EDGELOR / V1.2 / 2026-11-05

Node-RED User Guide

Customer Scenarios, Stepwise Checks and Linked Tutorials

8 cases / 53 local checkpoints / native numbered screenshots / separate language editions

> For devices with a delivered Node-RED runtime. Back up and test in isolation. No real control outputs are provided; simulated results are not field acceptance.

1 Choose a Workflow for the Customer Task
2 Materials and an Isolated Lab Network
3 Open Node-RED from the Device Web Page
4 Understand the Editor and Back Up Before Import
5 N01: First Run, from 255 to 25.5
6 Connect MQTT: Broker Versus Client
7 N02: Scale Temperature and Republish to MQTT
8 N03: Combine Temperature and Running State
9 N04: Hysteresis, Deduplication, and Recovery
10 N05: Detect Missing Data and Recovery
11 Bring Modbus and PLC Data Into Node-RED
12 N06: Normalize Decoded LoRaWAN Uplinks
13 N07: Provide a Read-Only HTTP Query
14 N08: POST JSON to a Customer HTTP API
15 Boundaries for Remote Control
16 Backup, Compatibility, and Maintenance
17 Evidence Files, Handover and Restoration Records
18 Delivery Checklist, Files, and References

1 Choose a Workflow for the Customer Task

This guide is for customers who need to connect field data to a platform, process instrument values, monitor equipment, or maintain existing flows. Node-RED processes messages and orchestrates applications. It does not automatically provide serial hardware, PLC drivers, LoRaWAN codecs, an MQTT broker, or actuator safety interlocks.

Customer Task	Start With	Required Result
Open the editor and run a first example	Chapters 2-5, N01	Manually inject 255 and observe 25.5 in Debug
Scale raw temperature or energy values and publish them	Chapters 6-7, N02	An independent subscriber receives the processed JSON, not just a green node status
Combine temperature and machine running state	Chapter 8, N03	true for a running machine above the threshold; false for the other valid conditions
Avoid repeated alarm notifications and report recovery	Chapter 9, N04	Raise at 30, clear at 28, and send no duplicate event while state is unchanged
Detect missing data and subsequent recovery	Chapter 10, N05	One timeout after 30 seconds without valid data, then one recovery event
Bring PLC or Modbus values into Node-RED	Chapter 11	Prove raw acquisition of 255, then adapt the actual message before processing
Normalize LoRaWAN sensor reports	Chapter 12, N06	Convert decoded temperature to a common JSON structure; do not treat Base64 as an engineering value
Let a host system query the latest state	Chapter 13, N07	HTTP 200 for fresh data; 503 for absent or expired data
Push JSON to a customer HTTP endpoint	Chapter 14, N08	Check HTTP status, business response, and receiver-side records
Handle remote control, backup, upgrades, or field faults	Chapters 15-18	Respect control boundaries, retain backups, isolate the fault, and restore safely

1.1 Current Product Applicability

Model or Platform	Current Documentation Status	Prerequisite for This Guide
YZW300L, YZW800, YZW900L, YZW1000	Current documentation lists a Node-RED environment	Verify the delivered runtime, entry point, storage, node dependencies, and project authorization

Model or Platform	Current Documentation Status	Prerequisite for This Guide
YZW001L	Standard-delivery enablement is documented; an earlier connected test interface did not show the entry	Retain the standard-delivery status and ask the supplier to verify the actual entry; a missing menu does not prove permanent lack of support
YZW900	Node-RED and its required storage/runtime are optional	Confirm that the option is fitted and delivered before following the guide
MT7628: YZW300X, YZW350L, YZW400, YZWK001, YZW1010	Current sources do not provide an on-device Node-RED environment	Do not install it using this guide; confirm a separate solution if an external PC will run it
MT7621: YZW280, YZW380, YZW800L	Current sources do not provide an on-device Node-RED environment	A shared brand or Ethernet port count does not prove software compatibility

The historical YZW350 name refers to YZW350L; use YZW350L consistently. The 4G/5G versions of YZW400 and YZW800 are configurations, not additional base models.

> Device management screenshots come from existing same-platform native English Web references. Example editor screenshots come from an isolated local Node-RED 3.0.2 environment in English. Screenshot numbers match the numbered explanations below them; the raw captures are retained beneath a separate annotation layer. Displayed status, capacity, and versions are not promises for every delivered unit. No existing device, real PLC, RF link, or actuator is accepted by this documentation exercise.

S01 | Confirm the delivered runtime before starting

Check	Explanation and Action
Expected behavior	The order and handover match the unit, with an identified Node-RED entry, runtime and maintenance owner.
Operating precautions	Complete Project_Parameters.csv first. Verify the YZW900 option; ask the supplier about a missing YZW001L entry rather than assuming no support.
Symptoms and ordered checks	Missing menu: 1. Check nameplate/order. 2. Establish whether the solution runs on an external PC. 3. Ask the supplier to verify delivery. A faulty service with a visible menu belongs to Chapter 3, not a reset.
Files, acquisition and what to inspect	On the PC retain order_scope.pdf or an order capture, plus S01_menu.png. Record model, options and approver in Project_Parameters.csv, without passwords.
Continue or roll back	Continue after the supplier confirms the entry and environment. Do not install packages or alter production settings while delivery is uncertain.

Related tutorials: T02 Editor guide and official Essentials video entry

1.2 Use This Book and Its Companion Files by Task

For first commissioning, finish Chapters 2-5 before selecting temperature publication, alarms, timeouts, native acquisition, LoRaWAN or HTTP. Each nearby checkpoint S01-S53 contains expected behavior, precautions, symptoms and ordered checks, evidence files, and continue/rollback conditions. Fault instructions are not deferred to a final FAQ.

Location in the Shared Delivery Folder	Purpose and How to Open
Two PDFs in this folder	One Chinese book and one English book with language-matched screenshots; contents, chapter references and tutorial titles are clickable
Two Word files with matching names	Editable editions; use the Word/WPS open-hyperlink action, which may require Ctrl+click depending on settings
Support/English/flows	Eight corresponding JSON flows; not upgrade packages and not for direct production import
Support/tools	site_lab.py, lab_http_receiver.py and 12 lab inputs; open PC terminals in this directory
Support/English	Blank project, 53-checkpoint, dependency, backup, evidence and timeline forms; enter actual results yourself
README_English.txt	Complete single-line lab commands; visual line wrapping in a PDF is not an extra command space/newline

Tutorial T identifiers open the original public resources, not Google search results. Section 18.3 retains a task-based index. Internet access and the PDF reader's browser-opening permission are required. If blocked, test that resource's public address in the browser; do not change gateway networking or disable security to open a tutorial.

1.3 Evidence Naming and Security

Sxx_*.txt,* .json and *.png are suggested customer-created PC evidence names, not existing device paths or automatically generated files. Create an Evidence folder in your controlled project directory and a dated, device-specific scenario subfolder for each run to avoid overwriting history.

To collect Debug evidence, attach temporary Debug, select complete-message output, finish and deploy. Expand the current entry and record node, topic, time and type. Use the version's copy-value function or manually copy a sanitized payload into a UTF-8 text file. If saving only payload, record device ID, message ID, type and time semantics in Evidence_Record.csv. Inspect ordinary warnings in Debug; use scoped Catch for catchable errors and Status for connection changes.

For Web evidence capture the complete relevant panel, retaining menu, fields, status and time. Use a real export control only if the delivered interface provides it, and record the original download name. If no log control exists, ask the administrator/supplier for the required time range instead of inventing a path. Platform/broker administrators provide their logs; identify device/Client ID, topic/endpoint, time window and errors to inspect.

Before sharing, remove passwords, tokens, Cookie, Authorization, private keys, personal information and sensitive business data. Never post complete credential files, unsanitized environment packages or raw Network HAR publicly. Evidence collection is read-only: do not erase history or change firmware source trees.

2 Materials and an Isolated Lab Network

Obtain the actual device management address, separate Node-RED credentials, and project authorization first. Do not change a production network simply to copy the addresses below. These are isolated lab values, not factory defaults.

Item	Lab Value or Material	Confirm Separately for the Project
Gateway LAN	192.168.10.1/24	Current management address, conflicts, and access scope
Lab PC	192.168.10.10/24, Python 3.8 or later	Windows can use py -3 instead of python
MQTT service	PC TCP 1884; field / DemoOnly2026	Public lab credentials only, not production secrets
Simulated Modbus TCP slave	PC TCP 1502; unit 1	Read-only register 0 starts at 255; no real instrument is connected
Simulated HTTP receiver	PC TCP 18081; /ingest	A local exercise, not the actual customer platform
Editor	Use Development > Open on the device management page	Do not assume a fixed port or path for every device
Project data	Point list, scale, units, topics, and payload samples	Explicit read/write permissions, timestamp semantics, and control approval
Safety and restoration	Full flow export, dependency list, credential handover, rollback window	Never place passwords, private keys, or certificates in public records

2.1 Start the PC Lab Services

Open two terminals in Support/tools. Confirm that the PC actually has 192.168.10.10 assigned. Keep these commands running throughout the exercises.

```
python site_lab.py bench --bind 192.168.10.10
python lab_http_receiver.py --bind 192.168.10.10 --port 18081
```

The first terminal displays LAB READY and the second displays LAB HTTP READY. If binding fails, check the PC address. If a port is in use, identify its owner instead of terminating an unknown process. When using `py -3` on Windows, replace `python` consistently at the beginning of each command.

Allow TCP 1502, 1884, and 18081 only from the isolated LAN, only when required, and remove the temporary permissions afterward. Do not disable the entire firewall or expose the services to the Internet. The MQTT lab tool does not implement TLS, persistent sessions, retained messages, or production broker

administration.

S02 | Start lab services and check network direction

Check	Explanation and Action
Expected behavior	Keep both LAB READY and LAB HTTP READY terminals open. The PC listens on 1502/1884/18081 and the gateway can reach its address.
Operating precautions	Use the tools in Support/tools. On Windows check the isolated adapter with ipconfig; PowerShell can run Get-NetTCPConnection -State Listen -LocalPort 1502,1884,18081. A PC self-test does not prove gateway-to-PC access.
Symptoms and ordered checks	Cannot bind: check adapter IP first. Address already in use: identify the listener without killing an unknown process. Gateway timeout: check the isolated route and narrowly scoped firewall rules before changing topics.
Files, acquisition and what to inspect	Copy ipconfig, listener results and startup errors into PC-created S02_network.txt. Save S02_services.png showing both READY lines and the time. The device does not generate these files.
Continue or roll back	Continue when services remain up and the MQTT node can connect. Stop them with Ctrl+C and revoke temporary firewall permissions after the exercise.

2.2 Do Not Confuse Three Levels of Saving

Action 1. Save on a device Web page updates that configuration area; it does not deploy a Node-RED flow.

Action 2. Done in a node editor updates the current draft; it does not immediately run the new flow.

Action 3. Deploy applies the draft to the runtime. After deployment, verify business behavior with inputs, outputs, and an independent receiver.

S03 | Distinguish saving a draft from running it

Check	Explanation and Action
Expected behavior	Device fields read back after Save; Done updates the draft; after Deploy, a new test input reaches its intended output.
Operating precautions	Make one traceable change at a time and record deployment scope. Deploying all flows can restart other business nodes; do not select it merely to clear a blue dot.
Symptoms and ordered checks	Saved but no change: 1. Read back device fields. 2. Inspect draft markers. 3. Read deployment feedback. 4. Send a new input. Preserve deployment errors and do not count old Debug entries as success.
Files, acquisition and what to inspect	Save S03_deploy.png with feedback, tab and test time. Export the current editor draft as flows_after.json and review the changed property. The export alone does not prove deployment.
Continue or roll back	Proceed only when new input/output matches the deployment record. Otherwise restore known-good properties and redeploy within the approved scope.

Related tutorials: T01 Creating your first flow / T03 Importing and exporting flows

3 Open Node-RED from the Device Web Page

Log in to device Web management and select Application > NODE-RED. Inspect status before opening the editor. Layouts can vary by firmware; follow the delivered interface and do not reset the environment merely because an entry is missing.

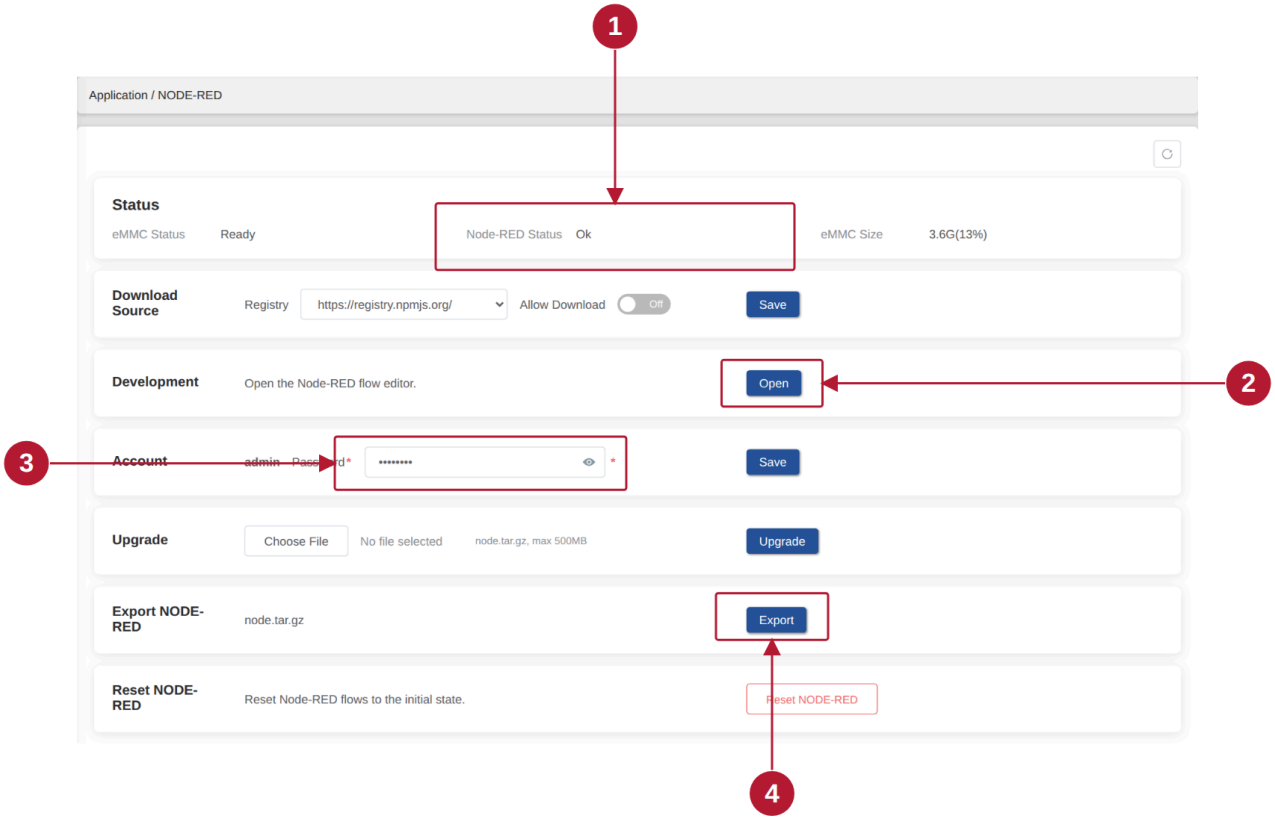


Figure 3-1 Native same-platform Node-RED management reference. Status and capacity are capture-time values only.

Callout	Action and Check
1	Check runtime and storage status first. A healthy service still needs input/output verification.
2	Click Open in Development to use the delivered editor entry.
3	Change the separate account password, save, and verify in a new session.
4	This exports the application environment. Package upgrade/reset and flow-JSON import are different operations.

Page Area	Customer Action	Check Afterward
Status: eMMC Status, Node-RED Status, eMMC Size	Check storage readiness, runtime health, and remaining capacity	Healthy service status does not prove MQTT or flow success
Download Source: Registry, Allow Download, Save	Temporarily enable only when an approved dependency installation requires it	Read back the setting and restore the previous download policy afterward

Page Area	Customer Action	Check Afterward
Development: Open	Open the supplied editor link in a new tab	Use delivered Node-RED credentials; Web login does not automatically imply editor authorization
Account: Password, Save	Change the separate password when authorized and test it in a new session	Retain a working session until the new login is verified
Upgrade: Choose File, Upgrade	Use only a supplier-approved, matching node.tar.gz	This is not the flow-JSON import location; back up before the maintenance window
Export NODE-RED	Export the application environment and check that the file is nonempty	Store it separately from editor-exported flow JSON
Reset NODE-RED	Do not use during normal operation	It may discard or restore flows; it is not the first troubleshooting action

S04 | Check storage and service, not business acceptance

Check	Explanation and Action
Expected behavior	Figure 3-1 callout 1 reports usable storage/runtime with adequate free capacity. The data path still needs independent checks.
Operating precautions	Capture the current state first. Upgrade accepts the approved environment package; Reset NODE-RED is not a routine diagnostic. Do not format eMMC.
Symptoms and ordered checks	Storage not ready/service fault: 1. Refresh once to confirm persistence. 2. Check delivery and space. 3. Ask an administrator for current startup errors. Investigate Debug/business log retention before deleting anything.
Files, acquisition and what to inspect	Save S04_status.png from this Web page. Obtain S04_startup.txt from the actual logging facility/startup output via the administrator, retaining time, error and exit reason. If no log entry exists, ask the supplier; do not invent a /var/log path.
Continue or roll back	Open the editor after storage/runtime are stable and original flows are retained. Escalate persistent faults rather than repeatedly rebooting.

3.1 If the Editor Entry Does Not Open

Action 1. Confirm that the model and order include this function. Check the YZW900 option and ask the supplier to verify the YZW001L entry.

Action 2. Refresh the management page, inspect runtime/storage status, and record the exact error. Closing a browser tab does not stop running flows.

Action 3. Check popup blocking, the PC management route, and access permissions. If the supplied entry opens an incorrect URL, retain it for the supplier to investigate.

Action 4. Handle authentication errors, timeouts, and storage faults separately. Do not default to reinstalling, upgrading, resetting, or formatting storage.

