ToddySoft Connect IEC 60870-5-104 Driver - Ignition 8.1

Ignition 8.1 module for IEC 60870-5-104 stations: RTUs, substation gateways and telecontrol units. The module is the **controlling station** (the master):

- it receives the station's measured values, single and double points as the station reports them;
- it sends commands;
- it watches the link and reports a station failure.

IEC 104 is built on spontaneous reporting. So every value comes **from the station's reports**, never from polling: a read in IEC 104 would be a general interrogation of the whole station.

> **What this module does not do.** TLS (IEC 60870-5-7 / IEC 62351) is not supported. IEC 101 over serial lines is not offered.

Installation

> **This module is for Ignition 8.1 only.** It is built against the Ignition 8.1 SDK and will not run on any other major version. For Ignition 8.3, install the separate `…-8.3.modl` build instead — the two are not interchangeable, and each gateway needs the build that matches it.
>
> **Ignition 8.0 and earlier are not supported and cannot be.** The module is built on the device-driver API that Ignition introduced with 8.1, so on 8.0 the gateway fails to load it with `NoClassDefFoundError: … opcua.server.api.AbstractDeviceModuleHook`.
>
> **Ignition 8.1.33 or later is required.** The module runs on Java 17, and Ignition 8.1 ships Java 17 only since 8.1.33. Every earlier 8.1 release runs Java 11, which cannot load the module (`UnsupportedClassVersionError`, "class file version 61.0"), so Ignition refuses to install it there. On an older gateway, upgrade Ignition within 8.1 first. The gateway names its Java version in its log at every start (`Java version: 17...`).
>
> Both builds carry the same name in the gateway's module list, so tell them apart by the file you installed or by the module description, which names the Ignition version.
>
> **Ignition's OPC UA module must be installed and running.** This is an OPC UA device driver: it registers with Ignition's own OPC UA server, and every tag it exposes is served through it. With that module absent the gateway cannot load this one, and the log fills with `ClassNotFoundException` / `NoClassDefFoundError` naming `com.inductiveautomation.opcua…` classes. The OPC UA module ships with Ignition and is installed by default, so in practice this bites only on a gateway where it has been removed or disabled. **Ignition 8.1 cannot refuse the module for you** — its descriptor has no way to mark a dependency as required, a mechanism that arrived in 8.3 — but it does name the cause plainly, one line before the stack trace. In `logs/wrapper.log`, look for:
>
> `Module "com.toddysoft.connect.…" depends on "com.inductiveautomation.opcua", but it doesn't exist.`
>
> It then attempts to load the module anyway, which produces the class errors above. Ignition's own drivers behave identically in this situation, so this is how the platform works and not a fault in this module.

The ToddySoft Connect modules are signed with a self-signed certificate (issued to *ToddySoft GmbH*) rather than a publicly accredited code-signing certificate. Ignition therefore asks you to review and accept that certificate once during installation - it is then remembered for all future ToddySoft modules signed with the same certificate.

1. Open the Ignition Gateway web interface and navigate to **Config > System > Modules**. Scroll to the bottom and click **Install or Upgrade a Module...**.
2. Choose the `ToddySoft-Connect-IEC-60870-5-104-Driver-8.1.modl` file and click **Install**.
3. Ignition shows the self-signed certificate details (subject and issuer *ToddySoft GmbH*) together with a warning that the module was not written by Inductive Automation. Review the certificate, tick **I understand the risk and want to install this module** and click **Add Certificate and Install Module**.
4. The module is now trusted and listed under the **ToddySoft GmbH (self-signed certificate)** group with the state **Running**.



