, the virtual disk and virtual motherboard serial numbers are used. Cloning a VM usually produces a different Hardware ID.

C.5 Licence files

File	Where	Content
*.lic	Supplied by the vendor	The licence in JSON form: licence key, customer, edition, issue and expiry dates, Hardware ID, feature list and a signature. Any change to the content invalidates the signature.
license.dat	The folder above the <code>backend</code> folder that runs the server: C:\OptiZeus\ for the Windows service, or C:\OptiZeus\electron\resources\ if the desktop client started its own server (see Appendix B)	The activated licence, encrypted
agreement.dat	Same folder	Records acceptance of the licence agreement
grace-period.dat	Same folder	Trial state, encrypted
license-return.dat	Same folder	Written when a licence is deactivated
revoked-licenses.dat	Same folder	Licences deactivated on this computer; they cannot be activated again here

Back up these files together with the project (Chapter 26), but note that they work only on the same hardware.

C.6 Activating, moving and renewing a licence

C.6.1 First activation

When no valid licence or trial exists, OptiZeus shows the licence dialog at start-up.

1. **License Agreement.** Scroll to the end of the agreement, tick **I have read and agree to the terms of the License Agreement**, and click **Accept and Continue**.
2. **Activate Your License.** Either click **Start Free Trial**, or copy the Hardware ID with **Copy** and send it to your vendor.
3. When you receive the `.lic` file, click **Select License File (.lic)**, choose the file and click **Activate License**.
4. The dialog shows **License Activated!** with the customer name, edition and expiry date, then opens the application.

Note: The licence dialog lists edition limits (Basic 50 tags, Professional 200 tags and so on) that are out of date. The limits in C.1 are the ones the server applies.

Activation fails with one of these messages:

Message	Cause
License key is not valid for this machine. Hardware ID mismatch.	The licence was issued for another Hardware ID.
License signature is invalid or corrupted	The file was edited or damaged.
This license has been deactivated and cannot be reused...	The licence was deactivated on this computer earlier.
License has expired	The expiry date has passed.
Invalid license file. Missing required license information.	The file is not an OptiZeus licence.

C.6.2 Activating or renewing from the running system

1. Open **Config** → **Admin** → **License Activation**.
2. Check **Current License Status: Status, Edition, Expires, Licensed To**.
3. Under **Activate New License**, click **Choose License File...** and select the `.lic` file. `.lic`, `.license` and `.dat` files up to 1 MB are accepted.
4. Click **Activate License**.

The new licence replaces the old one immediately. Existing client sessions stay connected.

C.6.3 Re-binding after a hardware change

If a licence that was working stops with a Hardware ID mismatch (for example, after a motherboard replacement), open the licence dialog and click **Re-bind license to this machine** under "Already had a license that stopped working?".

Re-binding moves the existing licence to the current Hardware ID. It is refused if:

- the licence signature is invalid;
- the licence has expired;
- the licence already matches this computer.

C.6.4 Moving a licence to another computer

1. On the old server, open **Config** → **Admin** → **License Activation** → **Transfer License to Another Machine**.
2. Click **Deactivate License**. Deactivation requires an administrator login (`POST /api/license/deactivate` is administrator-only since 2.6.1.24). The licence is removed, recorded as revoked on this computer, and a **Deactivation Code** is shown.

- Click **Copy** or **Download Return File** and send the code or file to your vendor, together with the Hardware ID of the new computer.
- Activate the new licence on the new computer (C.6.1).

WARNING: Deactivation is immediate and cannot be undone on that computer. The old server falls back to trial behaviour at once. Do not deactivate a licence on a server that is controlling a running plant until the replacement server is ready.

C.7 Technical limits

The values below are built into version 2.6.1.26. "Configurable" means an administrator can change the value in the stated place.

C.7.1 Login, sessions and passwords

Item	Default	Configurable
Failed log-ins before lock-out	5 (the 6th attempt is locked)	Config → Admin → Security → Account Lockout
Lock-out duration	30 min	Same place
Lock-out scope	Per client IP address, 15-min window. The server answers "Too many login attempts. Try again later." Password changes from the login screen count as attempts too (since 2.6.1.24).	No
Inactivity auto-logout	60 min	Config → Admin → Security → Logout & Auto-Logout
Maximum session length	24 h, after which the user must log in again	No
Minimum password length	8 characters	Per user and global security settings
Password complexity	On: upper case, lower case, digit and special character	Same
Password expiry	180 days	Same
Password history	Last 3 passwords cannot be reused	Same
Disable inactive accounts after	180 days (checked daily)	Same
Factory default password	Change forced at the first login with it (since 2.6.1.24; not in demo mode)	No
General API rate limit	1,500 requests per minute per IP address ("Too many requests. Please slow down.")	No

C.7.2 Uploads and imports

Item	Maximum size
Any JSON request (for example saving a large screen)	25 MB
Bulk tag import (JSON)	50 MB
Form submissions	5 MB
Project archive import (.zip)	500 MB
Import from another HMI/SCADA project	50 MB (project archives up to 200 MB, graphics files up to 50 MB)
Alarm list import (XML)	50 MB
Screenshot-to-objects conversion	20 MB
Recipe workbook import (Excel)	10 MB
Company logo	2 MB
Branding package	5 MB
Licence file	1 MB
Backup file restore	No limit

C.7.3 Communication and scan rates

Item	Value
Main scan cycle (reads all PLC tags)	1,000 ms. It can be lowered to a minimum of 100 ms with <code>scanIntervalMs</code> in <code>performance-config.json</code> (Appendix B).
Screen update (broadcast to clients)	1,000 ms. It can be lowered to a minimum of 100 ms with <code>broadcastIntervalMs</code> in <code>performance-config.json</code> .
Siemens S7: largest block per request	S7-1200 / LOGO! 222 bytes, S7-300/400 440 bytes, S7-1500 880 bytes, other 960 bytes. Larger data blocks are read in several requests.
Modbus TCP: registers per request	120
OPC UA client	Subscription: publishing 250 ms, sampling 100 ms, queue 10, 100 nodes per request. Fallback polling 350 ms. A value is flagged stale after 5 s without an update.
EtherNet/IP (Logix tags)	Separate worker, reading continuously with a 50 ms pause between cycles. Parallel Connections: 1–6, default 2. The number is halved automatically after two timeouts. Reconnect after 5 s. A failed tag is retried every 5 min.
MQTT (Sparkplug B)	Keep-alive 60 s, reconnect 5 s, connect time-out 10 s
Edge gateway publishing	Every 1,000 ms, QoS 1
PLC reconnection	30 s between attempts. The communication alarm clears after 3 good reads in a row. A connection watchdog runs every 15 s.
Batch engine cycle	1,000 ms

Note: For Siemens S7 and Allen-Bradley PLCs, a faster main scan puts proportionally more load on the PLC and the server. Lower `scanIntervalMs` only after measuring the PLC's communication load. For OPC UA sources, values come from the subscription, so a faster scan costs very little.

C.7.4 Live data connection

Item	Value
Keep-alive check	Every 30 s. A client that misses one check is disconnected and must reconnect.
Full refresh to each client	About every 2 min, in addition to change updates
Slow clients	A client with more than 4 MB of unsent data skips change updates until it catches up
Compression	On (for messages over 1 KB)

C.7.5 Historian, datalogs and retention

Item	Default	Where to change
Historian retention	90 days	Config → Data Storage → Historian Configuration → Retention (days, 0=unlimited) . 0 = keep for ever (since 2.6.1.25; earlier versions treated 0 as 90 days).
Historian recording interval	1,000 ms (minimum 100 ms per tag)	Per tag, and Default Interval (ms)
Historian deadband	0 (every sample stored)	Per tag, and Default Deadband
Historian recording method	Interval	Per tag, and Default Method
Historian purge	Hourly	—
Historian trend: tags per chart	20	—
Historian time ranges	1 h up to 12 months, or a custom range	—
Datalog retention	0 = keep for ever; otherwise purged hourly	Per datalog
Alarm, event and audit-trail retention	0 = keep for ever; purged hourly when set	Config → Admin → Database
Datalog buffer while an external database is unreachable	10,000 records per datalog. The primary database is retried every 30 s.	—
Store-and-forward buffer	10 MB per file, 2 GB per queue (the oldest data is deleted beyond that). Flushed every 30 s, 500 records per transfer.	Chapter 27
Automatic backup	Off. When on: 02:00, kept for 30 days. 0 days = keep all (since 2.6.1.25).	Config → Admin → Backup & Restore

Important: For regulated (GxP) records, set explicit retention periods and verify them during validation (Chapter 31). Do not rely on a value of 0.

C.7.6 Alarms and notifications

Item	Value
Active alarms held in memory	1,000. Beyond that, the oldest acknowledged alarms are removed first, then the oldest unacknowledged ones. They remain in the alarm history.
Alarm flood detection	Starts at 10 alarms in 60 s. Ends when the rate falls below 5 per 60 s.
Notification throttling (defaults)	30 notifications per minute, 200 per hour, 60 s cooldown per alarm
SMS	At most 10 per hour (default)
Alarm shelving	At most 3,600 s (default)
Notification grouping	Every 60 s, at most 10 alarms per message
Webhook delivery	3 retries, 5 s apart, 10 s time-out
Escalation delays (defaults)	5, 15 and 30 min

Chapter 17 describes how to change the notification settings.