S05 | Separate connection, authorization and password failures

Check	Explanation and Action
Expected behavior	Figure 3-1 callout 2 opens the delivered URL, showing the editor or a valid login page accessible with the separate account.
Operating precautions	Do not copy localhost:1880 to the customer PC or assume Web credentials also authorize the editor. Keep the current session while verifying a changed password in a new session.
Symptoms and ordered checks	No new tab: check popup blocking. Refused/timed-out connection: verify entry, service and management route. For 401 check separate authentication; for 403 check role/access policy. Verify hostname/certificate instead of disabling checks or opening public ports.
Files, acquisition and what to inspect	Save S05_entry.txt with sanitized URL, time and browser error/HTTP status, plus S05_entry.png. Use browser Network tools only when support requests them and strip Cookie/Authorization before sharing exports.
Continue or roll back	Continue after a new session reaches the authorized editor. Stop changes while management routing or access rights remain uncertain.

Related tutorials: T02 Editor guide and official Essentials video entry / T21 Securing the editor and HTTP endpoints

4 Understand the Editor and Back Up Before Import

The editor has a node palette on the left, the workspace in the center, flow tabs and Deploy above it, and Info/Help/Debug panels on the right. Node names are editable labels; node types and properties determine behavior.

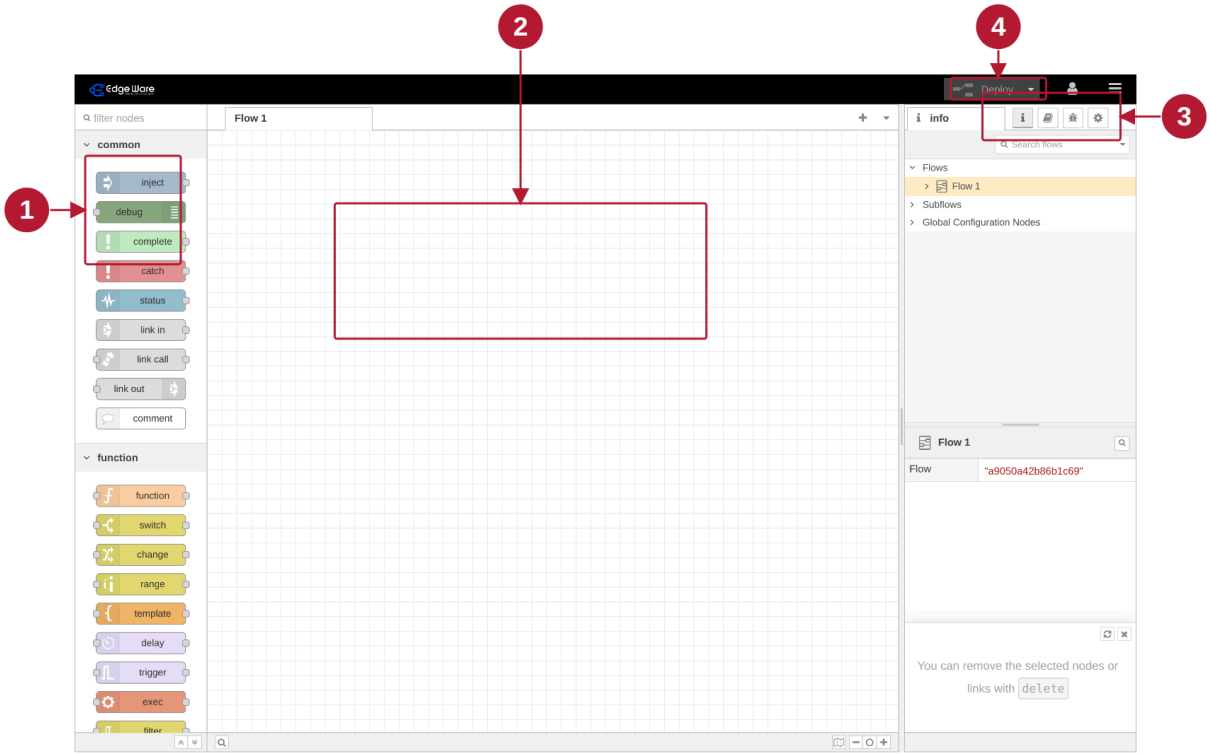


Figure 4-1 Native English editor reference: palette, workspace, sidebar, and the Deploy control.

Callout	Action and Check
1	Choose nodes in the palette; a category does not prove a third-party driver is delivered.
2	Wire the workspace and keep lab tabs separate from production tabs.
3	Use Info, Help or Debug to inspect message types and paths.
4	Deploy applies the draft. Then inspect the actual receiver result.

Area or Indicator	Purpose	Common Misunderstanding
Node palette	Find and drag inject, debug, json, function, mqtt, and other nodes	A visible category does not mean all third-party drivers are installed
Flow tabs	Separate lab and production work	Removing a tab changes the runtime only after deployment
Wires and ports	Pass data from a node output to the next node input	A connected wire does not prove a correct topic or JSON path
Blue dot or changed indicator	Identify unapplied draft edits	Finishing a node edit is not deployment

Area or Indicator	Purpose	Common Misunderstanding
Red validation indicator	Identify missing fields or invalid settings	Resolve errors instead of ignoring unknown nodes
Debug	Inspect payload, data types, and complete messages	Debug alone does not prove the cloud received the output
Main menu	Import, Export, settings, and node management	Installing a third-party node is not the same as upgrading the runtime package

4.1 Export the Current Complete Flows First

Choose Menu > Export > All flows, use formatted JSON, and download it to a controlled PC. Record device, project, and date. A flow export generally does not contain directly restorable plaintext credentials; hand over dependencies, secrets, and the application environment separately. Exporting one selected node is not a complete backup.

S06 | Make the export sufficient for rollback

Check	Explanation and Action
Expected behavior	All flows produces nonempty JSON containing the existing tabs, nodes and required configuration references, while the original service remains operational.
Operating precautions	Name the downloaded copy flows_before.json without overwriting the only backup. JSON is not an environment package; hand over credentials, external files and node packages separately.
Symptoms and ordered checks	One node/missing tabs: reopen Export and choose All flows. Invalid JSON: check for a saved HTML login page or truncated text. Unknown nodes on import require Chapter 16 dependency checks, not a latest-version production install.
Files, acquisition and what to inspect	Inspect both ends of flows_before.json with a PC text editor; python -m json.tool flows_before.json checks syntax. Review type=tab and node types. Keep secrets out of public JSON. Capture the selected scope in S06_export.png.
Continue or roll back	Import only after confirming a readable backup, restoration environment and credential owner. Do not deploy new business without a backup.

Related tutorials: T03 Importing and exporting flows

4.2 Import a Supplied Example

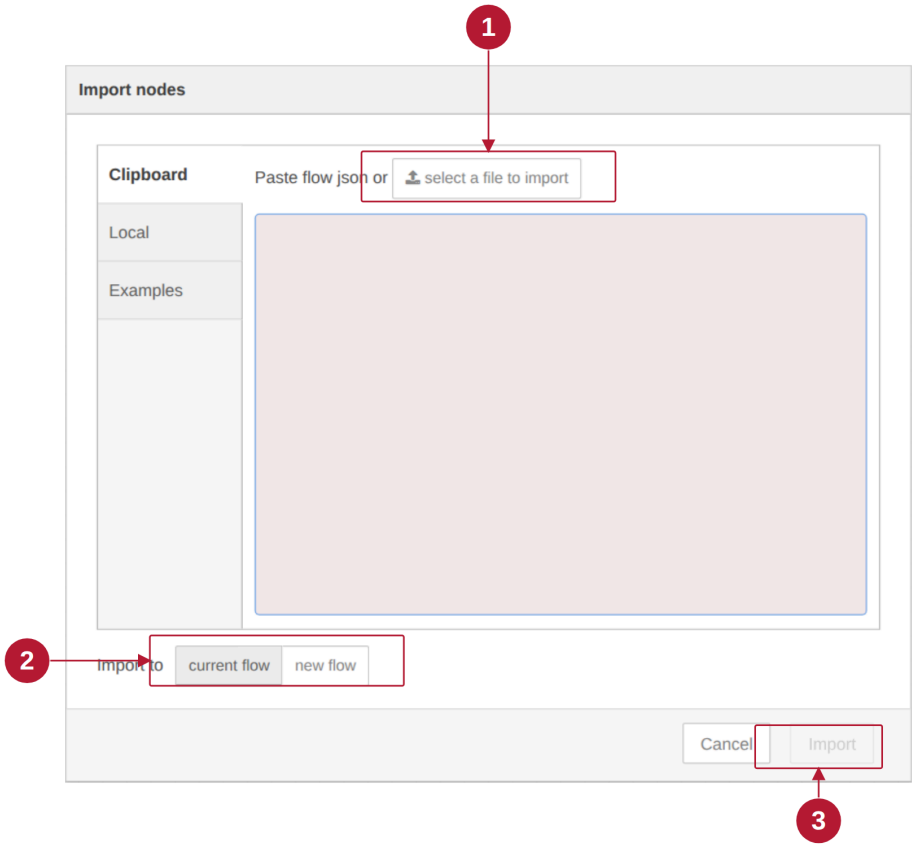


Figure 4-2 Import dialog: select the supplied flow JSON or paste it. Captured in the isolated editor.

Callout	Action and Check
1	Select a flow JSON file or paste the full JSON. This is not for node.tar.gz.
2	Import to a new flow/tab to avoid overwriting or duplicating an existing service.
3	Import becomes available after valid JSON is selected. Import creates a draft; complete parameters before deploying.

Action 1. Choose Menu > Import and select one JSON from Support/English/flows. Start with 01 and import other cases as needed; do not import everything merely to begin.

S07 | Import only the flow needed now

Check	Explanation and Action
Expected behavior	The language-specific 01_first_run.json enables preview/import and creates a separate lab tab.
Operating precautions	Use Support/English/flows for the English examples. PDF, ZIP and node.tar.gz are not flow imports.
Symptoms and ordered checks	Import disabled: 1. Check extension. 2. Confirm a complete JSON array in a text editor. 3. Reselect the original package file. Unknown nodes mean stop before deploying and record missing types.
Files, acquisition and what to inspect	Review type fields in 01_first_run.json. Save S07_import.png with filename, exact error and missing types. Do not remove unknown nodes merely to bypass validation.

Check	Explanation and Action
Continue or roll back	Continue when JSON is valid, every required node type is delivered, and the flow is isolated in a new tab.

Related tutorials: T03 Importing and exporting flows / T20 Palette Manager and dependencies

Action 2. Prefer a new flow/tab and verify that production nodes are untouched. Keep required configuration nodes. Before reimporting, check for duplicate subscriptions and HTTP paths.

S08 | Exclude duplicate tabs and HTTP routes

Check	Explanation and Action
Expected behavior	Only the intended lab instance is present, input/output topics differ, and production nodes are untouched.
Operating precautions	Inspect tabs and configuration nodes before reimporting. New node IDs do not prevent duplicate subscriptions; equal HTTP method/path pairs can conflict.
Symptoms and ordered checks	Two outputs per input: inspect duplicate tabs, then upstream retransmission, then message identity. A message storm requires disabling the lab tab and deploying before checking whether an input wildcard includes its output.
Files, acquisition and what to inspect	Export flows_after.json and inspect type=tab, mqtt in topic, and http in method/url. Save S08_tabs.png showing tabs and duplicate paths. Sanitize configuration references before sharing.
Continue or roll back	Continue after one input traverses exactly the intended chain and production remains normal. If needed, restore the approved version represented by flows_before.json.

Related tutorials: T08 Subscribe to a topic

Action 3. Double-click nodes, fill properties, and click Done. Supply broker authentication for MQTT examples; the exported JSON contains no plaintext password.

S09 | Complete configuration nodes and authentication

Check	Explanation and Action
Expected behavior	Nodes have no missing-configuration indicators and reference the intended broker; Done still needs Deploy.
Operating precautions	Enter credentials on the broker Security tab, not in a Function or public JSON. Editing a shared broker configuration affects every referencing node.
Symptoms and ordered checks	Topic present but connection fails after import: verify the configuration reference, reentered credentials, then permissions/TLS. Simultaneous failures on other tabs suggest a shared configuration was changed.
Files, acquisition and what to inspect	Save S09_broker.png with address, port, configuration name and status, never passwords. Record a credential handover reference in Project_Parameters.csv. Ask the broker administrator to match Client ID, time and rejection reason.
Continue or roll back	Deploy after configuration references and permissions are verified. Do not change shared production broker settings without authorization.

Related tutorials: T07 Connect to an MQTT broker

Action 4. Resolve unknown nodes and red validation errors before choosing an appropriate deployment scope. Full deployment may affect other flows; agree a production maintenance window first.

S10 | Validate before deployment and test a new message afterward

Check	Explanation and Action
Expected behavior	There are no unknown nodes or required-field errors; deployment completes and the current test produces a new output.
Operating precautions	Use the approved deployment scope. Resolve nodes listed as unconfigured rather than dismissing all warnings.
Symptoms and ordered checks	Failed deployment: retain the error, inspect required fields/syntax on the named node, then service/permissions. Successful deployment with no messages: check tab enabled state, Debug switch/filter and wires before the external network.
Files, acquisition and what to inspect	Capture dialog and node in S10_deploy.png; inspect changed fields in flows_after.json. Copy runtime errors from Debug or administrator-provided logs to S10_runtime.txt; a deployment-success capture alone is insufficient.
Continue or roll back	Continue when the current input, message time and output agree. Roll back the individual change on failure rather than resetting the environment.

Related tutorials: T01 Creating your first flow / T11 Handling errors, Catch and Status

Action 5. Run the case checks and export the verified version. Closing the browser is not rollback: remove or disable the lab tab and deploy the change.

S11 | Actually stop background lab flows

Check	Explanation and Action
Expected behavior	After disabling/removing the lab tab and deploying, lab timers/output stop while production remains healthy.
Operating precautions	Identify shared configuration nodes before deleting a broker. Disabling a lab flow is not the same as stopping the device runtime.
Symptoms and ordered checks	Messages continue after closing the browser: reopen, disable the tab and deploy. If output persists, find duplicate tabs or other publishers. Do not stop the production broker to stop one experiment.
Files, acquisition and what to inspect	Save S11_cleanup.png with disabled tab/deployment time. Retain S11_observation.txt from an independent subscriber showing no new lab messages and record restoration in Customer_Test_Record.csv.
Continue or roll back	Finish after lab output stops, production checks pass and temporary permissions are revoked; otherwise follow the backup rollback plan.

5 N01: First Run, from 255 to 25.5

Customer task: Verify editing, deployment, message types, and Function processing without any platform or instrument. Use `01_first_run.json`, containing only core nodes and no external write.

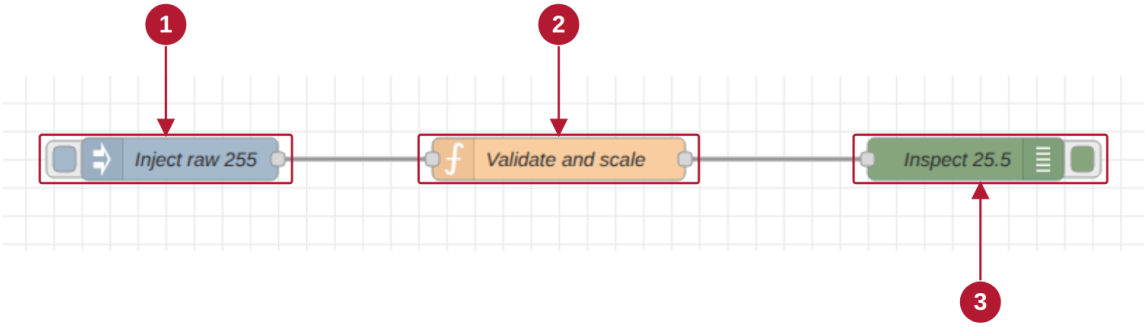


Figure 5-1 N01: inject raw 255, validate and scale by 0.1, then inspect 25.5. Isolated editor draft.

Callout	Action and Check
1	Click the Inject button manually. The input is numeric 255 inside a JSON object.
2	The Function validates the number, multiplies by 0.1, and returns msg.
3	Debug should show temperature=25.5 and unit=C; this is not a real-instrument test.

5.1 Fill the Node Properties

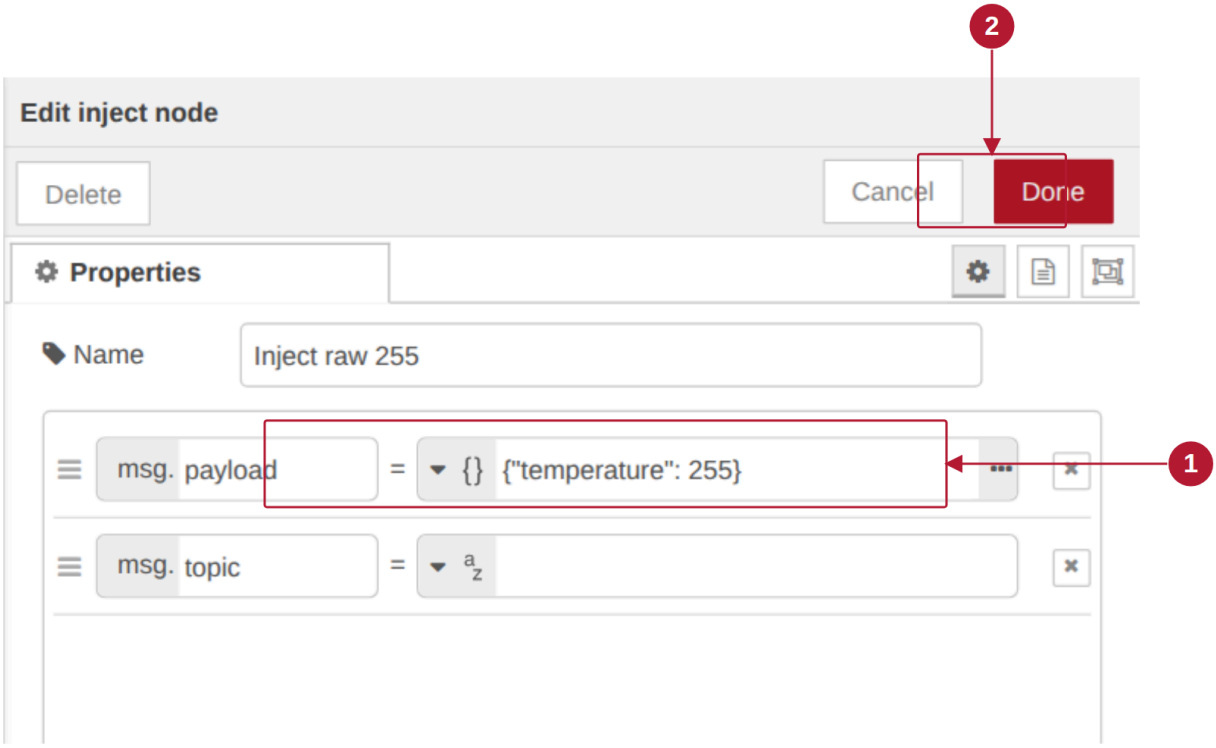


Figure 5-2 Inject properties: choose a JSON payload containing numeric 255. Leave automatic startup and repeat disabled.