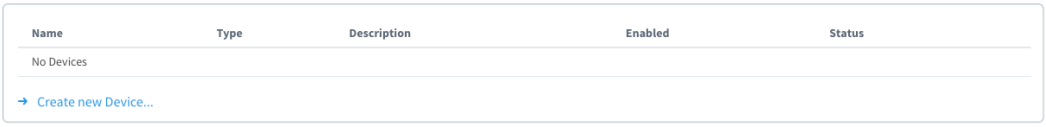
> **No gateway restart is needed on Ignition 8.1.** A module installed or upgraded through **Install or Upgrade a Module...** runs at once. Connected Vision clients and Designers have to restart; Perspective sessions do not. A `.modl` file copied into the gateway's `user-lib/modules` folder instead is picked up only at the next gateway start - use the web interface.

Before you start: the station side

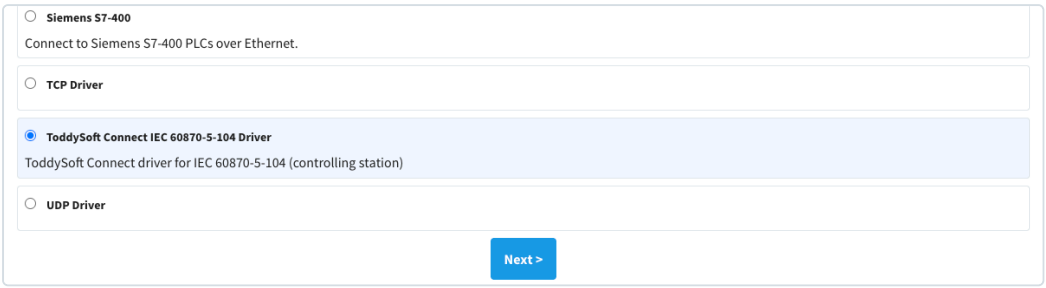
- **The station must accept the gateway as a master.** Most stations keep an allowed-master list (the IP addresses that may connect); add the gateway's address, and check how many masters may be connected at once.
- **The field sizes must match the station's.** IEC 104 does not negotiate the size of the common address (CASDU), the information object address (IOA) or the originator address. The station's configuration lists them; see Station Settings. A mismatch shows as frames the driver cannot parse.
- **Know the station's common address (CASDU)** and, for commands, the **addresses of its command points**. A station never announces its command addresses; see Command Points.
- **Know the station's time zone**, if its values carry time tags: IEC 104 time tags carry no offset.

Creating a Device Connection

1. Go to **Config > OPC UA > Device Connections**.

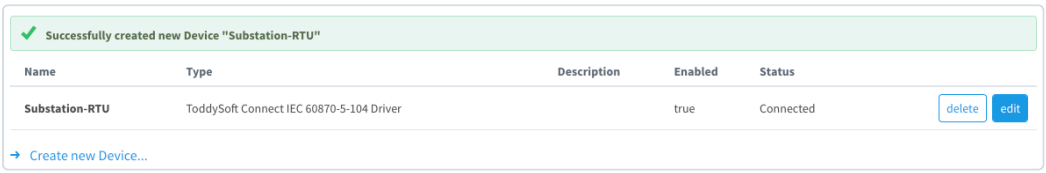


2. Click **Create new Device...** and select **ToddySoft Connect IEC 60870-5-104 Driver** as the device type.



3. Enter a **Name** and at least the station's **Host**, and check the field sizes and the **Interrogation CASDU** under Station Settings. Click **Create New Device**.

4. The new device appears in the Device Connections list. Once connected it shows **Connected**, and the station's points appear below the device in Ignition's tag browser.



Finding devices on the network

The Ignition 8.1 module has no device discovery; enter the station's address by hand. (The Ignition 8.3 module can probe address ranges for stations.)