C.7.7 Queries and exports

Function	Default rows	Maximum
Historian query (trend view)	100,000	—
Tag history range	50,000	—
Alarm history range	1,000 (500 in trend charts)	—
Audit trail	1,000	—
Sequence of events	1,000 (5,000 on the timeline)	—
System logs	1,000	—
Batch history	50	—
REST export: tags	1,000 per page	10,000
REST export: historian	5,000	50,000
REST export: alarms, events	1,000	10,000
Visual report data table	50 rows	500
Live mini-trends on screens	60 to 100 points	—

There is no fixed limit on the number of screens or objects per screen, apart from the licence screen count, which is not enforced. Screens with several thousand animated objects load slowly on older client PCs. Split them into several screens.

C.7.8 Other limits

Item	Value
Python script time-out (click actions and scheduled scripts)	30 s
AI assistant answer length	16,384 tokens
Undo history in the editors	50 steps

Appendix D. Glossary

This glossary defines the OptiZeus-specific terms used in this manual, together with the general automation, networking and regulatory terms that engineers, supervisors and operators meet when they work with the system. Where a term is a menu item or field name, it is shown in **bold** as the user interface labels it. The chapter in brackets is where the subject is described in full.

D.1 Terms

Acknowledge (ACK) : The operator action that confirms an alarm has been seen. An acknowledged alarm stays in the alarm list until its condition clears. Acknowledging stops a running notification pipeline for that alarm unless the pipeline is set to continue. (Chapter 17)

Active alarm : An alarm whose condition is currently true, acknowledged or not. The server holds up to 1,000 active alarms in memory. (Appendix C)

Add-on : A licensed module sold on top of an edition, for example Batch Control (S88). (Appendix C)

Alarm & Event datalog : A datalog type that stores alarms and events in an external database table. (Chapter 19)

Alarm area : A plant area (for example "Line 1") assigned to an alarm. It is used to filter alarm lists, reports and notification rules. Maintained in **Config** → **Alarms**. (Chapter 17)

Alarm flood : A burst of alarms faster than an operator can handle. OptiZeus declares a flood at 10 alarms in 60 s and ends it below 5. See also *Smart alarm grouping*. (Chapter 17)

Alarm group : A named group of alarms, which can be nested, used to organise the alarm list and to route notifications. (Chapter 17)

Alarm pipeline : A named, reusable notification flow made of steps (notify, delay, condition, set tag, end). It starts when a matching alarm is raised and stops when the alarm is acknowledged or cleared. It replaces fixed three-level escalation. (Chapter 17)

Alarm type : The way an alarm condition is evaluated. The types are **Analog (Threshold-based)**, **Boolean (On/Off)**, **Bit (Read specific bit from integer)** and **Communication (PLC Disconnect)**. (Chapter 17)

Animation : A dynamic property of a screen object driven by a tag, for example **Fill Level**, **Color Change (On/Off)** or **Blink on Alarm**. An object can carry several animations. (Chapter 12)

Anomaly detection : A statistical function that learns the normal behaviour of selected tags and flags unusual values. (Chapter 23)

Audit trail : The tamper-evident record of who changed what, when and why: configuration changes, tag writes, log-ins and electronic signatures. (Chapter 18)

Backup : A copy of the project database and configuration files, taken by hand or on a schedule. (Chapter 26)

Batch (S88) : One execution of a control recipe on a unit, run by the S88 batch engine with PLC phases. Requires the Batch Control add-on. (Chapter 20)

Batch datalog : A datalog that records tag values between a PLC start and a stop signal and names each run with a batch number. It does not need the Batch Control add-on and is the data source for batch reports. (Chapters 19 and 20)

Bit tag : A tag that extracts one bit (0–15 of an INT, 0–31 of a DINT) from an integer PLC word. (Chapter 7)

Calculated tag : A tag whose value is computed by a formula from other tags, for example $(TempF - 32) / 1.8$. It has no PLC address. (Chapter 7)

CIP (Common Industrial Protocol) : The application protocol carried by EtherNet/IP. Logix controllers are read by tag name over CIP. (Chapter 6)

Click action : What happens when an operator clicks a screen object, for example **Write Tag Value**, **Navigate to Screen** or **Open Faceplate**. It can be protected by a security letter. (Chapter 12)

Client : A browser tab or desktop application connected to the OptiZeus server. Each connected client uses one licence seat. (Appendix C)

Client ID : The identity under which a client station registers with the server. Per-station settings such as touch-screen mode and station layout are stored against it. (Chapter 4)

Communication alarm : An alarm raised when a PLC connection is lost. It clears after three good reads in a row. (Chapter 17)

Condition (alarm pipeline) : A pipeline step that branches on the alarm state, for example "still active" or "acknowledged". (Chapter 17)

Datalog : A configured recording of selected tags into a database table. The types are Standard, Batch and Alarm & Event. It can be kept in the local database or in an external SQL database. (Chapter 19)

Deadband : A band inside which changes are ignored. The *historian deadband* suppresses storing small changes. The *alarm deadband* keeps an analog alarm active until the value has moved back past the limit by the deadband, which prevents chattering alarms. (Chapters 17 and 19)

Delay (alarm) : The time a condition must stay true before the alarm is raised (**Delay (sec)**, **Delay On (sec)**) or stay false before it clears (**Delay Off (sec)**). (Chapter 17)

Deploy : Copies staged configuration changes into the live project. See *Staging*. (Chapter 5)

Digital twin : A live model view of equipment that combines real-time values with a model of the plant. (Chapter 15)

Download (recipe) : Writes a recipe's parameter values to the PLC tags. The opposite, **Upload from PLC**, copies current tag values into the recipe. (Chapter 20)

Edge gateway : The service that publishes OptiZeus tag values to an external MQTT broker. (Chapter 28)

Edition (tier) : The licence level: Starter, Basic, Professional, Professional Max or Enterprise. (Appendix C)

Electronic signature : A re-entry of user name and password, with a stated meaning and reason, required before a protected action. Signatures are stored with the audit trail for 21 CFR Part 11. (Chapter 25)

Electron client : The OptiZeus desktop application: a dedicated window around the web interface with kiosk, tray and auto-start options. (Chapter 4, Appendix A)

Engineering units (EU) : The physical unit of a tag value, for example m³/h or °C, and the scaled range the raw PLC value is mapped to. (Chapter 7)

Escalation : Sending an unacknowledged alarm to further recipients after a delay. In OptiZeus it is built with delay and notify steps in an alarm pipeline. (Chapter 17)

EtherNet/IP : An industrial Ethernet protocol (TCP port 44818) used by Allen-Bradley Logix controllers and many devices, including the Zeus PLC tag server. (Chapter 6)

Event : A time-stamped record of something that happened: an operator action, a log-in, a state change or a system message. Events are shown in the event logger and the event log. (Chapter 18)

Faceplate : A pop-up panel for one tag or piece of equipment, showing its value, limits and controls, opened from a screen object. (Chapter 13)

First-out : In a sequence of events, the first alarm of a trip sequence, which usually shows the root cause. (Chapter 18)

Hardware ID (HWID) : The identifier of a server computer, derived from its disk, motherboard, processor and name, to which a licence is bound. (Appendix C)

Historian : The OptiZeus function that stores tag values over time for trends, reports and analysis. It can store to the local database, an external database, or file-based archives with compression. (Chapter 19)

Historian tiers : Keeping recent data at full resolution ("hot", 30 days by default) and older data as aggregated buckets ("warm", 365 days by default). (Chapter 19)

Instance (UDT instance) : A set of tags created from a UDT for one piece of equipment, for example `Pump_P201` from UDT `Pump`. Screens can bind to an instance and pick its members. (Chapter 7)

ISA-18.2 : The standard for alarm management in the process industries: alarm life cycle, rationalisation, shelving, flood handling. (Chapter 17)

ISA-88 (S88) : The standard for batch control: physical model (process cell, unit, equipment module), procedural model (procedure, operation, phase) and recipe types. (Chapter 20)

KPI dashboard : A configurable dashboard of key performance indicators built from tag values and calculations. (Chapter 22)

Kiosk role : A role with **Kiosk Mode** switched on. Users with this role run full screen and cannot leave the application. (Appendix A)

Layout : The arrangement of panels (alarm banner, navigation, synoptic, trends and others) on the main operator screen. It can be defined per station. (Chapter 14)

Memory tag : A tag that exists only in OptiZeus, not in a PLC. It holds a value written by an operator, script or action. With **Retentive**, it keeps its value over a server restart. (Chapter 7)

MQTT : A publish/subscribe messaging protocol (TCP 1883, or 8883 with TLS) used for IIoT data. (Chapters 6 and 28)

Multi-site : Viewing data from several OptiZeus servers at other sites from one server. (Chapter 27)

Named query : A saved SQL query with parameters that reports and scripts can run by name. (Chapter 21)