Callout	Action and Check
1	Set payload to JSON and enter {"temperature":255}, without quotes around 255.
2	Done updates the node draft; deploy from the editor afterward.

Node and Field	Example Setting	Reason
Inject: payload type	JSON	temperature is numeric, not quoted text
Inject: payload content	{"temperature":255}	Lab raw value, not actual device data
Inject: repeat / inject once at start	No repeat; no automatic injection	Manually click once for the first check
Function: input	msg.payload.temperature	Object field path, not msg.temperature
Function: output	temperature multiplied by 0.1; unit C	Apply the scale exactly once
Debug: output property	msg.payload	Expand the object to inspect temperature and unit

S12 | Set the Inject payload type and value

Check	Explanation and Action
Expected behavior	In Figure 5-2 callout 1 choose JSON with {"temperature":255}, a numeric value, with startup/repeat disabled.
Operating precautions	Click Done, then Deploy. Text that looks like JSON is not the JSON payload type. Keep periodic injection off for this first test.
Symptoms and ordered checks	Type warning: temporarily inspect a complete Debug message after Inject. For string payload fix the payload type; for temperature="255" remove the value quotes; for another path correct the structure.
Files, acquisition and what to inspect	Compare payloadType=json and payload in 01_first_run.json. Save the copied Debug payload as S12_input.json and capture the displayed object/string type, which can be lost when merely copying text.
Continue or roll back	Test the Function only after object, numeric type and path are correct. Do not coerce failures silently to zero.

Related tutorials: T04 Messages, property paths and types / T10 Convert to/from JSON

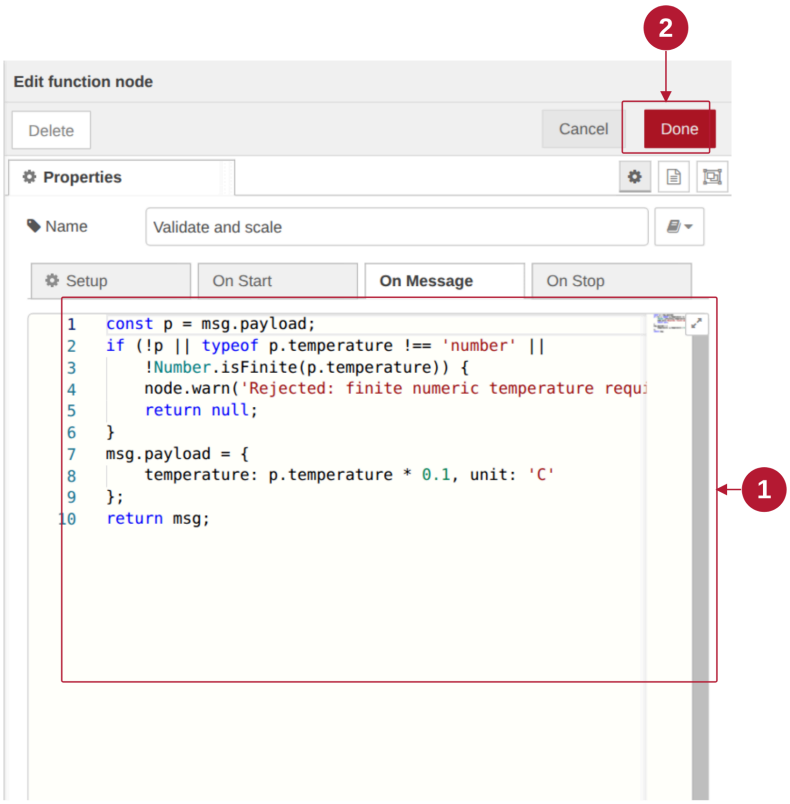


Figure 5-3 Function properties: validate the numeric type, scale it, and return msg.

Callout	Action and Check
1	Check input type, scaling and return msg in the Function/On Message area.
2	Finish editing and deploy. Do not return a bare number.

5.2 Deploy and Judge the Result

Action 1. Import 01 and check both wires. Double-click the Function to see the supplied code; no additional node installation is required.

S13 | Inspect the Function and wires

Check	Explanation and Action
Expected behavior	Inject -> Function -> Debug has both wires; the Function reads msg.payload.temperature and returns msg.
Operating precautions	Keep validation and the 0.1 scale. The Function processes a message object, not a bare return 25.5.
Symptoms and ordered checks	Input Debug has 255 but nothing downstream: 1. Read Function warnings. 2. Check path. 3. Check return null. 4. Inspect output port/wire. Fix invalid input instead of deleting validation.
Files, acquisition and what to inspect	Compare the function node's func and wires in flows_after.json with 01_first_run.json. Save S13_function.png showing the Figure 5-3 code and adjacent wires.
Continue or roll back	Continue once input reaches the Function and a valid message reaches the next node.

Related tutorials: T06 Writing Functions

Action 2. Deploy, open Debug, clear old messages, and click the Inject button once.

S14 | Inject once and identify the new message

Check	Explanation and Action
Expected behavior	After deployment, click the left Inject button once; Debug shows a current-time message from the expected node, not an old entry.
Operating precautions	Enable the Debug node switch and allow the current node/flow in sidebar filters. Clear only the displayed debug entries, not production logs or flows.
Symptoms and ordered checks	No message: check deployment and Debug enabled state, then add temporary complete-message Debug after Inject. Input present but no output points to S13. Stop/deploy unintended repeat injection if messages flood in.
Files, acquisition and what to inspect	S14_debug.png must show source node, time and filter scope. Record this click time in Customer_Test_Record.csv rather than retaining an untraceable 25.5 capture.
Continue or roll back	Continue after one click corresponds to one new output.

Related tutorials: T05 Debug sidebar and complete messages

Action 3. Expand the new object and expect `{"temperature":25.5,"unit":"C"}`. If absent, check deployment, wiring, and the Debug node's enabled state.

S15 | Accept 255 to 25.5, not merely any value

Check	Explanation and Action
Expected behavior	The new Figure 5-4 output contains numeric temperature=25.5 and unit=C.

Check	Explanation and Action
Operating precautions	Distinguish raw from engineering values. Apply 0.1 once; do not repeat scaling already performed upstream.
Symptoms and ordered checks	255: check a bypassed Function or unapplied scale. 2.55: locate a second scale from input onward. NaN/no output: inspect field and type. Compare complete current input/output; changing a unit label cannot fix the value.
Files, acquisition and what to inspect	Copy S15_before.json and S15_after.json from Debug and retain a capture showing number type. Inspect relevant Functions in flows_after.json for repeated multiplication.
Continue or roll back	Connect MQTT only after raw/scaled values, types and unit match.

Related tutorials: T04 Messages, property paths and types / T06 Writing Functions

Action 4. Change the input to `{"temperature":"255"}`, finish editing, deploy, and inject again. Expect rejection with no new valid temperature output; inspect the warning in Debug/logs.

S16 | Reject invalid input and restore valid input

Check	Explanation and Action
Expected behavior	String "255" produces no new valid output and a validation warning; restoring numeric 255 produces 25.5 again.
Operating precautions	node.warn can normally be inspected as a Debug warning; it is not a Catch event. Keep warning filters enabled and do not assume every rejected message is a crash.
Symptoms and ordered checks	No warning: check Debug level/filter and deployed revision. Valid output still appears: exclude old messages, duplicate flows or removed validation. If valid input does not recover, revisit S13-S14.
Files, acquisition and what to inspect	Save S16_rejected.png with warning, node and time, and S16_recovery.json with valid recovered output. If runtime logs are needed, request a same-period S16_runtime.txt from the administrator without assuming a path.
Continue or roll back	Invalid input must not produce a valid value and valid input must recover. Restore numeric Inject input or disable the experiment afterward.

Related tutorials: T11 Handling errors, Catch and Status

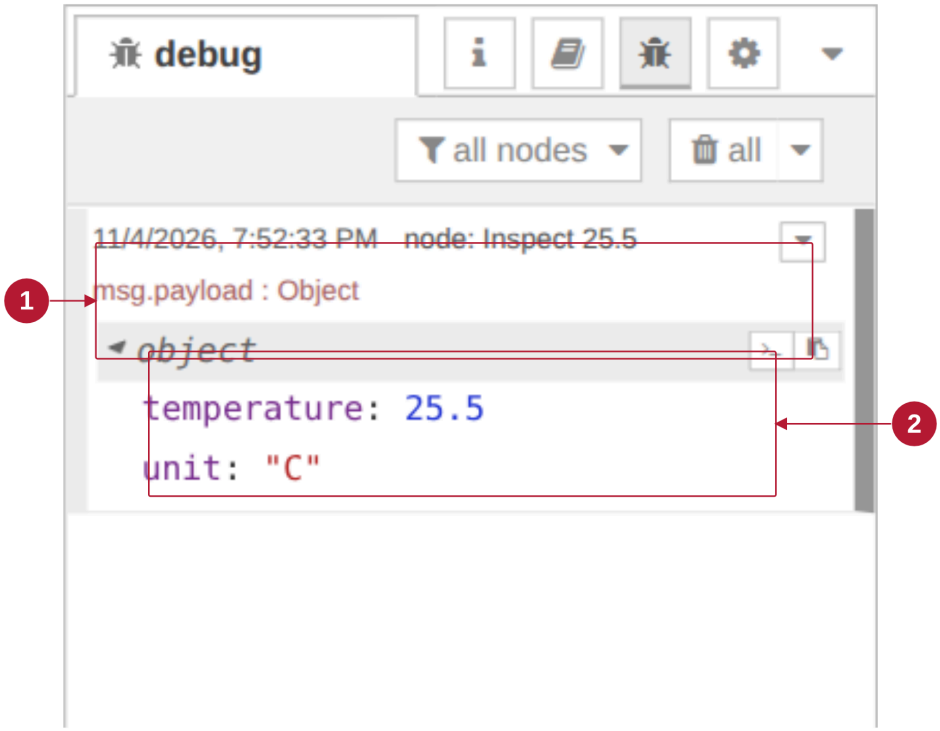


Figure 5-4 Actual N01 Debug result in the isolated local runtime; not acceptance of a real device.

Callout	Action and Check
1	Expand the current Debug message and verify it came from this Inject, not an old record.
2	Verify numeric temperature=25.5 and unit=C. Invalid types must produce no new valid result.

Restoration: Remove or disable N01 and deploy without disturbing production flows. A copied Function must return a message object, not a bare 25.5 or string. `return null` stops that message from proceeding.

6 Connect MQTT: Broker Versus Client

MQTT Input and Output nodes are clients. They connect to a broker; dragging an MQTT node into a flow does not create a broker. Start by connecting to the isolated PC broker.

6.1 Configure the Broker Through the Node Dialog

Double-click an MQTT Input node and edit its Server configuration. Fill the connection and security tabs, finish the configuration node, finish the business node, then deploy.

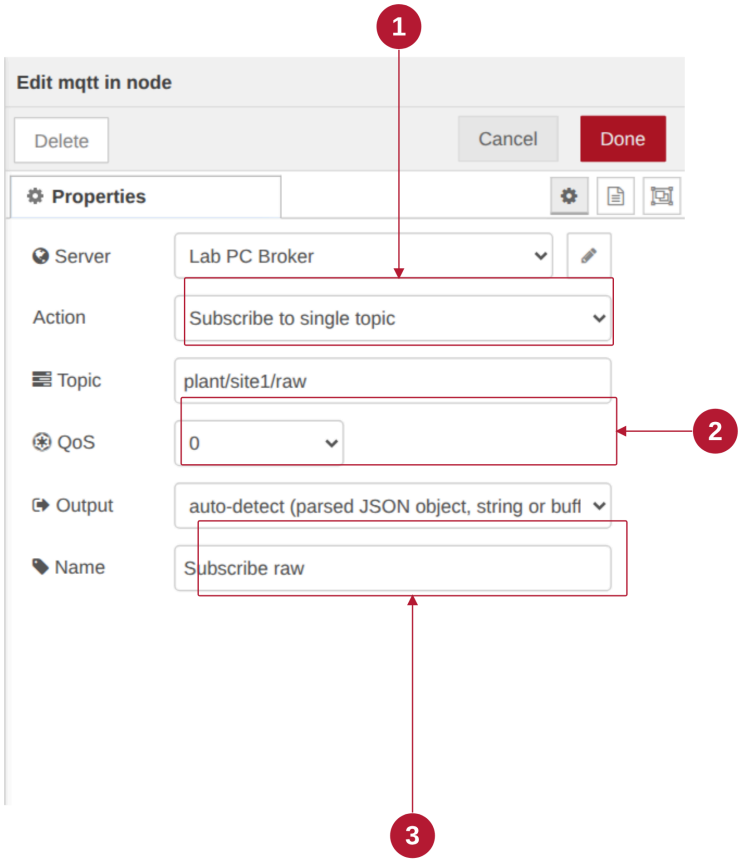


Figure 6-1 Native MQTT subscription properties: server, topic, QoS, and output type.

Callout	Action and Check
1	Use the server edit button for the broker config; it is separate from the device MQTT Service page.
2	Subscribe to plant/site1/raw; check case, slashes and QoS against the case.
3	Check the output type, then normalize it to an object with JSON before reading fields.

Field	Lab Setting	Confirm for Production
Server/Broker	192.168.10.10	A server reachable by the device, not the browser PC's localhost
Port	1884	Match the listener; do not blindly use 1883 or 8883
TLS	Off for this isolated lab	Enable and validate the CA/hostname as required by the production broker
Protocol	MQTT 3.1.1	Match the delivered broker; the lab tool does not implement MQTT 5
Username/Password	field / DemoOnly2026	Hand over secrets separately and authorize the required topics
Client ID	Leave blank for generation, or assign a unique value	Duplicate IDs can cause repeated disconnections
Clean Session	Enabled in the lab	This lab does not validate persistent sessions

Field	Lab Setting	Confirm for Production
Input topic	Defined in each case	Match case, slashes, and hierarchy exactly
QoS / Retain	QoS 0; output Retain off	QoS acknowledgement is not execution acknowledgement; the tool has no retained-message support

S17 | Check every broker connection field

Check	Explanation and Action
Expected behavior	Following Figures 6-1/6-2, use PC 192.168.10.10:1884 and lab credentials; the deployed MQTT node remains connected.
Operating precautions	Enter credentials on Figure 6-2's Security tab. MQTT 3.1.1 and TLS off apply only to this isolated service; production follows the TLS/CA/hostname handover. Client IDs must be unique.
Symptoms and ordered checks	Refused: inspect listener. Timeout: inspect route/firewall. Unauthorized: inspect credentials and broker log. Reconnecting: then check duplicate Client ID/TLS. Topics are not the first check while the transport connection is failing.
Files, acquisition and what to inspect	Save S17_connection.png with status and nonsensitive configuration. Ask the broker administrator for S17_broker.txt with time, client ID, authentication result/disconnect reason. The lab tool does not validate production ACL/TLS behavior.
Continue or roll back	Pass only after a stable connection and fresh data in the next subscription test. Keep credentials out of shared diagnostic records.

Related tutorials: T07 Connect to an MQTT broker / T11 Handling errors, Catch and Status

6.2 Difference from the Device MQTT Service Page

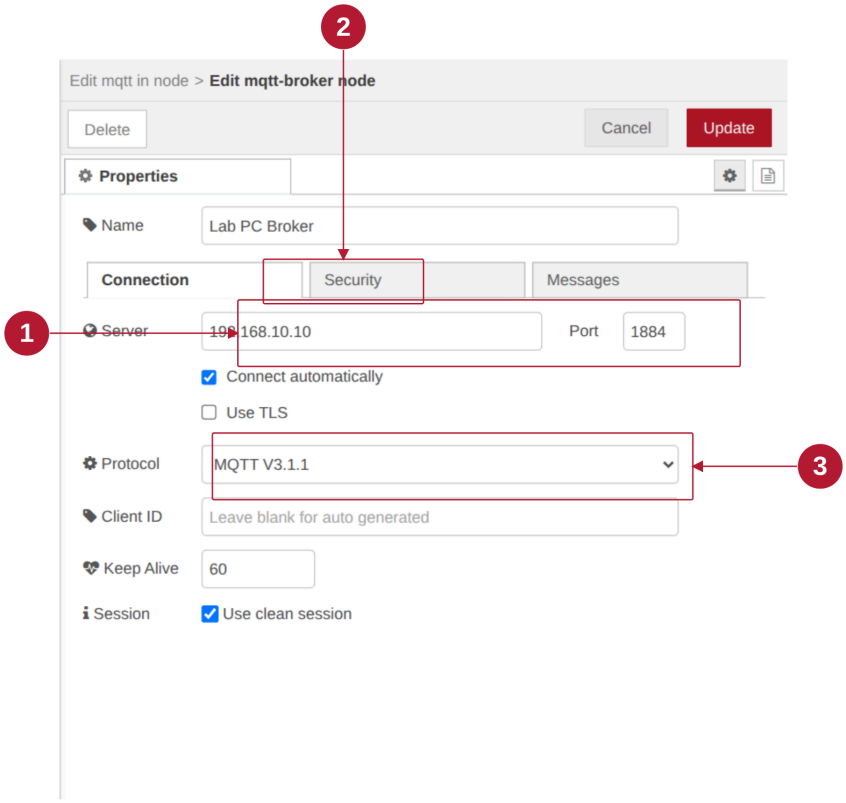


Figure 6-2 Node-RED broker configuration: verify the PC address, port 1884, protocol, and TLS; enter credentials on Security.

Callout	Action and Check
1	The lab broker is PC 192.168.10.10:1884, not the device localhost.
2	Enter lab credentials on Security. Exported flow JSON contains no plaintext password.
3	The lab uses MQTT 3.1.1. Configure production TLS and certificates to the delivered requirements.