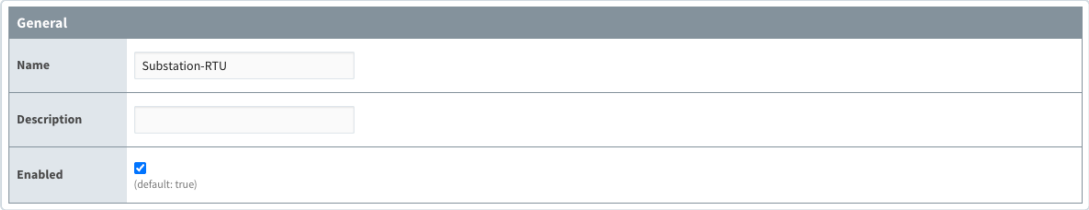
Configuration

The configuration form has these sections, top to bottom:

- **General**
- **Transport Settings**
- **Station Settings**
- **Command Settings**
- **Advanced Settings**
- **Audit Log Settings**
- **Triggered Groups**

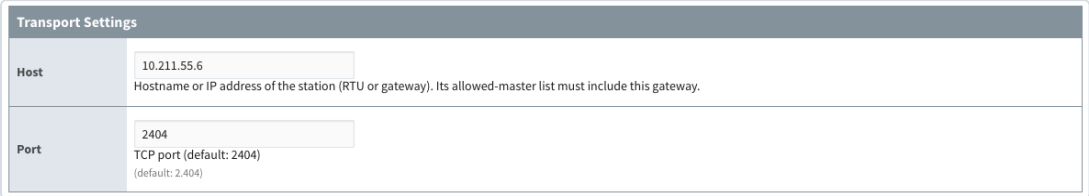
General

Assigns a name (and optional description) to the connection and controls whether it is enabled.



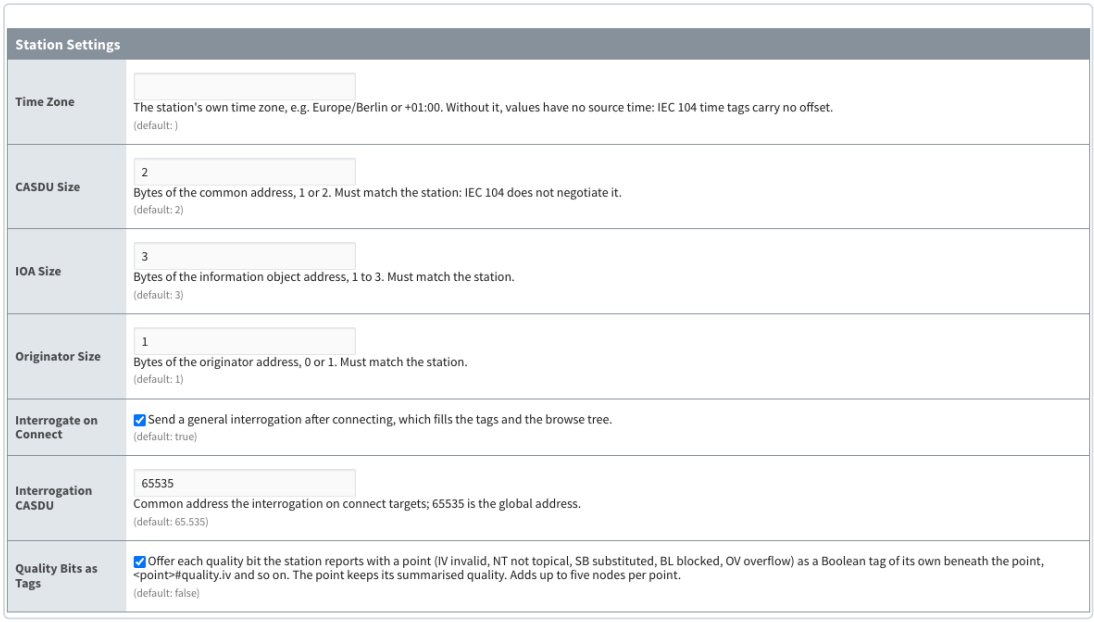
Setting	Description	Default
Name	Unique name for this device connection	
Description	Optional description for this connection	
Enabled	Whether the connection is active	`true`

Transport Settings



Setting	Description	Default
Host	Hostname or IP address of the station. Required	`localhost`
Port	TCP port	`2404`