Node ID : The identifier of a variable in an OPC UA server, for example `ns=1;s=TT_101.Value` (namespace 1, string identifier). (Chapter 6)

Notification priority : A number from 1 to 1000 on an alarm (lower means more important) used by notification rules to decide which alarms are sent. The default is 500. (Chapter 17)

OEE (Overall Equipment Effectiveness) : Availability × Performance × Quality, expressed as a percentage. It measures how much of the planned production time is truly productive. (Chapter 22)

OPC UA : OPC Unified Architecture: a secure, platform-independent client/server protocol (usually TCP 4840). OptiZeus is an OPC UA client for PLCs and can also be an OPC UA server for other systems. (Chapters 6 and 8)

OPC UA Alarms & Conditions : The OPC UA mechanism for alarms. OptiZeus can subscribe to the alarms of a connected OPC UA server and show them in its own alarm list. (Chapter 6)

Operator notepad : Shared shift notes that operators write and read in OptiZeus. (Chapter 23)

PCCC : The older Allen-Bradley command set used by MicroLogix and SLC 500 controllers, carried over EtherNet/IP. Addresses look like `N7:0` or `B3:0/5`. (Chapter 6)

Phase (S88) : The smallest procedural element, executed by the PLC, for example Fill, Heat or Agitate. OptiZeus commands it through the PLI handshake. (Chapter 20)

Phase class : The definition of a phase type with its parameters and report parameters. It is reused on every unit that supports it. (Chapter 20)

PLI (Phase Logic Interface) : The tag handshake between the batch engine and a PLC phase: command, state, owner, step, message, parameters and report parameters, named `<Unit prefix>_<Phase suffix>_CMD` and so on. (Chapter 20)

Process Viewer : A special screen type for viewing a process area with its equipment, edited from the Viewer Builder. (Chapter 14)

Quality (tag quality) : Whether a tag value is trustworthy (good, bad, stale). A bad-quality value is shown with a distinct colour by the quality and alarm animations. (Chapter 12)

Recipe : A named set of tag/value pairs (parameters) that can be downloaded to a PLC in one action. In S88, recipes are SFC procedures with phase parameters. (Chapter 20)

Retentive : Keeps the last value of a memory tag across restarts. (Chapter 7)

Retrieval mode : How the historian returns data for a trend: Raw, Interpolated, Average, Minimum, Maximum, Delta, Count or Range per interval. Modes other than Raw and Average need file-based archive storage. (Chapter 19)

Role : A named set of permissions (for example admin, engineer, operator, viewer) and security letters assigned to users. (Chapter 24)

Scaling (linear) : Converting a raw PLC value (for example 0–32767) to engineering units (for example 0–100 %) with **Raw Min/Max** and **Eng Min/Max**. (Chapter 7)

Scan cycle : The interval at which the server reads all PLC tags (1,000 ms by default). (Appendix C)

Screen replay : Plays back a synoptic screen with historical values, as it looked at a past time. The replayed screen is read-only. (Chapter 15)

Security letter : One of 26 letters (A–Z) given to roles and required by protected objects and actions. A user can operate an object only if one of the user's roles holds the object's letter. (Chapter 24)

Sequence of events (SOE) : A time-ordered list of alarms and events around an incident, used to find the first-out and analyse trips. (Chapter 18)

Severity : How serious an alarm is: **Critical**, **Warning** or **Info**. It drives colours, sounds, filters and notifications. (Chapter 17)

SFC (Sequential Function Chart) : A graphical sequence of steps and transitions. OptiZeus uses it for S88 recipe procedures. (Chapter 20)

Shelving : Temporarily suppressing an alarm, with a reason, for up to one hour by default. (Chapter 17)

Simulation tag : A tag that generates values without a PLC (random, sine, ramp, constant and others). Use it to build and test screens before commissioning. (Chapter 7)

Smart alarm grouping : A view that groups related alarms and shows flood status, so that the operator can handle a burst of alarms by cause. (Chapter 17)

Sparkplug B : A specification for MQTT topics and payloads for industrial data, with birth and death certificates. (Chapter 28)

SPC (Statistical Process Control) : Control charts, histograms and capability indices (Cp, Cpk) that show whether a process is stable and within specification. (Chapter 22)

Staging : Collecting configuration changes (tags, screens, alarms) in a staging area and applying them to the live system together with **Deploy**. It is on by default. (Chapter 5)

Store-and-forward : Buffers data on disk when a remote database or server cannot be reached, and sends it when the link is back. (Chapter 27)

Super object : A reusable graphic built from several objects with animations and parameters, placed on screens many times. The OptiZeus equivalent of a symbol with properties. (Chapter 13)

Super Tag : See *UDT*. The **UDT / Super Tags** tab is where UDTs and their instances are managed. (Chapter 7)

Swinging-door compression : The compression method of the file-based historian archive. A new point is stored only when the trend can no longer be represented within the **Compression Deviation**. (Chapter 19)

Synoptic (screen) : A graphical process screen (mimic) showing the plant with live values and controls. (Chapters 10 and 16)

Tag : A named data point: a PLC value, a calculated value, a memory value or a simulated value, with its type, units, limits, alarms and historian settings. (Chapter 7)

Touch-screen mode : A per-station setting that opens an on-screen keyboard for value entry. (Appendix A)

Trial : The 7-day evaluation period of an unlicensed installation. (Appendix C)

UDT (User-Defined Type) : A tag template with members (for example `Run`, `Fault`, `Speed`, `Hours`) from which instances for many identical machines are created. (Chapter 7)

Unit (S88) : A piece of equipment that runs one batch at a time, for example a mixer or a reactor. (Chapter 20)

Viewer Builder : The OptiZeus screen editor, in **Config** → **Viewer Builder**. (Chapter 10)

Webhook : An HTTP call that OptiZeus makes to another system when an event occurs, for example when an alarm is raised. (Chapter 28)

Zeus PLC : The Zeus Automation soft-PLC runtime for Raspberry Pi Compute Module controllers, programmed in IEC 61131-3 Structured Text and Ladder. OptiZeus connects to it over OPC UA or EtherNet/IP. (Appendix E)

D.2 Abbreviations

Abbreviation	Meaning
ACK	Acknowledge
AD	Active Directory
API	Application Programming Interface

Abbreviation	Meaning
BOOL, INT, DINT, REAL	PLC data types: Boolean, 16-bit integer, 32-bit integer, 32-bit floating point
CFR	Code of Federal Regulations (21 CFR Part 11: electronic records and signatures)
CIP	Common Industrial Protocol
CSV	Comma-separated values
DB	Database, or Siemens data block
ERP	Enterprise Resource Planning
EU	Engineering units
GAMP	Good Automated Manufacturing Practice
GxP	Good (manufacturing, laboratory, clinical...) practice regulations
HH, H, L, LL	High-high, high, low, low-low alarm limits
HMI	Human-Machine Interface
HWID	Hardware ID
IdP	Identity provider (for single sign-on)
IQ / OQ / PQ	Installation / Operational / Performance Qualification
JWT	JSON Web Token (the log-in session token)
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
MFA	Multi-factor authentication
MES	Manufacturing Execution System
OEE	Overall Equipment Effectiveness
OIDC	OpenID Connect
OPC UA	OPC Unified Architecture
P&ID	Piping and instrumentation diagram
PLC	Programmable Logic Controller
PLI	Phase Logic Interface
PWA	Progressive Web App
QoS	Quality of Service (MQTT: 0, 1 or 2)
REST	Representational State Transfer (the web API style)
RTU	Remote Terminal Unit
S88	ISA-88 batch control standard
SCADA	Supervisory Control and Data Acquisition
SFC	Sequential Function Chart

Abbreviation	Meaning
SMTP	Simple Mail Transfer Protocol (e-mail sending)
SOE	Sequence of events
SPC	Statistical Process Control
SQL	Structured Query Language
ST	Structured Text (IEC 61131-3)
TLS / SSL	Transport Layer Security (encryption for HTTPS, MQTTS, SMTP)
TOTP	Time-based one-time password (authenticator app code)
UDT	User-Defined Type
URS	User Requirements Specification
WAL	Write-ahead log (SQLite journal mode)
WS / WSS	WebSocket / secure WebSocket (live data connection)

Appendix E. Worked example projects

This appendix walks through six complete example projects. Each one covers the connection, the tags, the screen, the alarms and the recording, followed by a test procedure. Engineers can rebuild them on a test server to learn the product, or use them as templates for real projects.

Every field and button named here exists in version 2.6.1.26. Where the software cannot yet do something the example would naturally need, the example says so and shows the workaround.

The chapters that describe each subject in full are cross-referenced. Plant names such as Tank_101 or Line 1 are generic.

WARNING: Build and test these examples on a test server with simulated tags or a PLC that is not controlling equipment. Every **Write Tag Value**, recipe download and tag-monitor edit writes to the PLC immediately.

E.0 Common procedures

The examples use the following steps repeatedly.