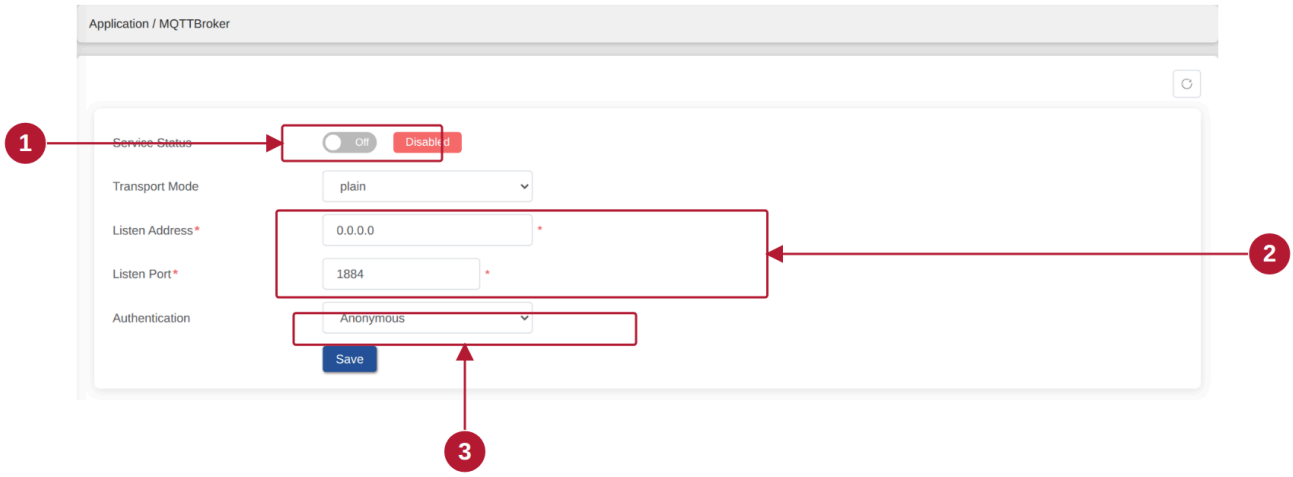


Figure 6-3 Native device MQTT Service reference: these are broker listener settings, not Node-RED client settings.

Callout	Action and Check
1	This switch starts the device broker. It need not be enabled when using the PC broker.
2	These are listener address and port, not the Node-RED client remote-server settings.
3	Do not copy anonymous access or listen-all settings; follow the project access policy.

The service in Figure 6-3 is disabled. Do not copy its listen-all address, anonymous authentication, or port. The device broker need not run when using the external PC broker. If a project requires a local broker, obtain approval for its listener, authentication, topic ACLs, access scope, and resource use before pointing clients to it.

Connection pass condition: The node is connected and an independent subscriber receives the current test input's corresponding output. A green dot, BROKER ACCEPTED, or editor Debug alone is not complete end-to-end proof.

S18 | Verify subscription after connection succeeds

Check	Explanation and Action
Expected behavior	The device broker need not run for the PC broker exercise; both independent subscriber and Node-RED receive current messages on the approved topic.
Operating precautions	Use Status for connection state and Catch for catchable errors; neither replaces independent reception. Do not subscribe to production # and copy all traffic into shared evidence.
Symptoms and ordered checks	Green status but no data: 1. Independently subscribe on the same broker and exact source topic. 2. Send one source message. 3. Match case/levels. 4. Ask the administrator to check subscribe ACL. If the source topic is empty, diagnose the source before the Function.
Files, acquisition and what to inspect	Save independent subscriber output as S18_subscription.txt with broker, topic and time, plus S18_input.json from raw-input Debug. Request same-period subscription authorization records without exposing credentials.
Continue or roll back	Continue when source, independent subscriber and Node-RED can be matched to the same new message.

Related tutorials: T08 Subscribe to a topic / T09 Publish messages to a topic

7 N02: Scale Temperature and Republish to MQTT

Customer task: An instrument reports raw integer 255 while the platform needs 25.5 degrees Celsius. Use `02_mqtt_temperature.json`, corresponding to E1. Keep input and output topics different to avoid feedback.

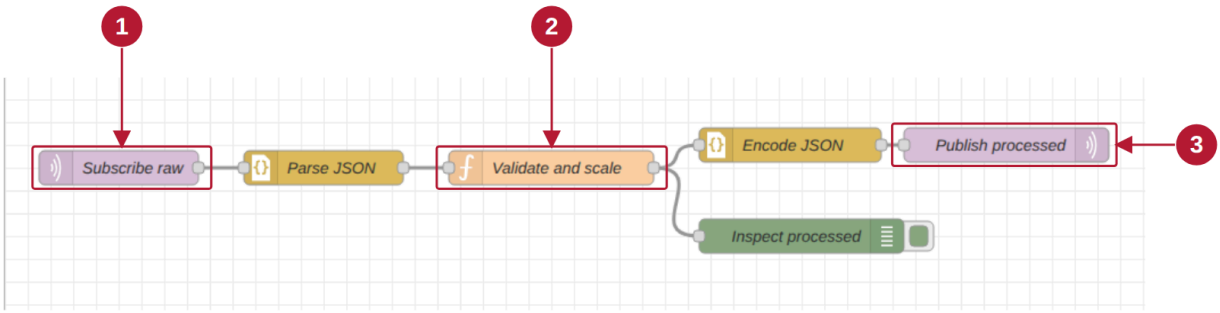


Figure 7-1 N02: subscribe raw values, parse JSON, validate and scale, then encode and republish. Isolated editor.

Callout	Action and Check
1	Use plant/site1/raw as input; first confirm raw messages arrive.
2	Parse JSON, validate it, then scale raw 255 to 25.5.
3	Republish to processed and verify with an independent subscriber, not only Debug.

Location	Lab Value	Avoid
MQTT Input	plant/site1/raw	Broad wildcard subscriptions that include the flow's own output
JSON node	Force conversion to a JavaScript object	Reading temperature directly from an unparsed JSON string
Function	Multiply numeric temperature by 0.1	Accepting strings, missing fields, NaN, or Infinity
MQTT Output	plant/site1/processed	Reusing the raw input topic
Output payload	temperature=25.5; unit=C	Assuming every unapproved raw field is preserved

7.1 Verify in Order

Action 1. Start the services in Chapter 2. Import 02, enter its broker authentication, check topics, and deploy.

S19 | Prevent feedback before deploying the temperature flow

Check	Explanation and Action
Expected behavior	N02 reads only plant/site1/raw and outputs plant/site1/processed with verified broker configurations.
Operating precautions	Do not widen input to plant/site1/# while publishing processed. Keep Retain off; the lab tool does not implement retained messages.
Symptoms and ordered checks	Traffic surges or values keep shrinking: disable the lab tab and deploy, then inspect self-matching topics, duplicate tabs and repeated scaling. Delay does not fix feedback.
Files, acquisition and what to inspect	Inspect mqtt in/out topic, broker references and Function code in 02_mqtt_temperature.json/current flows_after.json. Save S19_topics.png showing both topics and wires.
Continue or roll back	Reenable/deploy only after topic separation and a single input produce no ongoing feedback.

Related tutorials: T08 Subscribe to a topic

Action 2. Start the independent subscription first. Publish from a separate terminal during the 60-second observation window.

S20 | Subscribe before publishing one raw sample

Check	Explanation and Action
Expected behavior	An independent watch is active for 60 seconds before another terminal publishes 255 and receives the current processed message.
Operating precautions	Run watch and publish in separate terminals. Both tools run on the PC; gateway MQTT still targets 192.168.10.10, not 127.0.0.1.
Symptoms and ordered checks	watch times out: first exclude a late subscription and repeat in a fresh window. Then inspect MQTT input, after JSON, and after Function with temporary Debug. The first missing stage defines the next check.
Files, acquisition and what to inspect	Copy command, time and results into S20_watch.txt/S20_publish.txt; save raw Debug payload as S20_raw.json and check source topic/type.
Continue or roll back	Continue when a subscription established before the single 255 input receives its corresponding output.

Related tutorials: T05 Debug sidebar and complete messages / T09 Publish messages to a topic

```
python site_lab.py watch --topic plant/site1/processed --seconds 60 --expect
temperature=25.5
python site_lab.py publish --topic plant/site1/raw --set temperature=255
```

Action 3. Expect temperature=25.5 and unit=C at the independent subscriber. Check that Debug and the received payload agree.

S21 | Verify value, type and receiver together

Check	Explanation and Action
Expected behavior	The independent subscriber receives temperature=25.5 and unit=C, with a numeric temperature after JSON parsing; Debug agrees.
Operating precautions	JSON text before MQTT Output is a transport encoding; parsing at the platform should yield numeric fields, not a JSON string wrapped inside another string.
Symptoms and ordered checks	Correct Debug but no receiver message: check output broker/topic and publish ACL. Received but no platform point: inspect field mapping/type. For 2.55/255 trace scales as in S15.
Files, acquisition and what to inspect	Save independent subscriber text as S21_processed.json and platform field/update time as S21_platform.png. Ask the platform administrator for same-period S21_parser.txt, checking missing fields, types and device identity.
Continue or roll back	Accept cloud integration only when transmitted messages and platform-parsed values agree; Debug alone proves processing only.

Related tutorials: T04 Messages, property paths and types / T10 Convert to/from JSON

Action 4. Open a new observation window and send an invalid type followed by non-JSON text. Expect no new valid processed message and a validation or parsing warning.

S22 | Test JSON errors separately from numeric-type errors

Check	Explanation and Action
Expected behavior	invalid_type.json is rejected by the Function; INVALID fails JSON parsing; neither produces a new valid processed message.
Operating precautions	Send one invalid case at a time and record the time. Do not replace a failed payload with the previous value.
Symptoms and ordered checks	If output appears, check time, other publishers and duplicate tabs. Inspect JSON errors with a scoped Catch -> Debug; inspect node.warn as a warning. Send valid 255 afterward to prove recovery.
Files, acquisition and what to inspect	Inspect supplied invalid_type.json. Save S22_json_error.png and S22_type_warning.png with their distinct source nodes; retain S22_recovered.json showing 25.5 after recovery.
Continue or roll back	Pass after both invalid cases are blocked and valid input recovers. Remove temporary Debug nodes and deploy to avoid continued disclosure.

Related tutorials: T10 Convert to/from JSON / T11 Handling errors, Catch and Status

```
python site_lab.py publish --topic plant/site1/raw --json-file invalid_type.json
python site_lab.py publish --topic plant/site1/raw --payload INVALID
```

7.2 What Must Change for the Actual Platform

Item	Customer-Supplied Information	Change and Verification
Raw payload	One complete, real, sanitized JSON sample	Inspect the complete Debug message and copy its actual field path
Scale and unit	Point list, raw unit, and platform unit	Do not multiply an existing engineering value of 25.5 by 0.1 again
Output structure	Required device ID, field names, and time format	Build an adapter; do not assume the platform accepts this sample schema
Message identity	ID, duplicate handling, and acknowledgements	QoS is not application deduplication or acceptance
Broker and TLS	Address, port, account, CA, and permissions	Verify on approved test topics before requesting production cutover

Restoration: Disable N02, deploy, and restore original topics and configuration. Restoring only PC commands does not remove the lab broker address from the gateway.

S23 | Map the lab schema to the actual customer platform

Check	Explanation and Action
Expected behavior	Registered device, topic, fields, units and time semantics match the actual payload, and the test message is traceable on the platform.
Operating precautions	Use an approved test device/topic first. N02 outputs only temperature and unit; deviceId, time or platform envelopes need explicit adapters. It is not a universal cloud connector.
Symptoms and ordered checks	Device online but no point: 1. Inspect platform raw reception. 2. Verify device ownership. 3. Check envelope/fields/types. 4. Check timestamp units and query range. If nothing was received, return to broker publication/ACL.
Files, acquisition and what to inspect	Obtain platform_api.pdf or connection documentation, sanitized payload_actual.json and point_list.csv. Request S23_ingest.txt/parser records from the platform administrator. Compare paths, scale, device ID and timezone without sharing secrets.
Continue or roll back	Schedule production cutover only after actual reception/parsing passes. Preserve existing business and roll back test settings on failure.

Related tutorials: T04 Messages, property paths and types / T09 Publish messages to a topic

8 N03: Combine Temperature and Running State

Customer task: Report alarm state only when the machine is running and temperature is greater than 30. Use `03_combined_alarm.json`, corresponding to E2. temperature is already an engineering value; do not scale by 0.1 again.

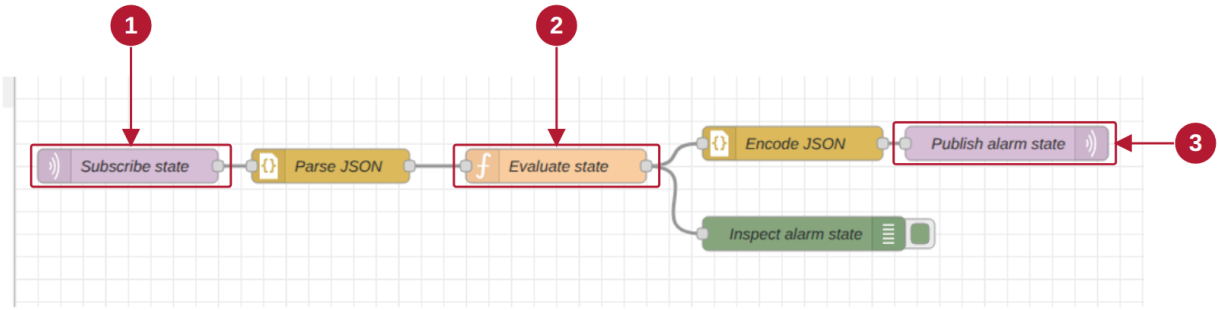


Figure 8-1 N03 produces an alarm state for every valid input; no pump or valve control output is present.

Callout	Action and Check
1	Input temperature and run from the same valid state object.
2	This case tests temperature>30 and run=1, returning a state for every valid input.
3	Output alarm=true/false; no SMS, DO, pump or valve command is sent.

Field or Node	Setting	Rule
Input topic	plant/site1/state	JSON temperature is finite numeric and run is 0 or 1
Condition	temperature greater than 30 and run equal to 1	Exactly 30 does not satisfy this case
Output topic	plant/site1/alarm	alarm is Boolean true/false, not a string
Invalid input	Missing field, string, or another run value	Reject it, inspect the reason, and do not reuse an old result

8.1 Three Business States and One Invalid Input

Import 03, enter broker authentication, and deploy. Subscribe first and keep the subscription running while publishing in order.

```
python site_lab.py watch --topic plant/site1/alarm --seconds 60
python site_lab.py publish --topic plant/site1/state --json-file alarm_hot.json
python site_lab.py publish --topic plant/site1/state --json-file alarm_normal.json
python site_lab.py publish --topic plant/site1/state --json-file machine_stopped.json
python site_lab.py publish --topic plant/site1/state --json-file invalid_type.json
```

Input	Expected Output	Check
temperature=35, run=1	alarm=true	Valid high temperature and a running machine
temperature=25, run=1	alarm=false	Normal temperature
temperature=35, run=0	alarm=false	Stopped state does not satisfy the condition
temperature is string 35	No new valid output	Do not silently accept a string as a measurement

S24 | Check each combined condition using engineering values

Check	Explanation and Action
Expected behavior	35/1 -> true, 25/1 -> false, 35/0 -> false; string 35 is rejected. Exactly 30 does not trigger N03.
Operating precautions	temperature and run belong to the same valid state object. Separate-topic samples from different times are not automatically one coherent state. N03 uses engineering values without further 0.1 scaling.
Symptoms and ordered checks	High value but no alarm: inspect numeric run=1, temperature path/unit, then >30. Alarm while stopped: check run type and old messages. Repeated true output is normal N03 behavior; use Chapter 9 for deduplication.
Files, acquisition and what to inspect	Compare alarm_hot.json, alarm_normal.json, machine_stopped.json and invalid_type.json. Save four sends/outputs in S24_alarm_sequence.txt with order and source node.
Continue or roll back	Check all four cases before adding a notification channel. Acceptance, delivery and equipment action require separate checks.

Related tutorials: T04 Messages, property paths and types / T13 Drop messages that have not changed value

This case reports state for every valid input; it does not deduplicate notifications. Repeated 35/1 inputs repeatedly output true. Connect SMS, email, business messaging, or annunciators only through separately approved channels. Do not claim that this example sends SMS or drives DO.

Restoration: Disable N03, check for other subscribers to lab topics, and restore the project's approved alarm rules.

9 N04: Hysteresis, Deduplication, and Recovery

Customer task: Avoid repeated alarms near the threshold and emit one recovery event. Use `04_hysteresis_events.json`. Unlike N03, this case raises at 30 and clears at 28.

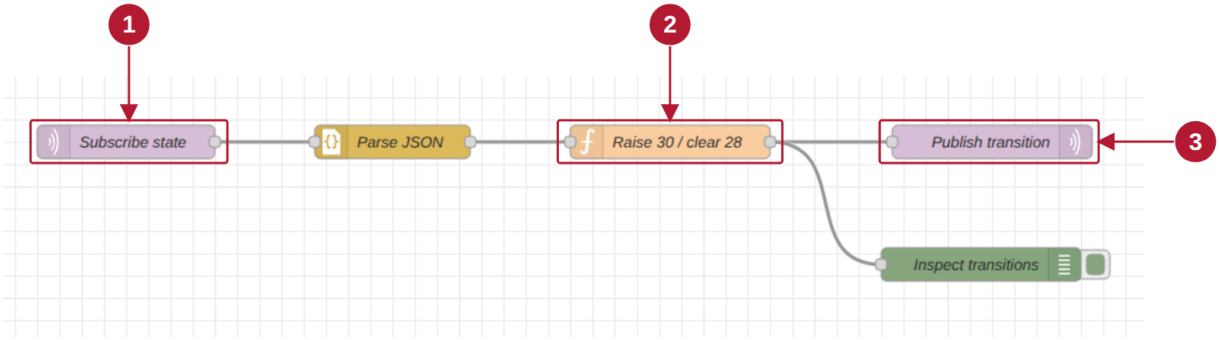


Figure 9-1 N04 stores alarm state in the Function and outputs only an initial state or a state transition.

Callout	Action and Check
1	Send engineering values 27, 31, 32, 29, 28 in sequence, all with run=1.
2	Raise at 30 and clear at 28. Keep state in the deadband; memory context does not survive a restart.
3	Output only initial/raised/cleared state events, not repeated notifications.