E.0.1 Build with simulation first

To build and test a screen before the PLC is available:

1. Create the tag with **Data Source = Simulation**.
2. Choose a **Simulation Type**, for example **Sine Wave** for a level, or **Random Toggle** for a run signal. Set **Simulation Min**, **Simulation Max** and **Update Interval (ms)**.
3. When the PLC is ready, edit the tag. Set **Data Source** to **PLC / External Device**, select the **PLC Connection** and enter the **PLC Address**. The tag name stays the same, so screens, alarms and trends keep working.

E.0.2 Staging and deployment

When staging is on (the default), new and changed tags, screens and alarms go to the staging area first. The message "...has been staged for deployment" appears. Click **Deploy** to make them live (Chapter 5). The test procedures assume the changes have been deployed.

E.0.3 Tag names

Use letters, digits and underscores. Put a plant area in the **Folder** field (for example TankFarm), not in the tag name, when you want the tags grouped in the tag browser. Recipe and report names must not contain dots.

E.1 Tank farm level monitoring with alarms

Goal. Show three storage tanks with live level, volume and inlet-valve state. Raise high-high, high, low and low-low level alarms. Record the levels in the historian and show them in a trend. This is the basic pattern for any analog monitoring.

Plant. Tank_101, Tank_102 and Tank_103, each 0–100 % (0–50 m³), with a 4–20 mA level transmitter and an inlet valve with an open limit switch. A PLC with a Modbus TCP server holds the values in holding registers, scaled 0–

10000.

E.1.1 Connection

In **Config** → **PLC Connections**, click **Add PLC Connection**:

Field	Value
Connection Name	TankFarm_PLC
PLC Brand	Generic / Other
Model	Modbus TCP Device
Protocol	Modbus TCP
Host / IP Address	192.168.10.20
Port	502
Unit ID (Slave Address)	1
Enabled	On

Click **Test Connection**. It must report success. Then click **Add Connection**.

E.1.2 Tags

Create the tags in **Config** → **Tags** with **Folder** = TankFarm .

Modbus addressing. OptiZeus accepts these address forms:

- 4xxxx or HRn for holding registers;
- 3xxxx or IRn for input registers;
- 0xxxx or Cn for coils;
- 1xxxx or DIn for discrete inputs.

In the 5-digit form the first register is 1 (40001 is register 0). In version 2.6.1.26 each analog Modbus tag reads **one 16-bit register as an unsigned integer**. Transfer analog values as scaled integers and use **Linear Scaling (Raw → Engineering)**. 32-bit floating-point values over Modbus are not decoded.

Tag Name	Tag Type	PLC Address	Unit	Min / Max	Linear Scaling	Historian
Tank_101_Level	Analog	40001	%	0 / 100	Raw 0–10000 → 0–100, Clamp output to engineering range on	Enable Recording, Interval, 5000 ms
Tank_102_Level	Analog	40002	%	0 / 100	Same	Same
Tank_103_Level	Analog	40003	%	0 / 100	Same	Same

Tag Name	Tag Type	PLC Address	Unit	Min / Max	Linear Scaling	Historian
Tank_101_Volume	Analog, Data Source = Calculated, formula Tank_101_Level * 0.5	—	m ³	0 / 50	—	On Change (Deadband) , deadband 0.5
Tank_101_InletOpen	Bool	00001	—	—	—	On Change (Deadband)
Tank_102_InletOpen	Bool	00002	—	—	—	Same
Tank_103_InletOpen	Bool	00003	—	—	—	Same

Create Tank_102_Volume and Tank_103_Volume in the same way.

To create one level tag:

1. Click **Add New Tag**. Enter **Tag Name** Tank_101_Level, **Folder** TankFarm and **Tag Type** Analog.
2. **Data Source: PLC / External Device. PLC Connection:** TankFarm_PLCL . **PLC Address:** 40001 .
3. Select **Unit** %. Set **Min Value** 0 and **Max Value** 100.
4. Tick **Linear Scaling (Raw → Engineering)**. Set **Raw Range (PLC)** Min 0 and Max 10000, the engineering range Min 0 and Max 100, and tick **Clamp output to engineering range**.
5. Under **Historian**, tick **Enable Recording**. Set **Recording Method** to **Interval** and **Interval (ms)** to 5000.
6. Enter the **Label** "Tank 101 level" and click **Add Tag**.

E.1.3 Alarms

In **Config → Alarms → Alarm Configurations**, click + **Create Alarm** for each tank:

Field	Tank_101 value
Alarm Name	Tank_101_Level
Tag	Tank_101_Level
Alarm Type	Analog (Threshold-based)
Description	Tank 101 level
Alarm Group / Area	Tanks / Tank Farm (create them first in Groups & Areas)
Deadband (for alarm clearing)	1
High-High Alarm (Critical)	Threshold 95, Severity Critical, Delay (sec) 2, Message "Tank 101 overfill – stop filling"
High Alarm (Warning)	Threshold 90, Severity Warning, Delay (sec) 5, Message "Tank 101 high level"
Low Alarm (Warning)	Threshold 15, Severity Warning, Delay (sec) 5, Message "Tank 101 low level"
Low-Low Alarm (Critical)	Threshold 5, Severity Critical, Delay (sec) 2, Message "Tank 101 empty – pump suction at risk"

The deadband of 1 % keeps a level that hovers at 90 % from raising and clearing the High alarm repeatedly. The High alarm clears only when the level falls below 89 %. The delays keep short splashes from raising an alarm: the

level must stay at or above 90 % for 5 s before **High** is raised (2 s for the critical limits).

Click **Create Alarm**. Repeat for Tank_102 and Tank_103. Alternatively, click **Export**, copy the rows in the exported file, change the tag names and use **Import**.

E.1.4 Screen

In **Config** → **Viewer Builder**, click + **Create New Screen**. Enter the **Screen Name** Tank Farm and a **Description**.

For each tank, place the following objects:

Object (library)	Settings and animations
Vertical Tank (Tanks)	Animation Fill Level : Tag Tank_101_Level1 , Min Value (0%) 0, Max Value (100%) 100, Fill Direction bottom-up, Fill Color blue. Second animation Blink on Alarm : same tag, Blink on Alarm Level HiHi/LoLo.
Value Display (Text) above the tank	Animation Value Display — Number (decimals, prefix/suffix) : Tag Tank_101_Level1 , Decimal Places 1, Suffix / Unit % . Second animation Color by Alarm State : Normal green, High/Low yellow, HiHi/LoLo red, Bad Quality grey.
Value Display below the tank	Value Display — Number : Tag Tank_101_Volume , Decimal Places 1, Suffix / Unit m ³
Solenoid Valve (Valves) on the inlet pipe	Animation Color Change (On/Off) : Tag Tank_101_InletOpen , Color when FALSE grey, Color when TRUE green
Horizontal Pipe / elbows (Piping)	Static
Text Label	"TK-101"
Alarm Panel (Live) (Components) at the bottom of the screen	Shows the live alarms

To add an animation, select the object, open  **Animations**, click + **Add Animation**, choose the **Animation Type**, then select the tag in **Tag / Instance**.

Build one tank completely. Select all of its objects with Ctrl+click, press **Ctrl+C** and **Ctrl+V**, and change the tags on the copy (Appendix A). Click  **Save**.

E.1.5 Trend

- **On the screen.** Add a **Real-Time Chart** component with the three level tags for the last minutes.
- **Historian.** In the **Historian** view, select the three level tags in the **Tags** panel, choose **24h** and **Retrieval mode** Raw. Save the selection as a template named Tank_levels .

E.1.6 Test procedure

#	Action	Expected result
1	Open the Tank Farm screen.	All three tanks show a level. No object shows bad-quality grey.
2	Set register 40001 in the PLC to 5000.	Tank_101 shows 50.0 % and 25.0 m ³ , and is half full.
3	Set 40001 to 9200 and wait 5 s.	A High alarm (Warning) "Tank 101 high level" appears. The value display turns yellow.

#	Action	Expected result
4	Set 40001 to 9600 and wait 2 s.	A High-High alarm (Critical) appears. The tank blinks and the value turns red.
5	Acknowledge both alarms in the alarm panel.	The alarms stay listed as acknowledged. Blinking follows the alarm state.
6	Set 40001 to 8950.	The High alarm is still active (deadband).
7	Set 40001 to 8800.	Both alarms clear.
8	Force the Tank_101 inlet coil on.	The valve turns green.
9	Disconnect the PLC network cable.	The values show bad quality and a communication alarm is raised if one is configured (E.2.3).
10	After 10 minutes, open the Historian template Tank levels .	The trend shows the test steps.

E.2 Pump station with duty/standby and run-hour reporting

Goal. Supervise a two-pump station (Pump_P201 and Pump_P202) that runs in duty/standby. The operator selects the duty pump and resets faults from the screen. Pump faults and PLC communication loss raise alarms. A daily report shows run hours per pump.

Division of work. The duty/standby logic, automatic changeover on fault, and the run-hour counters belong in the PLC. OptiZeus shows the states, sends operator commands and reports.

Note: OptiZeus has no built-in run-hours or totaliser function in 2.6.1.26. The "Running Hours" widget of the KPI dashboard counts in the browser only and restarts at every page reload. Always keep run hours in the PLC as a retentive counter and report the counter.

PLC. Siemens S7-1200 at 192.168.10.30, with data block DB10 (not optimised; PUT/GET access allowed in the PLC's protection settings).

Offset	PLC variable	Type
DBX0.0	P201 running	BOOL
DBX0.1	P201 fault	BOOL
DBX0.2	P202 running	BOOL
DBX0.3	P202 fault	BOOL
DBX0.4	Fault reset command (pulse, reset by PLC)	BOOL
DBW2	Duty select (1 = P201, 2 = P202)	INT
DBD4	P201 run hours	REAL
DBD8	P202 run hours	REAL
DBD12	Discharge pressure, bar	REAL

Offset	PLC variable	Type
DBD16	Outlet flow, m ³ /h	REAL

E.2.1 Connection

Field	Value
Connection Name	PumpStation_PLC
PLC Brand / Model / Protocol	Siemens / S7-1200 / Siemens S7 (ISO-on-TCP)
Host / IP Address / Port	192.168.10.30 / 102
Rack / Slot	0 / 1 (S7-1200 and S7-1500 use slot 1; S7-300 usually slot 2)

E.2.2 Tags

Folder = PumpStation. A DBD address is read as REAL unless the tag type says otherwise.

Tag Name	Tag Type	PLC Address	Unit	Historian
P201_Run	Bool	DB10.DBX0.0	—	On Change
P201_Fault	Bool	DB10.DBX0.1	—	On Change
P202_Run	Bool	DB10.DBX0.2	—	On Change
P202_Fault	Bool	DB10.DBX0.3	—	On Change
PS_FaultReset	Bool	DB10.DBX0.4	—	—
PS_DutySelect	Analog	DB10.DBW2	—	On Change
P201_RunHours	Float	DB10.DBD4	h	Interval 60000 ms
P202_RunHours	Float	DB10.DBD8	h	Interval 60000 ms
PS_Pressure	Float	DB10.DBD12	bar	Interval 1000 ms, Min Value 0, Max Value 10
PS_Flow	Float	DB10.DBD16	m ³ /h	Interval 1000 ms

Optionally, create a UDT Pump with members Run, Fault and RunHours, and instances P201 and P202, so that one super object serves both pumps (Chapters 7 and 13).

E.2.3 Alarms

Alarm Name	Alarm Type	Settings
P201_Fault	Boolean (On/Off)	Tag P201_Fault, Trigger Alarm When Value is ON (True/1), Severity (On Trigger) Critical, Alarm Message (On) "Pump P201 tripped", Clear Message (Off) "Pump P201 fault cleared", Delay On (sec) 0, Related Analog Tag (for Trend View) PS_Pressure
P202_Fault	Boolean (On/Off)	The same for P202
PS_Pressure	Analog (Threshold-based)	Low Alarm (Warning) 1.5 bar, delay 10 s, "Low discharge pressure"; Low-Low Alarm (Critical) 0.5 bar, delay 5 s

Alarm Name	Alarm Type	Settings
PumpStation_Comms	Communication (PLC Disconnect)	Alarm Message "Lost connection to PumpStation_PLC", Severity Critical

Related Analog Tag lets the operator open the pressure trend from the fault alarm.

E.2.4 Screen

Object	Configuration
Centrifugal Pump P201	Color Change (On/Off) on P201_Run (grey / green). Blink when True on P201_Fault (Blink Color 1 red, Blink Color 2 yellow). Click Action: Open Faceplate.
Centrifugal Pump P202	The same with the P202 tags
Text Label "DUTY" next to each pump	Visibility (Show/Hide) on PS_DutySelect , mode Analog (value compare) , operator == , value 1 for P201 and 2 for P202
Button "Make P201 duty"	Click Action Write Tag Value, Target Tag PS_DutySelect , Value to Write 1, Security Letter (A-Z) Letter B
Button "Make P202 duty"	The same with value 2
Button "Fault reset"	Write Tag Value, Target Tag PS_FaultReset , Value to Write 1, Letter B. The PLC resets the bit.
Value Display ×4	Pressure, flow, P201_RunHours , P202_RunHours (Decimal Places 1)
Pressure Gauge (Instruments)	PS_Pressure , 0–10 bar

Give letter **B** only to the roles that may operate pumps (**Config** → **Admin** → **Security**, Chapter 24). Other users can see the screen but cannot use the buttons.

E.2.5 Daily run-hour report

Method 1: datalog and Visual Report Builder (works with the default historian)

1. In **Config** → **Data Storage** → **Data Sources**, click + **Add Data Source**. Enter **Name** LocalSQL , **Database Type** SQLite and a **Database File Path**, for example C:\OptiZeus\data\reports.db .
2. Under **Data Logs**, click + **Add Datalog**. Enter **Name** PumpHours , **Datalog Type** Standard, **Data Source** LocalSQL , **Table Name** pump_hours , **Log Mode Time Interval, Interval Value** 1 and **Interval Unit** Hours. Under **Select Tags to Log**, select P201_RunHours and P202_RunHours . Tick **Enabled**.
3. In **Config** → **Reports** → **Visual Report Builder**, create a layout from the **Daily Production Log** template.
4. Set the data source of the **Data Table** element: **Data Mode** Date Range, **Source Type** Datalog PumpHours , **Date Range** Last 24 Hours, **Max Rows** 50.
5. Add two **Formula Field** elements:

```
MAX(P201_RunHours)-MIN(P201_RunHours)
MAX(P202_RunHours)-MIN(P202_RunHours)
```

Set **Decimals** 1 and give them the labels "P201 run hours today" and "P202 run hours today".

- Save the layout. Print it with **Export PDF**, or put a button on the pump screen with **Click Action Show Visual Report**.

The formula is evaluated over the rows that the report's data elements have loaded. Keep the report period and **Max Rows** consistent: 24 hourly rows fit easily.

Method 2: historian Delta (requires file-based archive)

If **File-Based Archive Storage (Industrial Mode)** is enabled in **Historian Configuration**, open the **Historian** view:

- select `P201_RunHours` and `P202_RunHours` ;
- set **Retrieval mode** to **Delta** and **Interval** to **1 day**;
- choose **30d**.

Each point is the run hours for one day. Export the table with **PDF** or print it.

Note: The Delta, Minimum, Maximum, Count and Range retrieval modes need the file-based archive. With the default local-database historian, every mode except Raw returns averages. The CSV export of the Historian view always exports raw values.

E.2.6 Test procedure

#	Action	Expected result
1	Log in as a user with letter B. Click Make P202 duty .	<code>PS_DutySelect</code> = 2. The DUTY label moves to P202. The audit trail records the write.
2	Log in as a user without letter B and click the button.	The write is refused.
3	Set P201 fault in the PLC.	P201 blinks. The Critical alarm "Pump P201 tripped" appears. The PLC starts the standby pump, and P202 turns green.
4	Open the alarm and use the related trend.	The pressure trend around the trip is shown.
5	Clear the fault in the PLC and click Fault reset .	The "cleared" message is logged, and the reset bit returns to 0.
6	Stop the PLC or unplug its network cable.	"Lost connection to PumpStation_PLC" is raised. It clears after three good reads once the cable is back.
7	After at least two hours of logging, open the report.	The run hours per pump match the change in the PLC counters.

E.3 Batch mixer with recipe and batch report

Goal. A mixer (MX_301) makes several products. Each product has a recipe of five set-points that the operator downloads to the PLC. The PLC runs the batch sequence and signals when a batch starts and ends. OptiZeus records the batch data and prints a batch report.

This example uses the simple recipes and the **batch datalog**. Neither needs the Batch Control add-on. For full ISA-88 control with PLC phases, SFC recipes and the Phase Logic Interface, see Chapter 20. That requires the Batch

Control (S88) add-on.

PLC. Allen-Bradley CompactLogix at 192.168.10.40, slot 0. Controller-scope tags:

PLC tag	Type	Meaning
MX301_SP_Water	REAL	Water quantity, kg
MX301_SP_Syrup	REAL	Syrup quantity, kg
MX301_SP_Temp	REAL	Heating set-point, °C
MX301_SP_MixTime	DINT	Mixing time, s
MX301_SP_Speed	REAL	Agitator speed, rpm
MX301_BatchActive	BOOL	TRUE from batch start to batch end
MX301_BatchID	STRING	Batch number entered on the PLC or HMI
MX301_Temp	REAL	Product temperature
MX301_Weight	REAL	Vessel weight, kg
MX301_AgitatorRun	BOOL	Agitator running

E.3.1 Connection

Field	Value
Connection Name	Mixer_PLC
PLC Brand / Model / Protocol	Allen-Bradley (Rockwell) / CompactLogix / EtherNet/IP (CIP)
Host / IP Address / Port	192.168.10.40 / 44818
Slot	0
Parallel Connections	2 (Default)

E.3.2 Tags

For each tag, set **PLC Connection** to `Mixer_PLC`. Next to **PLC Address**, click **Browse...** and pick the controller tag, or type the tag name exactly as it is in the PLC. **Folder** = `Mixer`.