Parameter	Example Setting	Meaning
Input topic	plant/site1/hysteresis/in	Engineering temperature and run
Raise / clear	Raise at temperature >=30; clear at <=28	Keep the previous state between 28 and 30
Stopped condition	Clear when run=0	The process owner must decide whether this is correct for the actual project
Output topic	plant/site1/hysteresis/event	Only an initial valid state or subsequent transition
Event type	initial, raised, cleared	initial may be true or false; it is not recovery of a historical alarm
State storage	Function-node context	Memory state does not guarantee survival across a restart

9.1 Verify a Continuous Sequence

Import 04 and configure the broker. Subscribe first, then publish the sequence below. temperature is numeric and run is 1 in each sample.

```
python site_lab.py watch --topic plant/site1/hysteresis/event --seconds 120
python site_lab.py publish --topic plant/site1/hysteresis/in --json-file hysteresis_27.json
```

Repeat the publish command with hysteresis_31.json, hysteresis_32.json, hysteresis_29.json, and hysteresis_28.json, once per step. Supplied JSON files avoid quoting differences between Windows shells.

Input Sequence	Expected Event	Avoid Misreading
27/1	initial, alarm=false	Initial state, not a historical clear event
31/1	raised, alarm=true	Normal-to-alarm transition
32/1	No new event	Unchanged state is deduplicated
29/1	No new event	Remain in alarm within the deadband
28/1	cleared, alarm=false	Reach the clear threshold
28/1 again	No new event	Do not repeatedly send recovery

S25 | Test the hysteresis sequence as events, not per-message output

Check	Explanation and Action
Expected behavior	In one continuous run: 27 -> initial(false), 31 -> raised(true), 32/29 -> none, 28 -> cleared(false), repeated 28 -> none.
Operating precautions	Do not restart, recreate the Function or clear context during this sequence. Use run=1 throughout. N04 raises at 30 and clears at 28, unlike N03's strict >30.
Symptoms and ordered checks	No output at 29 is expected. Repeated initial suggests recreated nodes/deployment/multiple instances. No raised at 31 may mean it was already active. No clear at 28: inspect engineering value, run and actual input arrival.
Files, acquisition and what to inspect	Send hysteresis_27/31/32/29/28.json individually. Save order, input times and output event in S25_hysteresis.txt and compare thresholds/context code with 04_hysteresis_events.json.
Continue or roll back	Pass when initialization, raise, hold and clear match, not when every input produces an output.

Related tutorials: T12 Context scope and storage / T13 Drop messages that have not changed value

9.2 Deduplication Is Not a Complete Alarm System

This example does not provide persistent alarm history, operator acknowledgement, approved clearing, notification retries, or deduplication across restarts. Default context is in memory. The first valid input after restart creates a new initial event. Project requirements need a separately verified persistence, notification, and audit solution.

Restoration: Disable N04 and deploy. Do not reset the entire Node-RED environment to clear one lab state. Manage production alarm history in the approved project system.

S26 | After restart, initial is not historical recovery

Check	Explanation and Action
Expected behavior	Default memory context is lost on runtime restart; the next valid input produces initial rather than restoring historical alarm state.

Check	Explanation and Action
Operating precautions	Production alarms need explicit acknowledgement, clearing, notification retry and persistence. Suppressing initial blindly can hide an already-high condition.
Symptoms and ordered checks	Reported repeated alarm after restart: match startup and initial times, then verify context design. If duplicates occur without a restart, check duplicated flows, notification delivery and state keys. Do not erase production history.
Files, acquisition and what to inspect	Obtain startup time/log in S26_restart.txt from the administrator and retain event times in S26_events.txt. Only an authorized administrator checks contextStorage in the actual settings.js, whose path comes from startup logs.
Continue or roll back	Record the documented restart behavior. Approve and separately test storage/notification design for restart-resilient alarms rather than experimenting in production.

Related tutorials: T12 Context scope and storage

10 N05: Detect Missing Data and Recovery

Customer task: Distinguish a still-displayed old value from fresh data when an instrument or acquisition path stops reporting. Use `05_timeout_recovery.json`. It emits state messages only, not restart or start/stop commands.

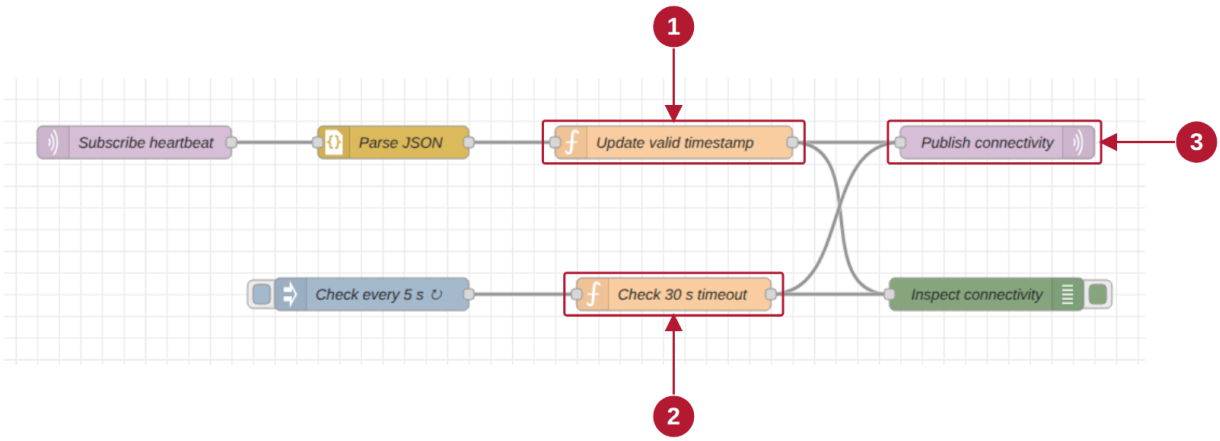


Figure 10-1 N05 timestamps valid messages and independently checks a 30-second timeout for this one data source.

Callout	Action and Check
1	Only valid numeric messages update the timestamp; invalid data cannot restore healthy state.
2	Check the 30-second limit every 5 seconds. Wait for the first valid sample at startup.
3	Check one timeout and one recovery event on the connectivity topic.

Parameter or Behavior	Example Definition	Limitation
Input topic	plant/site1/heartbeat	Only valid JSON with numeric temperature updates reception time

Parameter or Behavior	Example Definition	Limitation
Check period	Every 5 seconds	Not continuous millisecond detection; timeout may be detected at approximately 30-35 seconds
Timeout	30000 milliseconds	A lab value, not a universal instrument requirement
Event topic	plant/site1/connectivity	offline=true for timeout; false for recovery
No data at startup	Wait for the first valid sample	No immediate offline event merely because the runtime has no history
Multiple devices	This case covers one source	Maintain lastSeen separately by device ID, not one shared timestamp

10.1 Verify Healthy, Timed Out, and Recovered

Action 1. Import 05, configure the broker, deploy, and subscribe to connectivity.

S27 | Deploy the timeout flow and wait for the first valid sample

Check	Explanation and Action
Expected behavior	Before a valid sample arrives, the check node waits for the first valid sample rather than immediately emitting offline=true.
Operating precautions	N05 updates lastSeen only with valid messages. Keep the 5-second Inject check; importing only the upper message path omits timeout detection.
Symptoms and ordered checks	No timeout: first establish a valid temperature was received, then inspect the timer, wire and deployment. An immediate startup alarm suggests another flow or altered initialization.
Files, acquisition and what to inspect	Inspect inject.repeat=5, lastSeen and offline logic in 05_timeout_recovery.json. Save S27_waiting.png showing the lower check-node status in Figure 10-1, not only MQTT connection status.
Continue or roll back	Send the first valid sample after verifying initialization and the timer path.

Related tutorials: T14 Trigger when a message is not received

Action 2. Send one valid heartbeat and another approximately 10 seconds later. Expect no timeout; the Function status shows seconds since the latest valid reception.

S28 | Check freshness while valid input continues

Check	Explanation and Action
Expected behavior	Two valid samples about 10 seconds apart update reception age without a timeout on connectivity.
Operating precautions	This age is elapsed time since gateway reception, not measurement time; invalid_type.json must not refresh the valid-sample timestamp.

Check	Explanation and Action
Symptoms and ordered checks	Age increases while publishing: inspect actual input topic/type, then Function warnings. Mixed devices masking a failure require per-device state; this single-source lastSeen cannot monitor multiple devices.
Files, acquisition and what to inspect	Retain both samples/times in S28_heartbeat.txt, node age in S28_status.png and the original payload/type in S28_input.json.
Continue or roll back	Start the stop-sending test after valid input refreshes age and invalid input does not.

Related tutorials: T04 Messages, property paths and types / T05 Debug sidebar and complete messages

Action 3. Stop sending for more than 35 seconds. Expect one `{"offline":true,"event":"timeout"}`; further silence must not produce repeated timeout events.

S29 | Stop for more than 35 seconds and check one timeout

Check	Explanation and Action
Expected behavior	One timeout appears about 30-35 seconds after valid data stops, with no repeated event every 5 seconds.
Operating precautions	Observe for more than 35 seconds. Do not disconnect a combined power/data link or power-cycle production equipment to simulate missing data.
Symptoms and ordered checks	No timeout: look for another publisher, then timer/lastSeen. Repeated timeouts: inspect offline state, recreated nodes and duplicate tabs. A timeout proves stale data, not a failed cellular modem or instrument.
Files, acquisition and what to inspect	Save independent watch output as S29_timeout.txt with last valid input time, timeout time and ageMs. Use Chapter 11 acquisition/broker evidence to diagnose the actual source.
Continue or roll back	Send valid recovery data after confirming a single timeout. Do not attach the event to unapproved restart/control outputs.

Related tutorials: T11 Handling errors, Catch and Status / T14 Trigger when a message is not received

Action 4. Send valid data again. Expect one `{"offline":false,"event":"recovered"}`. Invalid JSON or a string-valued temperature must not restore healthy state.

S30 | Recover on valid data, not merely a green connection

Check	Explanation and Action
Expected behavior	The first valid temperature after timeout produces one recovered/offline=false event; subsequent valid samples do not repeat it.
Operating precautions	Invalid JSON, strings and MQTT reconnection alone are not recovery. Record runtime, broker connection and source state separately.
Symptoms and ordered checks	Connected but no recovered: 1. Check fresh source data. 2. Inspect type warnings. 3. Confirm a prior offline=true state. 4. Check for restart/state loss. Do not inject fake production heartbeats to hide a source failure.

Check	Explanation and Action
Files, acquisition and what to inspect	Retain timeout, valid recovery input and recovered in S30_recovery.txt. Obtain S30_restart.txt if a runtime restart must be excluded. Record connection and valid-data recovery times separately in Customer_Test_Record.csv.
Continue or roll back	Pass when recovery matches a valid new input; disable/deploy the timer-based lab tab afterward.

Related tutorials: T11 Handling errors, Catch and Status

```
python site_lab.py watch --topic plant/site1/connectivity --seconds 120
python site_lab.py publish --topic plant/site1/heartbeat --json-file
engineering_temperature.json
```

A data timeout does not identify whether the instrument, PLC power, acquisition service, broker, or cellular link failed. Continue checking the path. MQTT disconnection does not necessarily trigger Catch; use Status for connection-state changes and Catch for catchable errors. They are not interchangeable.

Restoration: Disable N05 and deploy so its five-second timer no longer runs. Remove lab topics and notification routes under the approved policy.

11 Bring Modbus and PLC Data Into Node-RED

Customer task: Read instruments or PLCs and reuse the processing examples. Prefer the device's already-delivered native acquisition functions followed by MQTT. This does not assume that a third-party Modbus node is installed.

11.1 Verify Acquisition Before Processing

Start with the read-only PC simulated slave M1. The path is native Modbus acquisition > native MQTT reporting > Node-RED subscription > payload adaptation > N02/N03 processing. Use a delivered PLC acquisition module for proprietary PLC protocols; Modbus success does not prove all PLC compatibility.

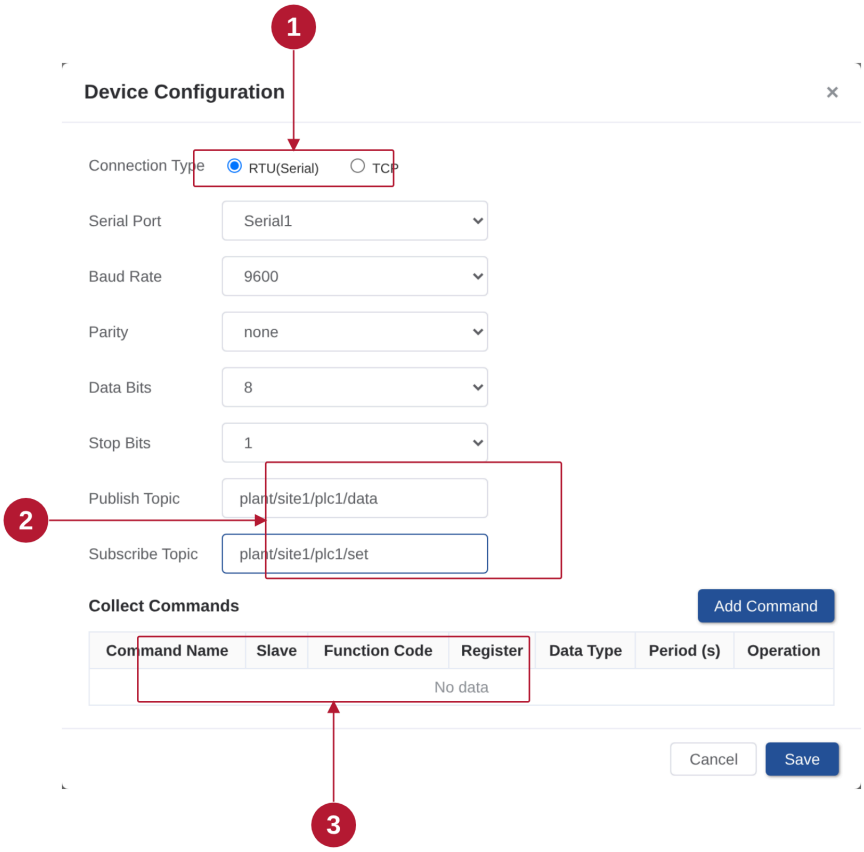


Figure 11-1 Native device configuration reference currently showing RTU. Select TCP for the PC simulated-slave exercise; do not copy the serial settings.

Callout	Action and Check
1	The reference shows RTU; select TCP for the PC simulated-slave exercise.
2	Set acquisition topics for the project; keep writes disabled for the read-only exercise.
3	Check unit 1, FC03, address 0, quantity 1 and UInt16 in the acquisition command.

Device or Command Field	M1 Lab Setting	Confirm in the Field
Connection type	TCP	RTU shown in the reference is not the selection for this exercise
IP and port	192.168.10.10; 1502	Use the real instrument's port for production
Slave / unit ID	1	Confirm TCP Unit ID or RTU slave address
Function code	FC03	Read holding registers only in this exercise
Start address / quantity	0; 1	Zero-based 0 is not a printed reference such as 40001 entered literally
Data type	UInt16	Confirm byte order, sign, and floating-point word order from the real point list
Name / period	temperature; 5 seconds	Lab naming and timing only
Write permission	Off	Address 0 is read-only with initial value 255, not a control target

Device or Command Field	M1 Lab Setting	Confirm in the Field
Uplink topic	plant/site1/plc1/data	Subscribe and inspect the complete actual payload first

Action 1. Run `python site_lab.py modbus-get --host 192.168.10.10` and expect values [255]. This proves only the PC simulated slave.

S31 | Read one simulated register first

Check	Explanation and Action
Expected behavior	The PC modbus-get returns values=[255], proving only the simulated slave, not gateway acquisition of a real instrument.
Operating precautions	Use TCP1502, unit 1, FC03, address 0, quantity 1. Use the actual register map for a real instrument; printed 40001 is not automatically the zero-based address.
Symptoms and ordered checks	Connection failure: check S02 service/port. Exception response: check function, address and quantity. Wrong returned value: inspect signedness, byte/word order and scale. Never probe production registers with writes.
Files, acquisition and what to inspect	Save S31_modbus_read.txt. For a real instrument obtain manufacturer_register_map.pdf and point_list.csv with printed/protocol addresses, type, ordering, scale and read-only access.
Continue or roll back	Configure the gateway after simulated read-only 255 is proven; field acceptance separately needs a contemporaneous instrument reading.

Related tutorials: T15 Community Modbus node and maintainer examples

Action 2. Save and read back the native device connection, command, and reporting settings before starting read-only acquisition. Check the native acquisition MQTT configuration separately from the Node-RED broker configuration.

S32 | Save native acquisition without serial-port contention

Check	Explanation and Action
Expected behavior	Choose TCP for M1 rather than the RTU reference in Figure 11-1. Read back TCP address, unit, command and read-only settings, then poll every 5 seconds.
Operating precautions	For real RS485 follow model wiring/instrument parameters and isolate power before wiring. Use one approved serial owner, not native acquisition, DTU and third-party nodes simultaneously. YZW900 has no serial port.
Symptoms and ordered checks	No instrument response: 1. Check power and both ends' RS485 markings. 2. Check address/baud/parity/stop bits. 3. Check port ownership. 4. Then inspect register/timeout. Do not assume A/B equals +/- or enable writes to troubleshoot.
Files, acquisition and what to inspect	Capture complete Web fields as S32_device.png/S32_command.png and photograph actual terminals for field work. Download diagnostics only where the delivered Web provides it; otherwise request S32_acquisition.txt with timeouts/exception codes and times.
Continue or roll back	Continue after correct read-only settings, no port contention and evidence of current raw acquisition.

Related tutorials: T15 Community Modbus node and maintainer examples

Action 3. Independently subscribe to plant/site1/plc1/data and verify 255 actually published by the device. A PC publish command must not impersonate a successful device-acquisition test.

S33 | Verify native acquisition publication, not a PC substitute

Check	Explanation and Action
Expected behavior	An independent subscription to plant/site1/plc1/data receives the device-acquired 255 in its actual structure, not a manual PC publish.
Operating precautions	Verify native acquisition's MQTT client separately from Node-RED; both must use the intended broker with the required publish/subscribe permissions.
Symptoms and ordered checks	Acquisition works but topic is empty: check native publication enable/cycle, broker and publish ACL. Topic populated but Node-RED empty: check its broker/topic/subscribe ACL. Do not change register addresses to fix publication.
Files, acquisition and what to inspect	Save actual subscribed payload as S33_native_uplink.json and timestamps in S33_watch.txt, plus native publication settings in S33_publish.png. Request S33_broker.txt for publish/subscribe rejections.
Continue or roll back	Connect processing only after native acquisition source, topic and current 255 are correlated.

Related tutorials: T08 Subscribe to a topic / T09 Publish messages to a topic

Action 4. Subscribe to the same topic in Node-RED. Temporarily connect Debug with complete-message output, confirm the actual path/type, then add JSON, adaptation, and processing.

S34 | Adapt actual payload paths without replacing missing fields with zero

Check	Explanation and Action
Expected behavior	Find the actual temperature path, type and identity in the complete message, then map it to payload.temperature for processing.
Operating precautions	payload.data.temperature differs from payload.temperature. Select array values by point identity, not automatically the first element. Do not repeat an upstream scale.
Symptoms and ordered checks	Cannot read properties/undefined: inspect complete Debug before the adapter, then validate parent/type before access. For a tenfold error inspect scale ownership. Reject missing fields with quality evidence, not zero or stale values.
Files, acquisition and what to inspect	Retain S34_before.json, S34_after.json and point_list.csv; inspect the adapter Function in flows_after.json. If JSON contains only payload, record topic, identity, type and timestamp semantics in S34_metadata.txt.
Continue or roll back	Pass after identity/path/type/scale/unit agree and a deliberately missing field is rejected.

Related tutorials: T04 Messages, property paths and types / T10 Convert to/from JSON

11.2 Adapt Payloads That Differ from the Template

Actual Payload	Adaptation	Verification
payload.temperature is numeric 255	Connect directly to N02 validation/scaling	Independent processed output is 25.5
payload.data.temperature is numeric 255	Map it to payload.temperature in a separate adapter	Validate data and type first; do not silently substitute zero
An array or device/point hierarchy	Select the named point and retain device identity	Do not assume that the first array element is correct
Acquisition already applies scale 0.1	Use an engineering-value flow without N02 scaling	Avoid changing 25.5 to 2.55
Missing or invalid field	Reject and record bad quality	Do not present the previous reading as a new measurement

For PLC projects, obtain the PLC model, driver/protocol, network or serial settings, point addresses, types, access protection, permissions, and actual payload samples. Figure 11-2 identifies interface locations only. Bool and DB1.0 are reference-form values, not settings for this temperature example.

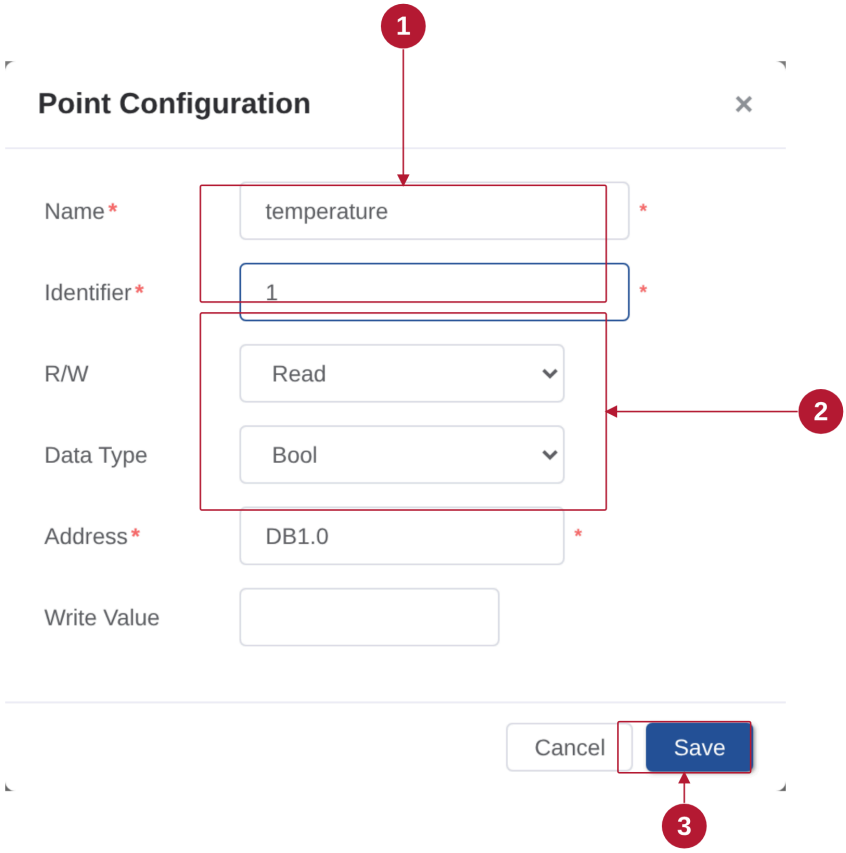


Figure 11-2 Native PLC point dialog: name, identifier, access, data type, and address must come from the project point list.

Callout	Action and Check
1	Take name and identifier from the project point list; do not invent a temperature point from this screenshot.
2	Match access, type and address to the real PLC. Bool/DB1.0 are only reference-form values.
3	Save after checking and then verify real acquisition. Saving a form is not business acceptance.

Assign each RTU serial port to one defined service. Native acquisition, DTU, and third-party serial/modbus nodes must not compete for it. YZW900 has no serial port. YZW900L internal RS485 also requires qualified access and resealing. Approve an exact package/version, architecture, and Node.js dependency set before using third-party direct acquisition; do not try the latest package on a production unit.

S35 | Do not infer PLC or third-party Modbus compatibility

Check	Explanation and Action
Expected behavior	The PLC driver/point list, read permission and protection settings are approved. Native acquisition to MQTT is the preferred path here.
Operating precautions	Figure 11-2 Bool/DB1.0 only illustrate the native form. At review, T15 requires Node.js >=22 and Node-RED >=4; the 3.0.2 captures do not prove latest-package compatibility.
Symptoms and ordered checks	PLC connected but wrong point: inspect CPU/driver, memory area/address, type and protection. Unknown nodes/engines errors on third-party examples: stop installation and ask the supplier to verify pinned version, architecture and dependencies.
Files, acquisition and what to inspect	Obtain plc_point_list.csv and CPU/engineering-software versions. Record exact packages/versions/sources in nodes_inventory.csv; save install errors as S35_dependency.txt and ask the administrator for inaccessible runtime details.

Check	Explanation and Action
Continue or roll back	Continue only with confirmed delivered drivers/nodes and actual point maps. Unconfirmed functionality remains a reference, not device support.

Related tutorials: T15 Community Modbus node and maintainer examples / T20 Palette Manager and dependencies

12 N06: Normalize Decoded LoRaWAN Uplinks

Customer task: Send already-decoded sensor data to a common platform. Apply this only to delivered LoRaWAN/LNS projects. YZW300L/YZW900L local-LNS solutions differ from YZW300X external-LNS forwarding; YZW300X is not documented as an on-device Node-RED model.

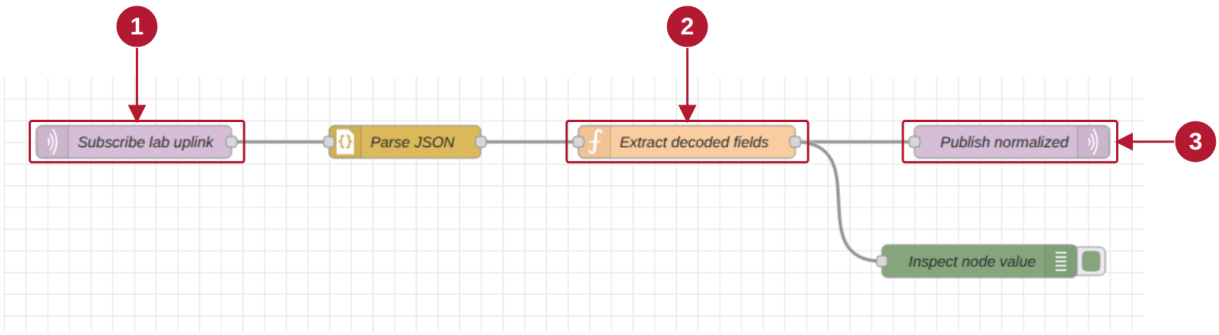


Figure 12-1 N06 reads decoded object fields only. It does not join a LoRaWAN device or send RF downlinks.

Callout	Action and Check
1	The lab topic simulates decoded uplinks; it does not prove joining or RF transmission.
2	Extract deviceInfo.devEui and object.temperature; do not treat Base64 as an engineering value.
3	Verify device identity, 25.5 and unit C in the normalized output.

Item	Isolated Lab	Actual Project
Input topic	application/demo/device/demo-001/event/up	Use the actual LNS Application ID, Device EUI, and topic template
Field paths	deviceInfo.devEui; object.temperature	Confirm from a sanitized real uplink
Temperature	Engineering value 25.5	Do not apply N02's 0.1 scale again
Output topic	plant/site1/lorawan/normalized	Approved platform topic and schema
Output	deviceId, temperature, unit	Extract required fields only; do not automatically forward secrets or all metadata

Action 1. Import 06 and configure the lab broker. Subscribe to output before publishing the simulated decoded message.

S36 | Simulate only decoded LoRaWAN uplinks

Check	Explanation and Action
Expected behavior	Subscribe to normalized before publishing lorawan_uplink.json, containing deviceInfo.devEui and object.temperature.
Operating precautions	This teaching sample does not prove joining, RF reception or actual LNS decoding.
Symptoms and ordered checks	No lab output: check broker/input topic, then JSON parsing and paths. A data/Base64 field requires a device protocol and matching codec, not an invented temperature.
Files, acquisition and what to inspect	Review supplied lorawan_uplink.json; save input Debug as S36_lns_input.json and mark simulated/actual provenance. Exclude AppKey, NwkKey and authentication secrets.
Continue or roll back	Validate the real RF/LNS path only after the simulated source and output chain are understood.

Related tutorials: T04 Messages, property paths and types / T08 Subscribe to a topic

```
python site_lab.py watch --topic plant/site1/lorawan/normalized --seconds 60
python site_lab.py publish --topic application/demo/device/demo-001/event/up --json-file
lorawan_uplink.json
```

Action 2. Expect deviceId=demo-001, temperature=25.5, and unit=C. Reject missing object.temperature or a string-valued temperature.

S37 | Keep device identity and verify engineering values

Check	Explanation and Action
Expected behavior	Output has deviceId=demo-001, temperature=25.5 and unit=C. Temperature is already engineering-scaled.
Operating precautions	Retain each device identity. RSSI, SNR and raw battery bytes are not temperature, and all messages must not be assigned one fixed deviceId.
Symptoms and ordered checks	2.55 suggests repeated scaling; different sensors overwriting one point suggests identity mapping. One sensor missing output requires its decoded path/type, not universal reuse of another model's codec.
Files, acquisition and what to inspect	Save S37_decoded.json and independently received S37_normalized.json; compare unit, protocol version and decoded field against sensor_protocol.pdf.
Continue or roll back	Pass after identity/decoded meaning agree and string/missing-field cases are rejected.

Related tutorials: T04 Messages, property paths and types / T10 Convert to/from JSON

Action 3. Before switching to the real LNS, verify gateway heartbeat, joining, uplink events, and decoding. This simulated publish does not prove an actual RF transmission.

S38 | Trace field heartbeat, join, uplink and decoding in order

Check	Explanation and Action
Expected behavior	The actual LNS shows gateway heartbeat, valid terminal uplinks and decoded object data; Node-RED receives that device event.
Operating precautions	Separate 300L/900L local LNS from 300X external LNS; this does not give 300X an on-device Node-RED runtime. Confirm region, band and key handling from project records.
Symptoms and ordered checks	No gateway heartbeat: check gateway-to-LNS. Heartbeat without uplink: check terminal power, region and joining. Uplink without object: inspect codec. Object present but no Node-RED input: inspect LNS MQTT topic/account/subscribe permission.
Files, acquisition and what to inspect	Copy sanitized S38_uplink.json from the actual LNS event page and capture S38_events.png with DevEUI, time and event type. Obtain codec/version from the administrator; keep keys separate.
Continue or roll back	Record field acceptance only after the same actual RF event is traced through decoding, MQTT and platform value.

Related tutorials: T08 Subscribe to a topic

If data contains only Base64, a matching endpoint Codec is still required. Receiving a message in Node-RED does not establish its sensor meaning. Do not interpret RSSI, SNR, counters, or battery bytes as temperature. Maintain decoders by endpoint model, protocol version, and byte order rather than generalizing from one sensor.

13 N07: Provide a Read-Only HTTP Query

Customer task: Let a host fetch the latest valid temperature and determine whether it has expired. Use `07_readonly_http_api.json`. The MQTT and HTTP paths must stay on the same example tab to share its flow cache.

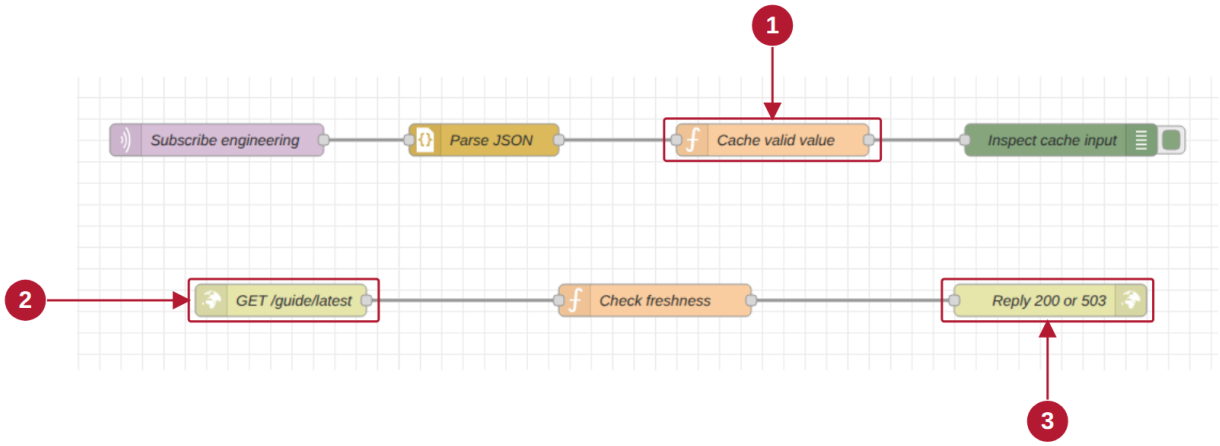


Figure 13-1 N07: MQTT updates a cache; HTTP GET returns the current value and freshness.

Callout	Action and Check
1	Valid engineering values update this tab flow cache. Reception time is not measurement time.
2	The host calls GET /guide/latest; replace the base URL with the delivered entry.
3	Return 200 for fresh data and 503 for no data or age at least 30 seconds; read ok and ageMs.

Parameter	Example Setting	Boundary
Input topic	plant/site1/api/input	Engineering temperature 25.5
Method / path	GET /guide/latest	The delivered HTTP service or reverse proxy defines the actual base URL
Lab URL	http://192.168.10.1:1880/guide/latest	A common-port lab example, not a replacement for the supplied entry
Validity	Less than 30 seconds since reception	Reception time is not sensor measurement time
No data / expired	HTTP 503, ok=false	Do not return a fixed 200 to pretend stale data is fresh
Fresh data	HTTP 200, ok=true	data contains temperature, unit, receivedAt; ageMs is reception age

13.1 Three Verification Steps

Action 1. Import 07, configure the broker, and deploy. Query before any sample arrives and expect 503; this alone does not mean the HTTP service is broken.

S39 | Distinguish initial HTTP 503 from an unreachable service

Check	Explanation and Action
Expected behavior	With no cache after startup, GET returns 503, ok=false and data=null: the endpoint responded but has no valid temperature.
Operating precautions	Use the delivered business HTTP base URL. A proxied editor path is not automatically the business prefix. MQTT cache and query nodes must remain in the same flow tab.
Symptoms and ordered checks	Refused/timeout: check entry/service. 404: check GET, route and deployment. Hanging request: inspect all HTTP Response paths and preserved msg.req/res. Only then diagnose 503 cache state; do not force 200.
Files, acquisition and what to inspect	Save curl.exe -i headers/status/body as S39_initial_http.txt. In flows_after.json inspect http in method/url, http response and the tab containing flow.get/set. Strip authentication headers before sharing.
Continue or roll back	Send valid temperature after confirming a bounded no-data 503 rather than a hanging request.

Related tutorials: T16 Create an HTTP endpoint / T06 Writing Functions

Action 2. Publish an engineering value and query immediately. Expect 200, temperature=25.5, and ageMs below 30000.

S40 | Check 200 and ageMs immediately after input

Check	Explanation and Action
Expected behavior	Immediately after publishing 25.5, GET returns 200, ok=true, data.temperature=25.5 and ageMs from 0 to 29999 ms.
Operating precautions	receivedAt is Node-RED reception time, not measurement time; flow cache is not a persistent database. A query copied to another tab does not share the original flow state.
Symptoms and ordered checks	Persistent 503/data=null: inspect valid api/input and matching cache/query tabs. Data present with large ageMs: check actual updates and recent clock changes. Wrong values require S34 adaptation/scale checks.
Files, acquisition and what to inspect	Retain S40_input.json/S40_http_200.txt and publish/GET times. In flows_after.json verify matching z values for n07_cache and HTTP processing, and a consistent latest key.
Continue or roll back	Test expiry only after value, status and age match; 200 with stale data is not a pass.

Related tutorials: T12 Context scope and storage / T16 Create an HTTP endpoint

Action 3. Stop publishing and query after 30 seconds. Expect 503. An old value may remain for diagnosis; the host must read ok and status, not only the old temperature.

S41 | Make the host recognize expired data

Check	Explanation and Action
Expected behavior	After 30 seconds return 503/ok=false, potentially with old diagnostic data that is no longer a valid live reading.
Operating precautions	The host must evaluate status and ok before extracting values. Do not read only data.temperature or treat a cached old 200 as a new response.
Symptoms and ordered checks	Still 200 after stopping: check other publishers, elapsed time and cached responses. Still 503 after valid publishing: inspect MQTT/validation. Negative ageMs can indicate a clock rollback; verify time rather than bypassing checks.
Files, acquisition and what to inspect	Retain S41_http_stale.txt, S41_http_recovered.txt and S41_timeline.csv with input/query times, status, ok and ageMs. Obtain system time from its actual management display or administrator.
Continue or roll back	The host must show expiry and recover on valid new data. Disable/deploy the isolated HTTP route afterward.