Tag Name	Tag Type	PLC Address	Unit	Min / Max	Historian
MX301_SP_Water	Float	MX301_SP_Water	kg	0 / 2000	On Change
MX301_SP_Syrup	Float	MX301_SP_Syrup	kg	0 / 500	On Change
MX301_SP_Temp	Float	MX301_SP_Temp	°C	20 / 90	On Change
MX301_SP_MixTime	Analog	MX301_SP_MixTime	s	60 / 3600	On Change
MX301_SP_Speed	Float	MX301_SP_Speed	rpm	0 / 120	On Change
MX301_BatchActive	Bool	MX301_BatchActive	—	—	On Change
MX301_BatchID	String	MX301_BatchID	—	—	—

Tag Name	Tag Type	PLC Address	Unit	Min / Max	Historian
MX301_Temp	Float	MX301_Temp	°C	0 / 100	Interval 2000 ms
MX301_Weight	Float	MX301_Weight	kg	0 / 3000	Interval 2000 ms
MX301_AgitatorRun	Bool	MX301_AgitatorRun	—	—	On Change

Also create a Temperature alarm: MX301_Temp , High 85 °C (Warning) and High-High 92 °C (Critical).

E.3.3 Recipes

In **Config** → **Recipes**, click + **New Recipe**:

Field	Value
Recipe Name	Product_A
Description	Standard syrup blend
Category	Production
Target PLC	Mixer_PLC
Status	Enabled (Released)

Under **Recipe Parameters**, add five rows with **Tag**, **Value** and **Description (optional)**, clicking + **Add Parameter** after each:

Tag	Product_A	Product_B	Description
MX301_SP_Water	1200	1000	Water charge
MX301_SP_Syrup	300	450	Syrup charge
MX301_SP_Temp	65	72	Heating set-point
MX301_SP_MixTime	900	1200	Mixing time
MX301_SP_Speed	60	75	Agitator speed

Click **Save Recipe**. To create Product_B, click **Duplicate** on the Product_A card. The copy is named "Product_A (Copy)" and saved as a draft. Rename it, change the values, set **Status** to Enabled and save.

Important: In version 2.6.1.26 the recipe parameters have no minimum, maximum or unit of their own, and the server does not check **Status** or **Security Role (Required to Download)** when a recipe is downloaded from the Recipe Manager. Protect downloads this way: - Let operators download only from a screen button. The screen's recipe list shows only enabled recipes. - Protect that button with a security letter. - Check the values against the limits in the PLC.

E.3.4 Batch datalog

1. Create a SQLite data source as in E.2.5, step 1, if you do not have one.
2. Under **Data Logs**, click + **Add Datalog**. Enter **Name** MX301_Batches , **Datalog Type** Batch, **Data Source** LocalSQL and **Table Name** mx301 .

3. Under **Select Tags to Log**, select `MX301_Temp` , `MX301_Weight` , `MX301_AgitatorRun` and the five set-point tags.
4. Under **Batch Configuration**, set:
 - **Batch Logging Type**: Interval-Based;
 - **Batch Number Source**: From Tag;
 - **Batch Number Tag**: `MX301_BatchID` ;
 - **Batch Start/Stop Tag**: `MX301_BatchActive` ;
 - **Log Interval**: 10, **Interval Unit** Second(s).
5. Tick **Enabled** and save.

Each batch creates a header row (`mx301_header` : batch number, user, start and end time) and data rows every 10 s (`mx301_data`) while `MX301_BatchActive` is TRUE. If the batch ID in the PLC is empty, set **Batch Number Source** to **Date & Time (Auto)** instead.

E.3.5 Screen

Object	Configuration
Mixer Vessel (Tanks)	Fill Level on <code>MX301_Weight</code> , 0–3000
Agitator (Mixers)	Rotate when True on <code>MX301_AgitatorRun</code>
Value Display ×3	Temperature, weight and batch ID. Use Value Display — String / Text for <code>MX301_BatchID</code> .
Recipe Table (Components)	Select Recipe <code>Product_A</code> , Highlight Match/Mismatch on. It shows the recipe values against the live PLC values.
Button "Load recipe"	Click Action Download Recipe , Security Letter C
Button "Save PLC values as recipe"	Upload Recipe (PLC → Recipe) , Security Letter D (supervisors)
Button "Attach recipe to batch"	Attach Recipe to Batch . It stores a copy of the recipe in the running batch record.
Button "Batch report"	Show Visual Report (see E.3.6)
Real-Time Chart	<code>MX301_Temp</code> and <code>MX301_Weight</code>

E.3.6 Batch report

Option A: Report Designer (PDF). In **Config → Reports → Template Reports**, click + **New Report Template**:

Field	Value
Report Name	<code>MX301 Batch Report</code>
Report Type	Batch Report
Batch Datalog	<code>MX301_Batches</code>
Batch Name Tag	<code>MX301_BatchID</code>
Detail Tags	<code>MX301_Temp</code> , <code>MX301_Weight</code>

Field	Value
Detail Mode	By Time (show data points over time)
Report Options	Include Chart , Include Data Table and Include Statistics all on. The statistics are minimum, maximum, average and number of samples.
Chart Type	Line Chart

Save, then click **Preview** or **Generate PDF**.

Note: In 2.6.1.26 the **PDF Page Layout** options and **Custom Pages** of the Report Designer are not stored when the template is saved. Set them just before generating the PDF, or use Option B.

Option B: Visual Report Builder.

1. Create a layout from the **Batch Report** template.
2. Set **Data Mode** to Batch Number, **Batch Datalog** to `MX301_Batches`.
3. Place text fields with placeholders:
 - batch information: `{{batchInfo.batchNumber}}`, `{{batchInfo.operator}}`, `{{batchInfo.startTime}}`, `{{batchInfo.endTime}}`, `{{batchInfo.duration}}`;
 - batch values: `{{batch.MX301_Temp.max}}`, `{{batch.MX301_Weight.last}}`;
 - recipe values: `{{recipe.Product_A.MX301_SP_Temp}}`.
4. Add a **Line Chart** of temperature and weight.
5. Export with **Export PDF** or **Export Excel**.

The "Batch report" button on the screen can pass the batch number to the layout (`{{param.__batchNumber}}`).

E.3.7 Test procedure

#	Action	Expected result
1	In the Recipe Manager, click Download to PLC on Product_A.	"Recipe downloaded successfully: 5 parameters written to PLC". The PLC tags hold the recipe values. The Recipe Table shows every row as matching.
2	Change <code>MX301_SP_Temp</code> in the PLC to 66.	The Recipe Table flags that row as a mismatch.
3	Click Upload from PLC on a copy of the recipe.	"Recipe updated with 5 current PLC values". The copy now holds 66.
4	In the PLC, set <code>MX301_BatchID = A-0001</code> and <code>MX301_BatchActive = TRUE</code> for 3 minutes, then FALSE.	A header row for A-0001 and about 18 data rows are stored.
5	Generate the batch report for A-0001.	It shows the start, end and duration, the temperature and weight chart and the statistics.
6	Try to download a recipe from the screen as a user without letter C.	The action is refused.

E.4 OEE dashboard for a packaging line

Goal. Calculate availability, performance, quality and OEE for the filler and the cartoner of packaging Line 1, record downtime with reasons, and show the results to supervisors.

How OEE is calculated in 2.6.1.26. Understand these rules before you set up the PLC signals:

- A snapshot is taken every 60 s for each enabled machine.
- Shifts are fixed at **06:00–14:00**, **14:00–22:00** and **22:00–06:00**. They cannot be changed.
- **Availability** = minutes running in the shift ÷ (**Planned Production Hours/Shift** × 60). Planned downtime is not deducted.
- **Performance** = (total parts × **Ideal Cycle Time**) ÷ running seconds, capped at 100 %.
- **Quality** = good parts ÷ total parts.
- **OEE** = Availability × Performance × Quality.
- The parts counters are used as they are, not as a difference. **The PLC must reset both counters at 06:00, 14:00 and 22:00.**
- A machine counts as running when the run tag is greater than 0.

PLC signals per machine: a run bit, a good-parts counter and a total-parts counter (DINT), all reset by the PLC at each shift change. In this example the line PLC is reached by OPC UA at `opc.tcp://192.168.10.50:4840`.

E.4.1 Connection and tags

Field	Value
Connection Name	Line1_PLC
PLC Brand / Model / Protocol	Generic / Other / OPC UA Server / OPC UA
Endpoint URL	<code>opc.tcp://192.168.10.50:4840</code>
Enable Username/Password Authentication	As required by the server

Tags (**Folder** = `Line1`). Next to **PLC Address**, click **Browse...** to open **Browse OPC UA Nodes** and pick each node. The node IDs below are examples.

Tag Name	Tag Type	PLC Address (node ID)
<code>L1_Filler_Run</code>	Bool	<code>ns=3;s="Filler"."Running"</code>
<code>L1_Filler_Good</code>	Analog	<code>ns=3;s="Filler"."GoodCount"</code>
<code>L1_Filler_Total</code>	Analog	<code>ns=3;s="Filler"."TotalCount"</code>
<code>L1_Cartoner_Run</code>	Bool	<code>ns=3;s="Cartoner"."Running"</code>
<code>L1_Cartoner_Good</code>	Analog	<code>ns=3;s="Cartoner"."GoodCount"</code>
<code>L1_Cartoner_Total</code>	Analog	<code>ns=3;s="Cartoner"."TotalCount"</code>

E.4.2 OEE configuration

Open **OEE** from the navigation bar. If there is no OEE button, add one to the layout with the **OEE** option (**Config** → **Display**, Chapter 14). The view is titled "OEE & Downtime Tracking".

In **Equipment Config**, add one entry per machine:

Field	Filler	Cartoner
Equipment Name	L1_Filler	L1_Cartoner
Display Label	Line 1 Filler	Line 1 Cartoner
Run Status Tag (1=Running, 0=Stopped)	L1_Filler_Run	L1_Cartoner_Run
Good Parts Counter Tag	L1_Filler_Good	L1_Cartoner_Good
Total Parts Counter Tag	L1_Filler_Total	L1_Cartoner_Total
Ideal Cycle Time (seconds/part)	0.5 (120 bottles/min)	3 (20 cartons/min)
Planned Production Hours/Shift	7.5 (8 h less a 30-min break)	7.5
Enabled	Enabled	Enabled

Adding and changing equipment needs the admin role.

Because planned stops are not deducted, express the planned time in **Planned Production Hours/Shift**. Here the 30-minute break is removed from the 8-hour shift.

E.4.3 Downtime reasons

When a run tag goes to 0, OptiZeus opens a downtime record ("Unplanned", "Unclassified") and closes it when the machine runs again. The supervisor classifies it on the **Downtime Log** tab by editing **Category** and **Reason**.

The reasons are fixed:

- **Planned:** Scheduled Maintenance, Changeover, Break/Meal, Planned Shutdown, Cleaning.
- **Unplanned:** Equipment Failure, Material Shortage, Operator Unavailable, Quality Issue, Power Failure, Other.

Downtime that the run signal does not show (for example a starved machine that still reports "running") can be added with **Record Downtime**.

E.4.4 Dashboard and reports

- The **Dashboard** tab shows one card per machine, refreshed every 10 s: Running/Stopped, A/P/Q/OEE for the current shift, good and total parts, and the active downtime reason. The OEE colour is green at 85 % or more, yellow from 60 %, orange from 40 %, and red below that.
- The **Reports** tab shows averages and the last 50 snapshots for **Last 1 Hour** to **30 Days**.
- For a wall display, add a **KPI Dashboard** with **Value Card** widgets on the counters and **LED Indicator** widgets on the run tags. KPI dashboards are stored in the browser that created them, so create the dashboard on the display PC itself.

E.4.5 Test procedure

#	Action	Expected result
1	At the start of a shift, set Filler run = 1 and let the counters increase (for example with PLC simulation).	After 2 minutes the card shows Running with non-zero A, P, Q and OEE.
2	Hold total = 200 and good = 190 for a check.	Quality = 95 %.
3	Set run = 0 for 5 minutes.	A downtime record "Unplanned / Unclassified" is open. Availability falls.

#	Action	Expected result
4	Set run = 1 and classify the stop as Changeover (Planned).	The record is closed with its duration, category and reason.
5	Check that the PLC resets the counters at the shift change.	Performance and quality start again from 0 in the new shift.

Important: Verify the availability figures against a manual calculation during commissioning. The OEE snapshot comparison uses time stamps in two different text formats, which may leave out snapshots from part of the day. If the availability is clearly too low, report it to your OptiZeus supplier before relying on it.

E.5 Connecting a Zeus PLC controller over OPC UA and EtherNet/IP

Goal. Supervise a Zeus PLC soft-controller (Structured Text program) from OptiZeus. The same controller variables can be read in two ways:

- **OPC UA:** browsable, typed, and writes are possible.
- **EtherNet/IP tags:** the controller answers like a CompactLogix, so OptiZeus reads variables by name, including function-block members such as `TT_101.EU_Value`, with no Modbus register map.

Use one of the two for a given set of tags, not both.

Example program. The controller at 192.168.1.11 runs a program with function-block instances `TT_101` (temperature transmitter, members `EU_Value`, `Status`, `AL_H_Active`, `AL_HH_Active`, `Manual`) and `PMP_201` (members `Run`, `Fault`, `Speed_SP`), plus a global `LineState` (INT).

E.5.1 Prepare the controller (Zeus PLC IDE)

For OPC UA, in Controller Settings → OPC UA Server:

1. Tick **Enabled**. Set **Port** to 4840.
2. Clear **Require security policy**. The OptiZeus OPC UA client connects with security mode None.
 - When a security policy is required, the Zeus server also makes every variable read-only.
3. Tick **Allow anonymous**, or configure a user name and password on the controller and use them in OptiZeus. Anonymous access with no security is acceptable only on an isolated control network.
4. Under **Published variables**, add the variables and members OptiZeus needs. Tick **Write** only for those it must change, for example `PMP_201.Speed_SP`. Nothing is published until you list it.
5. Click **Apply**, then **Restart runtime** in a safe state.

For EtherNet/IP, in Controller Settings → HMI tag server (EtherNet/IP):

1. Tick **Enable EtherNet/IP tags**. Leave **Bind address** empty (all interfaces) and **Port** 44818.
2. Leave **Allow writes** off unless OptiZeus must write. EtherNet/IP has no authentication.
3. Apply and download.

The tag server supports plain values only. Zeus types are presented as BOOL, SINT, INT, DINT or REAL. TIME and STRING variables are not exposed.

E.5.2 Option A: OPC UA

Connection:

Field	Value
Connection Name	Zeus_Line3
PLC Brand / Model / Protocol	Generic / Other / OPC UA Server / OPC UA
Endpoint URL	opc.tcp://192.168.1.11:4840
Enable Username/Password Authentication	Off for anonymous access, or on with the controller account
Subscribe to remote alarms (OPC UA Alarms & Conditions)	Off. Configure the alarms in OptiZeus.

Click **Test Connection**, then **Add Connection**.

Tags. In the Zeus server, published program variables are in **Objects** → **Symbols**. Function-block members are in a folder named after the instance. Every node ID is in namespace 1 with the full name as a string identifier. Browse with **Browse...**, or type the address:

Tag Name	Tag Type	PLC Address	Unit
TT101_Temp	Float	ns=1;s=TT_101.EU_Value	°C
TT101_H	Bool	ns=1;s=TT_101.AL_H_Active	—
TT101_HH	Bool	ns=1;s=TT_101.AL_HH_Active	—
P201_Run	Bool	ns=1;s=PMP_201.Run	—
P201_Fault	Bool	ns=1;s=PMP_201.Fault	—
P201_SpeedSP	Float	ns=1;s=PMP_201.Speed_SP (published with Write)	%
Line3_State	Analog	ns=1;s=LineState	—

The Zeus server also always publishes the controller status (folder **Controller**) and the raw process image (**ProcessImage**).

With OPC UA, OptiZeus uses a subscription (250 ms publishing, 100 ms sampling), so values reach the screens at the main scan rate with little PLC load.

E.5.3 Option B: EtherNet/IP tags

Connection:

Field	Value
Connection Name	Zeus_Line3_EIP
PLC Brand / Model / Protocol	Allen-Bradley (Rockwell) / CompactLogix / EtherNet/IP (CIP)
Host / IP Address / Port	192.168.1.11 / 44818
Slot	0

Field	Value
Parallel Connections	1 (Safe - slower) or 2 (Default)

Tags. Click **Browse...** next to **PLC Address**. The controller returns a flat list of every readable variable, with function-block members as `Instance.Member`. Or type the name:

Tag Name	Tag Type	PLC Address
TT101_Temp	Float	TT_101.EU_Value
TT101_H	Bool	TT_101.AL_H_Active
P201_Run	Bool	PMP_201.Run
Line3_State	Analog	LineState

Note: If OptiZeus and a Zeus PLC *simulator* run on the same Windows PC, another program on that PC may already use port 44818. Point the connection at the simulator's own IP address, not `127.0.0.1`. A real controller on the network has no such conflict.

E.5.4 Alarms and screen

Put the alarm logic in the PLC. The Zeus transmitter blocks already calculate `AL_H_Active` and `AL_HH_Active`. In OptiZeus, create **Boolean (On/Off)** alarms on `TT101_H` (Warning) and `TT101_HH` (Critical), with **Related Analog Tag (for Trend View)** `TT101_Temp`.

This keeps one set of limits, in the controller, and the alarms still work when OptiZeus is offline.

On the screen:

- **Temperature Sensor** with **Value Display — Number** on `TT101_Temp`;
- a pump with **Color Change (On/Off)** on `P201_Run` and **Blink when True** on `P201_Fault`;
- an **Input Value** object for `P201_SpeedSP`, with minimum 0 and maximum 100, **Enable Touch Keyboard** on and **Keyboard Type** Numeric (Option A only);
- **Text by State** on `Line3_State` (0 = STOPPED, 1 = RUNNING, 2 = FAULT).