Related tutorials: T16 Create an HTTP endpoint

```
python site_lab.py publish --topic plant/site1/api/input --json-file
engineering_temperature.json
curl.exe -i http://192.168.10.1:1880/guide/latest
```

Windows uses curl.exe to avoid a PowerShell alias conflict; other systems can use curl. Replace the example protocol, host, port, and path with the actual delivered business endpoint. If HTTP-node authentication is enabled, provide credentials through the administrator-approved mechanism without placing real passwords in public commands.

13.2 Do Not Publish This Example as a Public API

This example includes no access authentication, rate limit, or persistent cache and is for an isolated lab only. Editor adminAuth and HTTP business authentication are separate controls; signing in to the editor does not protect /guide/latest. Production publication needs authentication, TLS, restricted network access, quotas, and auditing. Preserve msg.req and msg.res through Functions, and return every HTTP branch through HTTP Response to avoid hanging requests.

S42 | Verify editor and business endpoint security separately

Check	Explanation and Action
Expected behavior	Only approved sources can access the endpoint; unauthorized requests are rejected, authorized ones work, and the editor is not publicly exposed.
Operating precautions	N07 itself has no authentication. adminAuth does not automatically protect HTTP nodes. The delivery administrator must configure TLS, endpoint authorization and access scope rather than blindly copying settings.js.

Check	Explanation and Action
Symptoms and ordered checks	External unauthenticated access: suspend exposure, test management/business planes separately, then inspect proxy/HTTP-node protection. For authorized users inspect credentials/roles on 401/403 and hostname/chain/time on TLS errors.
Files, acquisition and what to inspect	Obtain security_handover.pdf from the administrator. Record unauthorized/authorized outcomes in S42_auth_test.txt without tokens. Confirm configuration paths from actual startup logs/handover and exclude real secrets.
Continue or roll back	Publish only after both authorized/unauthorized tests meet policy; otherwise keep it within the isolated lab.

Related tutorials: T21 Securing the editor and HTTP endpoints

14 N08: POST JSON to a Customer HTTP API

Customer task: Use HTTP POST when the platform does not accept MQTT. Use `08_http_post_response.json`. It sends one message only when Inject is clicked and does not automatically retry in a loop.

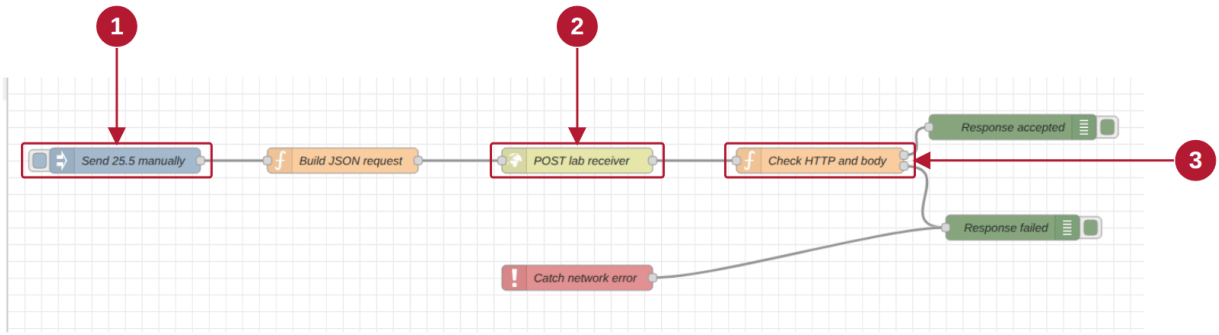


Figure 14-1 N08: build JSON, issue HTTP Request, and check HTTP plus business response. Catch handles network errors.

Callout	Action and Check
1	Inject once manually and confirm the target is the lab receiver, not production.
2	Build JSON and POST with a 10-second timeout; this example has no automatic retry.
3	Check both HTTP status and business ok; network errors must reach a failure branch.

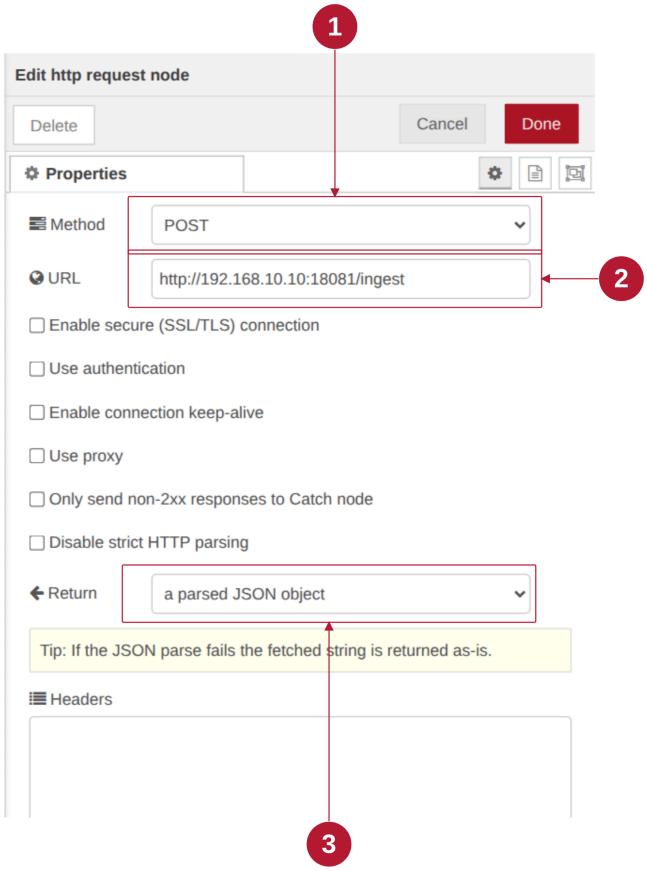


Figure 14-2 Native HTTP Request dialog: set method, URL, and response type for this example.

Callout	Action and Check
1	Set method to POST and parse the response as a JSON object.
2	Use the isolated PC at :18081/ingest. Production URL and authentication are separate handover items.
3	Check response parsing and connection options; do not disable certificate validation to troubleshoot.

Field	Lab Setting	Required from the Platform
Method	POST	The API's required method
URL	http://192.168.10.10:18081/ingest	HTTPS endpoint, path, and region
Content-Type	application/json	Accepted encoding and schema
Request body	deviceId=LAB-001, messageId, temperature=25.5, unit=C	Production identity, fields, and signature rules
Response type	Parsed JSON object	Do not access an unparsed response string as an object
requestTimeout	msg.requestTimeout=10000 milliseconds	Match delivered-node support and the platform latency budget
Success	HTTP 2xx and response ok=true	Actual business code, message ID, and acknowledgement definition

14.1 Check Three Sources of Evidence

Action 1. Keep the Chapter 2 HTTP receiver running. Import 08, verify that its URL targets the isolated PC rather than production, deploy, and click Inject once.

S43 | Verify POST destination and format, then inject once

Check	Explanation and Action
Expected behavior	Figure 14-2 uses POST, http://192.168.10.10:18081/ingest and a parsed JSON response; manual Inject produces one request.
Operating precautions	Use Content-Type=application/json and msg.requestTimeout=10000 ms. Keep the receiver running and do not substitute the linked GET tutorial's method.
Symptoms and ordered checks	Refused: check receiver READY/port. 400: check path/JSON. 405: check method. 422: check temperature type. Hanging: inspect timeout/network. Changing response parsing cannot fix server rejection.
Files, acquisition and what to inspect	Inspect method, url, ret, headers and requestTimeout in 08_http_post_response.json. Save nonsensitive properties in S43_request.png and copy receiver status/errors/time into S43_receiver.txt.
Continue or roll back	Check acknowledgement after confirming a single request to the lab receiver.

Related tutorials: T17 HTTP Request client basics / T19 Set a request header

Action 2. The success Debug should show ok=true. The PC receiver should print RECEIVED with the same messageId and temperature 25.5.

S44 | Correlate request, response and receiver by messageId

Check	Explanation and Action
Expected behavior	HTTP is 2xx, response object ok=true, and RECEIVED contains this request's messageId and temperature 25.5.
Operating precautions	HTTP status, business result and persistence are separate evidence layers. The local receiver prints reception and does not prove cloud storage.
Symptoms and ordered checks	200 routed as failure: inspect whether payload is a string and ok is strictly true. Correct Debug but no platform value: trace messageId through reception/parsing/storage. Login HTML means check URL/authentication, not JSON data.
Files, acquisition and what to inspect	Retain S44_request.json, S44_response.json and S44_receiver.txt. For a real platform ask its administrator for matching reception/storage evidence in S44_platform.txt. Strip Authorization, Cookie and sensitive data.
Continue or roll back	Matching identity/value across all three proves the lab only; production persistence needs independent platform evidence.

Related tutorials: T18 Get a parsed JSON response

Action 3. Stop the receiver and inject again. Expect a network failure/error and no success-path result. Restart the receiver and retest.

S45 | Stop the lab receiver and verify the failure branch

Check	Explanation and Action
Expected behavior	One injection with the receiver stopped produces a network error/failure; one new send succeeds after the receiver is restarted.
Operating precautions	Stop only the PC lab receiver, never the actual platform. This example has no automatic retry; a missing response is not success and must not cause an unbounded tight retry loop.
Symptoms and ordered checks	No Catch: check its scope/wire and inspect errors on the HTTP normal output too. Failure routed as success: inspect a status-only business test. No recovery output: revisit S43 destination/service.
Files, acquisition and what to inspect	Keep error.message, error.source and HTTP status when present in S45_error.txt, then recovery messageId/reception in S45_recovered.txt. Inspect Catch targets and success conditions in flows_after.json.
Continue or roll back	Pass only if failure is not accepted and service recovery is verifiable. Production retry needs bounded attempts, backoff, idempotency and dead-letter policy.

Related tutorials: T11 Handling errors, Catch and Status / T18 Get a parsed JSON response

HTTP 2xx describes interface-level acceptance only; database persistence, downstream business, and physical action depend on platform acknowledgements. Treat HTTP 200 with business failure as failure. Do not retry 401/403 indefinitely. Define retries for 429/5xx or network errors, intervals, limits, and idempotency separately. This example has no reliable retry queue.

Before production use, obtain authentication, token renewal, TLS validation, schema/time format, rate limits, idempotency keys, acknowledgements, and data ownership. Do not put a real token into a Function and publicly export it, or disable certificate validation for troubleshooting.

S46 | Authentication, rate limits and TLS on the actual HTTP platform

Check	Explanation and Action
Expected behavior	Region, HTTPS path, authentication and business acknowledgement follow platform documentation, with a traceable messageId.
Operating precautions	Obtain API documentation, token renewal, idempotency key and maximum rate. Define seconds/milliseconds, timezone and data ownership.
Symptoms and ordered checks	401: check token validity/format. 403: check device/project permissions. 429: check rate limits/Retry-After. For 5xx/timeouts use approved backoff. TLS errors require CA/hostname/clock checks, not disabled validation.
Files, acquisition and what to inspect	Compare platform_api.pdf, controlled certificate handover and nodes_inventory.csv. Save sanitized status/body/time/request ID in S46_http_failure.txt and request the matching server rejection reason.
Continue or roll back	Switch production only after authentication, TLS and business checks pass. Revoke test credentials and restore the original URL on failure.

Related tutorials: T19 Set a request header / T21 Securing the editor and HTTP endpoints

15 Boundaries for Remote Control

None of the eight examples includes real PLC writes, relay DO, or pump/valve start-stop output. alarm=true does not establish that stopping is correct, and HTTP success does not prove actuator movement.

Control Prerequisite	Minimum Definition	Verification Evidence
Permission	Operator, authorized scope, and control window	Reject unauthorized commands
Command structure	commandId, target, point, type, allowed range, and expiry	Reject expired, duplicate, and out-of-range messages
Execution interface	Actual PLC/Modbus/DO protocol and write permissions	Do not infer a command schema from the sample topic
Feedback	Separate received, executed, and physically read-back stages	Broker acknowledgement is not execution read-back
Interlocks	Local emergency stop, process interlocks, manual/auto mode, fail-safe state	Safe behavior after communication failure
Rollback	Flow shutdown, restored logic, and a responsible on-site contact	Closing the browser is insufficient

To learn write/read-back first, use only the documented PC memory simulated slave M2: address 1 starts at 100, write 300, and independently read back. Never transfer M2's address, setpoint 300, or message format to actual equipment without approval. A proprietary PLC write format is not a Modbus write format.

S47 | Keep a read-only boundary before any start/stop control

Check	Explanation and Action
Expected behavior	All eight flows compute/report/query only, without actual PLC writes, DO or pump/valve commands.
Operating precautions	A control project must first approve command authorization, expiry, interlocks, local emergency stop, readback, fail-safe state and field ownership.
Symptoms and ordered checks	Platform success but no action: do not repeat real commands blindly; distinguish platform acceptance, protocol execution and independent readback. Handle an unknown write output under the maintenance plan and notify the field owner.
Files, acquisition and what to inspect	Prepare control_requirements.pdf, approved_point_list.csv, interlock_matrix.pdf and command_ack_spec.pdf. S47_control_review.txt records approval/read-only review without production command secrets.
Continue or roll back	Do not test unapproved real control. Use only a PC in-memory simulator for write exercises, never transfer sample addresses to field equipment.

Related tutorials: T21 Securing the editor and HTTP endpoints

16 Backup, Compatibility, and Maintenance

Customer task: Restore, migrate, and maintain the project rather than retaining only an importable JSON that cannot run.

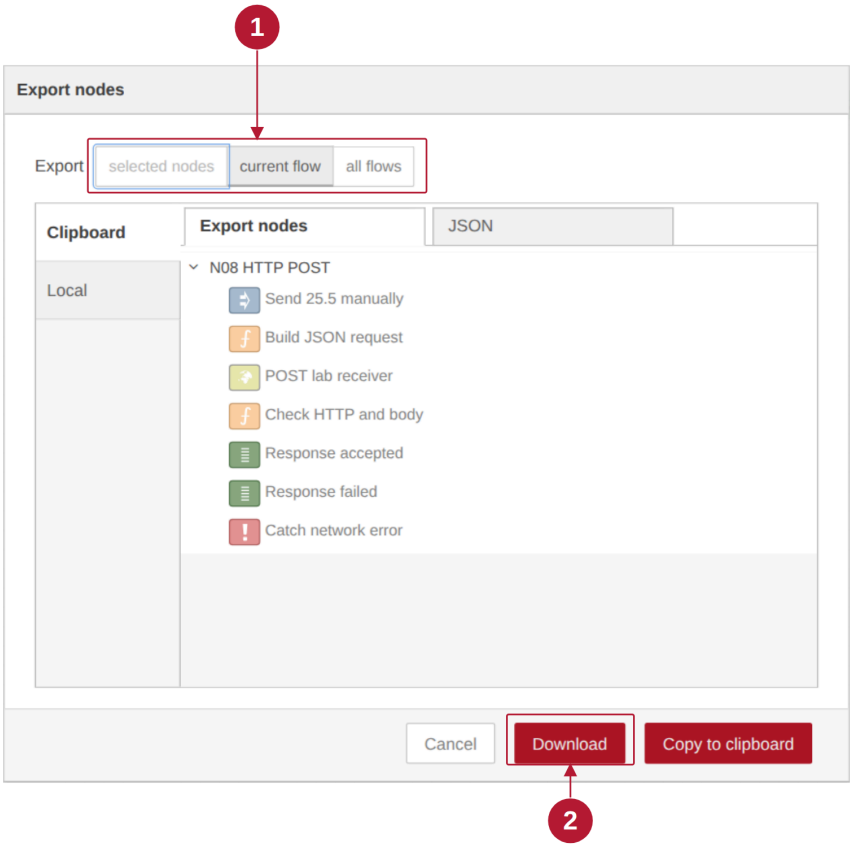


Figure 16-1 Editor Export: choose selected nodes, current flow, or all flows explicitly and record the backup scope.

Callout	Action and Check
1	Choose all flows, current flow or selected nodes explicitly and record the backup scope.
2	Download formatted JSON and verify it is nonempty. Hand over dependencies and credentials separately.

16.1 Keep Three Backup Types Separate

Backup	Contents	Does Not Replace
Editor-exported flow JSON	Tabs, nodes, wires, and configuration references	Restorable secrets, third-party packages, and external files are not guaranteed
Device Export NODE-RED environment	Page-provided node.tar.gz and its version information	Exact coverage follows the handover list; do not assume all histories and certificates are included
Project materials	Locked dependencies, point lists, topics, credential/TLS handover, files, and permissions	Keep passwords and private keys controlled separately from public flows

Include site, device, date, and version in backup names. Confirm that exports are nonempty and parseable. Restore in a spare matching environment, reenter credentials, and verify simulated data before scheduling production changes. Prove rollback first.

S48 | Verify flow, environment and controlled-secret backups

Check	Explanation and Action
Expected behavior	Flow JSON parses, the Web environment package is nonempty, dependency/secret handover is traceable, and a spare matching environment passes N01/N02 first.
Operating precautions	Confirm All flows in Figure 16-1. Preserve the environment package's actual download name and record name/time/scope in backup_inventory.csv. The package is not assumed to contain every history or certificate.
Symptoms and ordered checks	Unknown after restore: inspect dependencies. Broker failure: reenter separately controlled credentials. ENOENT: check delivered external-file paths. Lost alarm state: inspect context design. Never publicly decrypt credential files for convenience.
Files, acquisition and what to inspect	Retain flows_before.json, the original-name environment package, backup_inventory.csv, nodes_inventory.csv and controlled credential handover reference. An administrator uses actual startup logs to locate userDir/settings/flows, retrieving package.json/package-lock.json only under authorized maintenance.
Continue or roll back	Change production only after spare-environment tests and rollback rehearsal. Stop upgrades if backups are unreadable or credentials unavailable.