E.5.5 Test procedure

#	Action	Expected result
1	Test Connection on the connection.	Success.
2	Browse the controller.	The published variables (OPC UA) or all variables (EtherNet/IP) are listed.
3	Force <code>TT_101</code> in the Zeus IDE to a value above its H limit.	<code>TT101_Temp</code> follows. The Warning alarm appears with the PLC's own limit.
4	(Option A) Enter 55 in the speed input.	<code>PMP_201.Speed_SP</code> = 55 in the Zeus monitor. The Zeus audit log records the OPC UA write.
5	(Option A) Remove Write from <code>PMP_201.Speed_SP</code> in the Zeus IDE, apply, then write again.	The write is refused, and OptiZeus reports a failed write.

#	Action	Expected result
6	Restart the Zeus runtime.	OptiZeus shows bad quality, then reconnects by itself within about 30 s.
7	Download a new program to the controller.	Published names that still exist keep working. Names that were removed show bad quality.

E.6 Alarm notification by e-mail and WhatsApp, with escalation

Goal. Send critical alarms of the pump station (E.2) to the on-shift operators by e-mail and WhatsApp during working hours. If nobody acknowledges within 10 minutes, alert the maintenance supervisor.

What the software provides in 2.6.1.26. Read this before promising a notification design to a customer.

- **Alarm notification works** by e-mail, WhatsApp (a linked phone), Telegram and GSM SMS. You choose the alarms by a **notification priority** number, the recipients by **user groups**, and the time windows by a weekly schedule per group. A message is sent when the alarm is raised and again when it returns to normal. Nothing is sent on acknowledgement.
- **The escalation settings on the Notification Overview page**, and "Enable alarm escalation" in **Global Settings**, are not used by the server. The **Repeat Interval** setting is not used either. Do not rely on them.
- **Alarm pipelines** run their **Delay**, **Condition** and **Set Tag** steps, but in 2.6.1.26 the **Notify** step does not send anything. The escalation below therefore uses a pipeline to set a PLC bit, and a second alarm on that bit is routed to the supervisor.
- Every alarm configuration — **Analog (Threshold-based)**, **Boolean** and **Bit** — uses its own **Enable Notifications** and **Notification Priority** settings (for analog configurations since 2.6.1.25; before that they used the tag's hidden notification flag). This example uses Boolean alarms.
- Voice call-out and SMS acknowledgement exist in the interface but should be treated as not functional in 2.6.1.26.

E.6.1 Channels

Open **Config** → **Alarms** → **Notification**.

1. **Overview:** switch **Alarm Notifications** on. This is the master switch; nothing is sent while it is off.
2. **Channels** → **Email (SMTP):**

Field	Example
SMTP Host	smtp.example.com
Port	587
Username	the mailbox account (e-mail is configured only when a user name is entered)
Password	the account password; stored encrypted
From Address / From Name	scada-alarms@example.com / OptiZeus Pump Station
Secure (TLS)	Off for port 587 (STARTTLS); on for port 465

Click **Save & Enable Email** and then **Verify Connection**. Enter your own address and click **Send Test Email**.

3. Channels → WhatsApp:

1. Select **Personal WhatsApp** and click **Initialize WhatsApp**.
2. After 10–30 s a QR code appears. On the phone that will send the alarms, open **WhatsApp → Settings → Linked Devices → Link a Device** and scan it.
3. Click **Save WhatsApp Config**. Enter a mobile number with country code and click **Send Test**.
The server needs Google Chrome installed, and the phone must stay online. Use a dedicated company phone and SIM, and get the recipients' agreement to receive alarm messages.

Note: The **WhatsApp Business API** fields are not used by the server in 2.6.1.26. All WhatsApp messages go through the linked phone.

E.6.2 Users and groups

Users tab, + **Add User**:

Name	Email	WhatsApp Number	Channels
Shift Operator A	operator.a@example.com	+ <country> <number>	EMAIL, WHATSAPP
Shift Operator B	operator.b@example.com	+ <country> <number>	EMAIL, WHATSAPP
Maintenance Supervisor	maint.sup@example.com	+ <country> <number>	EMAIL, WHATSAPP

Groups tab:

Field	Group Name	Users	Notification Channels (in priority order)	Delivery mode	Priority Min / Priority Max	Alarm Areas (empty = all areas)	Schedule
	Operators	Shift Operator A, Shift Operator B	+ EMAIL, + WHATSAPP	Send to all selected	100 / 500	Pump Station	Weekly Schedule , Mon–Fri 06:00–18:00
	Supervisors	Maintenance Supervisor	+ WHATSAPP, + EMAIL	Send to all selected	1 / 99	empty	Always Active

A lower priority number means a more important alarm. The group's **Priority Min** and **Priority Max** decide which alarms it receives; the priority check boxes on the group form are overridden by these numbers. With **Fallback (try next on failure)** instead of **Send to all selected**, the second channel is used only if the first one fails.

E.6.3 Alarms to send

Edit the Boolean alarms `P201_Fault` and `P202_Fault` from E.2.3:

- **Area** Pump Station;
- **Severity (On Trigger)** Critical;
- under **Notification Settings**, tick **Enable Notifications** and set **Notification Priority** 100.

They reach the Operators group.

E.6.4 Escalation after 10 minutes

1. **Add an escalation bit.** Add a BOOL `PS_Escalate` to the PLC, for example `DB10.DBX0.5`, and a tag `PS_Escalate` for it.
 - Use a PLC tag here so that the PLC can reset the bit. (Since 2.6.1.26 the **Set Tag** step can also write memory tags.)
 - Program the PLC to reset the bit when no pump fault is active any more.
2. **Add the escalation alarm.** Create a **Boolean (On/Off)** alarm `PS_Escalation` on `PS_Escalate` :
 - **Severity** Critical;
 - **Alarm Message (On)** "Pump station alarm not acknowledged for 10 minutes";
 - **Area** Pump Station;
 - **Enable Notifications** on, **Notification Priority** 50, which only the Supervisors group receives.
3. **Build the pipeline.** Open **Config** → **Alarms** →  **Pipelines**, click **+ New** and set:

Setting	Value
Name	<code>PumpFault_Escalation</code>
Enabled	On
When to trigger: Min Severity	Critical. In 2.6.1.26 only alarms of severity Critical trigger a pipeline set to High or Critical.
Alarm Type	Empty (any)
Tag Prefix	<code>P20</code> (matches the P201 and P202 fault tags)
Stop pipeline if alarm is...	Acknowledged and Cleared (returned to normal) both ticked

Add the steps:

Step	Type	Settings
1	Delay	Wait for 10 minutes
2	Condition	Check Still unacknowledged. If TRUE go to step 3. If FALSE: abort pipeline.
3	Set Tag	Tag Name <code>PS_Escalate</code> , Value 1
4	End	

Click **Save**.

The resulting sequence:

1. A pump trips. The operators get an e-mail and a WhatsApp message at once.
2. If an operator acknowledges within 10 minutes, the pipeline stops.
3. If not, the pipeline sets `PS_Escalate` . That raises `PS_Escalation` , which goes to the supervisor on WhatsApp and e-mail at any time of day.
4. When the fault clears, the PLC resets `PS_Escalate` . Both alarms send a return-to-normal message.

Important: This escalation is a workaround built from supported steps. It depends on the PLC resetting the escalation bit. Test it end to end during commissioning, including an escalation at night, and document it in the site's alarm philosophy. Keep a second, independent path for safety-critical alarms, such as a horn or beacon driven by the PLC. Notifications depend on the SCADA server, the network and public messaging services.

E.6.5 Message templates

On the **Templates** tab, adjust the texts. For example, WhatsApp — Alarm fired:

```
ALARM {category} {tagName}: {message}
Value {value} {unit} Area {area}
{timestamp}
```

The available placeholders are {tagName} {description} {value} {unit} {threshold} {category} {priority} {area} {message} {timestamp} {timestampISO} {id} .

E.6.6 Test procedure

#	Action	Expected result
1	Overview → Trigger Test Alarm .	Every channel of every matching group receives a test message. The Log tab shows the sends.
2	During the operators' schedule, set P201 fault in the PLC.	Operators A and B receive an e-mail and a WhatsApp message. The supervisor receives nothing.
3	Acknowledge within 10 minutes.	No escalation. The pipeline stops.
4	Repeat step 2 and do not acknowledge.	After 10 minutes (plus up to 5 s), PS_Escalate = 1. The supervisor receives "Pump station alarm not acknowledged for 10 minutes".
5	Clear the fault.	Return-to-normal messages are sent. The PLC resets PS_Escalate .
6	Repeat step 2 outside the schedule, for example on Saturday.	The operators receive nothing. The Log shows a SKIPPED row with the reason. The escalation still reaches the supervisor after 10 minutes.
7	Disconnect the WhatsApp phone from the internet and trigger an alarm.	The e-mail is still sent. The Log shows the WhatsApp send as failed.

When a notification does not arrive, open the **Log** tab and set the status filter to **Skipped (filtered out)**. Each skipped row states why the alarm was not sent, for example: the master switch is off, notifications are off for the alarm (its configuration's **Enable Notifications** switch, or the tag's flag for a legacy tag alarm), the priority is outside every group's range, or the group's schedule is not active.