Related tutorials: T03 Importing and exporting flows / T20 Palette Manager and dependencies

16.2 Node Dependencies for These Examples

The eight examples use core nodes: inject, debug, function, json, mqtt in/out, mqtt-broker, http in, http response, http request, and catch. No third-party Dashboard, PLC, serial, database, or Modbus node is required. A delivered device may still omit components; follow its package list and ask the supplier about missing nodes rather than automatically enabling Internet installation.

Compatibility Item	Record	Maintenance Action
Runtime	Node-RED, Node.js, architecture, and free storage	Establish a supported combination and retain a rollback package
Nodes	Package, exact version, types, and dependencies	Latest is not a locked industrial-project dependency
Business configuration	Config nodes, topics, certificates, URLs, files, and permissions	Read back each setting after migration
State and data	Context store, queues, history, and power-loss behavior	Default memory context is lost on restart; file context is not a transactional database
Download source	Allowed access, maintenance window, restoration policy	Do not leave unapproved downloads permanently enabled

S49 | Check compatibility before adding nodes or trend charts

Check	Explanation and Action
Expected behavior	nodes_inventory.csv records Node-RED, Node.js, architecture and exact package versions; the eight core examples need no third-party Dashboard or Modbus nodes.
Operating precautions	T22/T23 are optional Dashboard references. Verify @flowfuse/node-red-dashboard rather than mixing it with legacy node-red-dashboard; test Base -> Page -> Group -> widget in a matching spare environment.
Symptoms and ordered checks	Blank chart: inspect numeric input, series and timestamp unit in Debug, then Page/Group. Missing nodes/engines errors: stop and inspect dependencies instead of unplanned production upgrades. A browser chart is not persistent history.
Files, acquisition and what to inspect	Obtain package versions from the actual installed-node list or administrator and record nodes_inventory.csv. Retain sanitized S49_chart_input.json and complete S49_install.txt errors. Ask the supplier for inaccessible Node.js/architecture details.
Continue or roll back	Deploy extensions only after supplier approval and spare-environment tests. Otherwise retain the learning links without claiming additional delivered capability.

Related tutorials: T20 Palette Manager and dependencies / T22 Getting started with FlowFuse Dashboard / T23 Temperature trends with ui-chart

16.3 Upgrades, Disconnection, and Restart

Action 1. Keep package upgrade separate from flow import. Prepare all backup types, confirm package/environment compatibility, space, power, and a maintenance window, and use an approved supplier package.

S50 | Verify rollback materials before upgrading the environment

Check	Explanation and Action
Expected behavior	The approved package matches delivery, all three backup types are complete, and power/space/maintenance window are adequate.
Operating precautions	Runtime package upgrade, flow import and device firmware upgrade are different operations. Follow the relevant product upgrade guide, not generic upstream install commands on an OEM unit.
Symptoms and ordered checks	Cannot open after upgrade: preserve Web state/startup errors, check package compatibility/space, then use approved rollback. Do not upgrade every node or immediately reset. Follow delivered timing/progress guidance rather than invented wait times.
Files, acquisition and what to inspect	Review package_release_notes.pdf, backup_inventory.csv and approved package checksum. Save actual action time, Web feedback and administrator-provided startup errors in S50_upgrade.txt; keep actual runtime/firmware inventory controlled.
Continue or roll back	Close the window only after runtime, N01, MQTT and actual reception pass; otherwise use the approved backup rollback.

Action 2. Assess CPU, memory, message rate, Debug volume, and storage writes in production. Do not change every poll to milliseconds merely for perceived real-time behavior.

S51 | Inspect message volume before rebooting a slow system

Check	Explanation and Action
Expected behavior	Polling/publication remain stable with resource headroom and no continuous flood of complete customer messages in Debug.
Operating precautions	Observe the approved load without stress-testing production. Disable/remove temporary Debug nodes and deploy when finished.
Symptoms and ordered checks	Slow editor: separate browser Debug flooding from device-wide resource pressure, then inspect feedback, duplicate tabs, millisecond timers and third-party queues. Have an administrator identify growing files rather than deleting unknown system files.
Files, acquisition and what to inspect	Save S51_resources.png from the actual resource page if available, otherwise request administrator samples. Use S51_rate.csv for input/output counts and periods in the same window; inspect repeat/topic/duplicates in flows_after.json.
Continue or roll back	Pass after removing abnormal message sources, restoring approved periods and verifying stable reception with before/after evidence.

Action 3. MQTT reconnection does not prove offline backfill. Verify queue capacity, dropping, persistence, ordering, and replay separately. This guide provides no persistent offline queue.

S52 | Verify new data after reconnection without assuming backfill

Check	Explanation and Action
Expected behavior	Nodes reconnect and new actual samples reach the platform; any backfill of missing intervals requires a separate design and proof.
Operating precautions	This guide has no persistent queue, power-loss retention or historical backfill. Interrupt only an approved data path while maintaining power and safe control.
Symptoms and ordered checks	Connected but updates stopped: trace latest source sample -> input topic -> JSON/Function -> output topic -> platform query window. Do not use Retain to pretend history was backfilled; retained messages are not new measurements.
Files, acquisition and what to inspect	Record outage, reconnect, first source/platform sample and gaps in S52_outage.csv. Retain S52_after.txt from independent subscription and S52_platform.png with update time. Backfill additionally needs queue/replay evidence.
Continue or roll back	Accept fresh-data recovery separately from backfill and explicitly record unverified gaps.

Related tutorials: T11 Handling errors, Catch and Status

Action 4. After restart, verify service, node connections, actual output, and state initialization. N04 initializes again, N05 waits for a valid sample, and N07 returns 503 with an empty cache.

S53 | Verify per-case initialization after restart

Check	Explanation and Action
Expected behavior	After runtime restart N04 emits initial on its first sample, N05 awaits valid data and N07 returns empty-cache 503; new input restores each case according to its rules.
Operating precautions	Restart only in an approved window after backup/notification. Connection recovery does not establish recovery of every scenario.
Symptoms and ordered checks	Healthy service but no business: check dependencies/credentials, then trace a read-only sample. Persistent N07 503 follows S39-S40 tab/cache checks; N05 needs an initial valid sample. Use approved environment rollback if recovery fails.
Files, acquisition and what to inspect	Obtain startup time/errors in S53_startup.txt. Record actual N01/N02/N04/N05/N07 outcomes, message times and owner in S53_retest.csv; do not prefill acceptance.
Continue or roll back	Handover only after runtime, core flows, actual platform and security retests. Record remaining issues, owner and rollback state.

Related tutorials: T12 Context scope and storage

17 Evidence Files, Handover and Restoration Records

This chapter covers evidence ownership and handover only. First checks and recovery conditions are already beside S01-S53; there is no separate final fault table.

17.1 Where Files Come From and Who Supplies Them

File or Record	Acquisition and Owner	Check Before Handover
flows_before.json, flows_after.json	Customer downloads Menu > Export > All flows and renames the PC copy	Valid JSON, tabs/node scope and changes; not proof of deployment
Original-name environment package	Customer downloads from Export on the device NODE-RED page	Nonempty, original name, environment and scope; verify secret/certificate/history coverage against handover
nodes_inventory.csv	Customer records visible installed-node values; administrator/supplier supplies inaccessible Node.js/architecture details	Exact package/version, source, runtime and approver; authorized package.json/lock file can corroborate
backup_inventory.csv	Customer records flows, environment, dependencies, external files and controlled secret references	Actual download time/name/version, custodian and spare-environment restore result; no plaintext secrets
Sxx_input/output.json, Debug captures	Customer copies/captures the relevant temporary Debug on the PC	Current time, node, topic, identity and number/string type; keep necessary structure after sanitizing

File or Record	Acquisition and Owner	Check Before Handover
Sxx_watch/publish/receiver.txt	Customer copies PC terminal output	Full command, start/end time, exact error and matching messageId; not an internal device log
Sxx_startup/runtime.txt	Administrator obtains actual startup/runtime logs or existing device exports	Relevant time, restart, error and real configuration path; ask supplier if no entry exists
Broker connection/authorization records	Broker administrator supplies a specified time window	Client ID, authentication, disconnect and publish/subscribe authorization; no password/full production log required
Platform reception/parser/storage records	Platform administrator supplies device/request/time-specific records	Reception, parsing, business code, persistence and update time; online state is not acquisition success
Register maps, terminal/PLC protocols, Codec	Instrument/PLC/terminal owners and project manager supply them	Model/version, address base, type, ordering, scale, unit and read/write rights; record codec version
TLS and credential handover	Customer security administrator provides a controlled handover	CA source, hostname, validity, scope, rotation and revocation; no private keys/tokens in public ZIP

Locate settings.js, flow files, package.json and package-lock.json only under authorized maintenance in the actual runtime. Confirm paths from real startup fields such as Settings file, User directory and Flows file. OEM layouts can differ; do not copy PC or other-model directories. Credential storage files are not public evidence attachments.

17.2 One Reproducible Record for Each Issue

In Evidence_Record.csv record the S checkpoint, model/options, time/timezone, expected/actual result, last working and first failing stages, recent changes, evidence filenames, owner and restoration state. Correlate input, processing, output and platform by message ID or device ID plus time/value. Record clock offsets before correlating; captures from different periods are not automatically one event.

At minimum hand over a sanitized relevant flow, one before/after input/output pair, node/page, exact error, dependency list, actual point map, reproduction steps and rollback version. Mark unavailable administrator files as pending; do not substitute teaching samples for field evidence.

17.3 Restore and Retain Materials

Disable/remove lab tabs and deploy, verifying lab timers stop. Restore approved broker, URL, topics and download policy; revoke temporary network permissions/test credentials and stop PC services. Recheck the original business with read-only actual data and record restoration time/remaining issues. Keep production credentials, backups and logs under customer access/retention controls, separate from the public tutorial package.

18 Delivery Checklist, Files, and References

18.1 Customer-Verifiable Completion Conditions

Check	Pass Evidence	Record
Environment and permission	Actual model, entry, dependencies, and authorization verified	Device/version/owner, no plaintext password
First run	One N01 injection gives 25.5; string rejected	Debug message and test time
Data publication	Independent N02 subscriber sees 25.5; invalid input has no valid output	Source/output topics and samples
Alarms and recovery	N03 three-state test; N04 hysteresis/dedup sequence	Do not claim notification delivery or control acceptance
Freshness	N05 timeout/recovery and N07 fresh 200/expired 503	Timing and HTTP status
HTTP POST	N08 matching messageId at receiver; disconnection is not success	Response and receiver record
Real-data integration	Actual point list, paths, scale, and quality verified	A simulated publish is not real-device acquisition acceptance
Safety and restoration	Disable/remove lab tabs and deploy; revoke temporary access and verify backups	Restoration time and remaining issues

18.2 Eight Importable Flows

File	Chapter	Input and Output
01_first_run.json	Chapter 5	Manual Inject to Debug; no external connection
02_mqtt_temperature.json	Chapter 7, E1	raw to processed, 255 to 25.5
03_combined_alarm.json	Chapter 8, E2	state to alarm for every valid input
04_hysteresis_events.json	Chapter 9	hysteresis/in to hysteresis/event, raise 30 / clear 28
05_timeout_recovery.json	Chapter 10	heartbeat to connectivity for one source
06_decoded_lorawan.json	Chapter 12	Simulated decoded uplink to normalized JSON
07_readonly_http_api.json	Chapter 13	api/input updates the cache; GET /guide/latest
08_http_post_response.json	Chapter 14	Manual POST to the lab receiver; no automatic retry

The package README provides complete single-line commands. PDF lines may wrap visually; do not insert spaces inside double-hyphen options or URLs. Enter MQTT credentials in config nodes after import; the JSON does not include them.

Support/English also contains Project_Parameters.csv and Customer_Test_Record.csv. Actual-result cells are blank for field use; the forms do not claim project acceptance. Never enter real passwords, tokens, or private keys in them.

18.3 Clickable Public Tutorials by Customer Task

These 23 resources link directly to Node-RED, its official Cookbook or the relevant package maintainer. Reviewed on 2026-11-05. All original resources are in English. T02 contains the official video entry; video availability depends on the reader's network. Translation does not change the delivered interface or its field names.

Choose by task: T01-T06 for a first run, T07-T10 for MQTT processing, T11-T14 for alarms/timeouts and T16-T19 for HTTP. T15 and T22-T23 are optional extensions, not delivered capability. Online addresses, credentials, topics and versions may differ from this guide; never overwrite project parameters blindly.

Getting started and editor

Tutorial and Public URL	Customer Use and Limits	Chapters
T01 Creating your first flow https://nodered.org/docs/tutorials/first-flow	General reference / English Learn Inject, Debug, Function and deployment. Keep this guide's 255 scaling exercise rather than replacing it with the tutorial timestamp.	Chapter 4 / Chapter 5
T02 Editor guide and official Essentials video entry https://nodered.org/docs/user-guide/editor/	General reference / English Find workspace/sidebar guidance and the English video series embedded on the page. Newer layouts may differ from the 3.0.2 captures.	Chapter 3 / Chapter 4
T03 Importing and exporting flows https://nodered.org/docs/user-guide/editor/workspace/import-export	General reference / English Review export scope, new-tab imports and JSON. Keep environment packages separate from flow exports.	Chapter 4 / Chapter 16

Messages, JSON and processing

Tutorial and Public URL	Customer Use and Limits	Chapters
T04 Messages, property paths and types https://nodered.org/docs/user-guide/messages	General reference / English Investigate string/object confusion and incorrect temperature paths. Verify values, identity and time separately.	Chapter 5 / Chapter 7 / Chapter 11 / Chapter 12
T05 Debug sidebar and complete messages https://nodered.org/docs/user-guide/editor/sidebar/debug	General reference / English Copy sanitized messages at checkpoints and identify their source node and time.	Chapter 4 / Chapter 14
T06 Writing Functions https://nodered.org/docs/user-guide/writing-functions	General reference / English Understand validation, return msg and HTTP request context. Review code before any connection to control systems.	Chapter 5 / Chapter 7 / Chapter 14

Tutorial and Public URL	Customer Use and Limits	Chapters
T10 Convert to/from JSON https://cookbook.nodered.org/basic/convert-json	General reference / English Normalize inputs to objects before reading fields and avoid double serialization on output.	Chapter 7 / Chapter 11 / Chapter 12 / Chapter 14

MQTT acquisition and publication

Tutorial and Public URL	Customer Use and Limits	Chapters
T07 Connect to an MQTT broker https://cookbook.nodered.org/mqtt/connect-to-broker	General reference / English Understand the client configuration node. The tutorial localhost is not the PC broker address used from a gateway.	Chapter 6
T08 Subscribe to a topic https://cookbook.nodered.org/mqtt/subscribe-to-topic	General reference / English Understand topic levels and wildcards. Start with the exact input topic and exclude the flow's own output.	Chapter 6 / Chapter 12
T09 Publish messages to a topic https://cookbook.nodered.org/mqtt/publish-to-topic	General reference / English Check MQTT Output and receiver topics. A connection is not proof of delivery to the intended application.	Chapter 6 / Chapter 12

Alarms, timeouts and state

Tutorial and Public URL	Customer Use and Limits	Chapters
T11 Handling errors, Catch and Status https://nodered.org/docs/user-guide/handling-errors	General reference / English Separate catchable errors from connection state. A warning does not automatically generate a Catch event.	Chapter 5 / Chapter 14
T12 Context scope and storage https://nodered.org/docs/user-guide/context	General reference / English Understand node, flow and global scope, and whether state survives a restart.	Chapter 9 / Chapter 10 / Chapter 13 / Chapter 16
T13 Drop messages that have not changed value https://cookbook.nodered.org/basic/report-by-exception	General reference / English Explore report-by-exception. Simple value filtering does not replace N04's 30/28 hysteresis state machine.	Chapter 8 / Chapter 9
T14 Trigger when a message is not received https://cookbook.nodered.org/basic/trigger-timeout	General reference / English Study timeout patterns while retaining this guide's valid-sample, first-sample and per-device checks.	Chapter 10

Optional Modbus reference

Tutorial and Public URL	Customer Use and Limits	Chapters
T15 Community Modbus node and maintainer examples https://flows.nodered.org/node/node-red-contrib-modbus	Optional extension / English Optional reference only. At review, the page requires Node.js >=22 and Node-RED >=4; this does not fit 3.0.2 directly. Start with native acquisition to MQTT.	Chapter 11 / Chapter 16

HTTP queries and requests

Tutorial and Public URL	Customer Use and Limits	Chapters
T16 Create an HTTP endpoint https://cookbook.nodered.org/http/create-an-http-endpoint	General reference / English For a host calling Node-RED; every branch needs an HTTP Response. This is not an outbound platform POST tutorial.	Chapter 13
T17 HTTP Request client basics https://cookbook.nodered.org/http/simple-get-request	General reference / English The source uses GET to teach the client node. N08 still requires its specified POST method, body and address.	Chapter 14
T18 Get a parsed JSON response https://cookbook.nodered.org/http/parse-json-response	General reference / English Investigate string-versus-object responses. Evaluate business ok separately from the HTTP status.	Chapter 14
T19 Set a request header https://cookbook.nodered.org/http/set-request-header	General reference / English Review Content-Type and platform authentication headers. Keep real tokens out of public examples.	Chapter 14

Dependencies, security and maintenance

Tutorial and Public URL	Customer Use and Limits	Chapters
T20 Palette Manager and dependencies https://nodered.org/docs/user-guide/editor/palette/manager	General reference / English Record installed packages and versions first. Ask the administrator if management access is unavailable.	Chapter 4 / Chapter 16
T21 Securing the editor and HTTP endpoints https://nodered.org/docs/user-guide/runtime/securing-node-red	General reference / English Separate editor authentication from business endpoint protection. Follow the supplier's delivered OEM configuration method.	Chapter 3 / Chapter 13 / Chapter 16

Optional visualization

Tutorial and Public URL	Customer Use and Limits	Chapters
T22 Getting started with FlowFuse Dashboard https://dashboard.flowfuse.com/getting-started.html	Optional extension / English Optional visualization: verify @flowfuse/node-red-dashboard compatibility before studying Base, Page, Group and widgets. Not a standard-delivery promise.	Chapter 16
T23 Temperature trends with ui-chart https://dashboard.flowfuse.com/nodes/widgets/ui-chart.html	Optional extension / English Optional: check numeric type, series and timestamp units. A displayed chart is not a historical database.	Chapter 16

Tutorials support learning and diagnosis; they supply no production credentials or universal compatibility promise. Native device references and isolated-editor captures are retained; no new real PLC, RF, actuator or production-platform acceptance was performed.