Station Settings

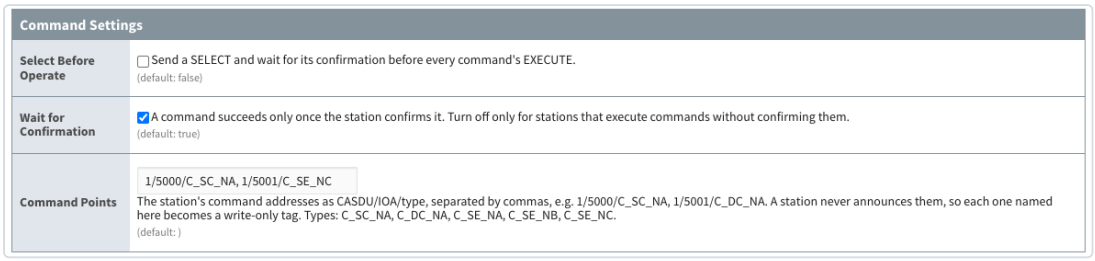


Setting	Description	Default
Time Zone	The station's own time zone, e.g. `Europe/Berlin` or `+01:00`. Without it, values have no source time	(empty)
CASDU Size	Bytes of the common address, 1 or 2. Must match the station	`2`
IOA Size	Bytes of the information object address, 1 to 3. Must match the station	`3`
Originator Size	Bytes of the originator address, 0 or 1. Must match the station	`1`
Interrogate on Connect	Send a general interrogation after connecting, which fills the tags and the browse tree	`true`
Interrogation CASDU	Common address the interrogation targets; `65535` is the global address	`65535`
Quality Bits as Tags	Offer each quality bit of a point as a Boolean tag of its own beneath it (see Quality bits as tags)	`false`

Interrogate on Connect is what fills the tag tree: IEC 104 has no directory, so the module lists the points the station has sent, and the interrogation makes the station send all of them. Leave it on. Most stations answer an interrogation to the global address `65535`; a station that answers only its own common address needs that address here.

Time Zone: IEC 104 time tags (the `..._TB` types) carry the station's local time without an offset. With the station's zone set, a reported value carries the station's own time stamp; without it, the time of receipt.

Command Settings



Setting	Description	Default
Select Before Operate	Send a SELECT and wait for its confirmation before every command's EXECUTE	`false`
Wait for Confirmation	A command succeeds only once the station confirms it. Turn off only for stations that execute commands without confirming them	`true`
Command Points	The station's command addresses as `CASDU/IOA/type`, separated by commas, e.g. `1/5000/C_SC_NA, 1/5001/C_DC_NA`	(empty)

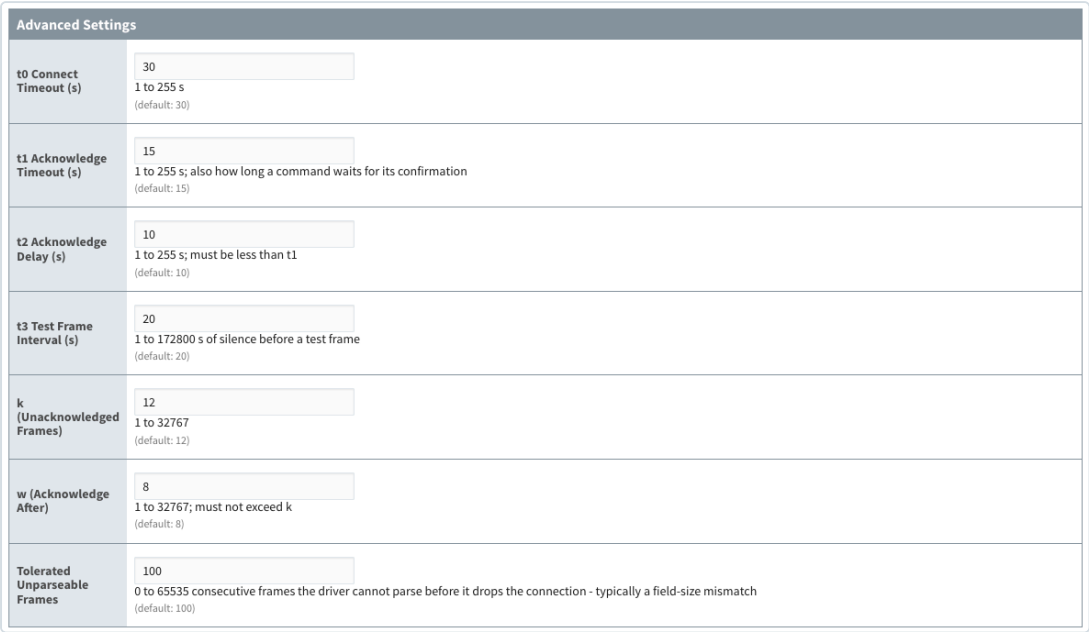
Command Points are needed for every command. A station never announces its command addresses, and the module lists only what the station has sent, so a command point exists in Ignition only when it is named here. Each entry becomes a tag in its station's folder, typed by its command type:

Command type	Command	Value
`C_SC_NA`	single command	Boolean
`C_DC_NA`	double command	SByte (1 off, 2 on)
`C_SE_NA`	set point, normalised	Int16
`C_SE_NB`	set point, scaled	Int16
`C_SE_NC`	set point, short float	Float

An entry that is not a command point is refused when the device is saved, with the entry named.

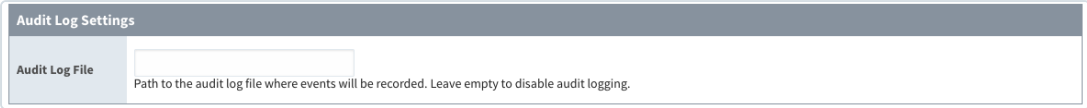
Advanced Settings

The link-layer timers and windows of IEC 104. The defaults are the standard's; change them only to match a station that is configured otherwise.



Setting	Description	Default
t0 Connect Timeout (s)	1 to 255 s	`30`
t1 Acknowledge Timeout (s)	1 to 255 s; also how long a command waits for its confirmation	`15`
t2 Acknowledge Delay (s)	1 to 255 s; must be less than t1	`10`
t3 Test Frame Interval (s)	1 to 172800 s of silence before a test frame	`20`
k (Unacknowledged Frames)	1 to 32767	`12`
w (Acknowledge After)	1 to 32767; must not exceed k	`8`
Tolerated Unparseable Frames	0 to 65535 consecutive frames the driver cannot parse before it drops the connection - typically a field-size mismatch	`100`

Audit Log Settings



Setting	Description	Default
Audit Log File	Path to the audit log file where events are recorded; empty disables	

Triggered Groups

Groups of tags read together when a trigger tag changes as a condition describes, as a JSON list. Empty for none. The same as for the other ToddySoft modules.

Tags

The browse tree

The tree has one folder per station (common address), with everything of that station inside:

```
CASDU 1
  1/100/M_ME_NB          scaled measured value
  1/104/M_SP             single point
  1/300/M_ST             step position, a folder
    1/300/M_ST.step
    1/300/M_ST.transient
  1/600/M_IT             integrated totals, a folder
    1/600/M_IT.counter
    ...
  ...
  $control:1/test        test command
  $control:1/clock-sync  clock synchronisation
  $control:1/counter-interrogation  counter interrogation
  $control:1/general-interrogation  general interrogation
  $status:1/station-failure
  1/5000/C_SC_NA         a command point named under Command Points
```

IEC 104 has no directory, so the tree lists **what the station has sent**: after the interrogation on connect, every point the station reports, plus the station's control and status points and the command points you named. A point the station leaves out of its interrogation answer appears about a second after the station first sends it.

A point is listed **once, by its type family** (`M_SP`, `M_DP`, `M_ST`, `M_IT`, `M_ME_NB`, ...), whichever variant the station sends it in. Stations often answer the interrogation with a point's untimed variant and report each change with a time tag: a SICAM A8000 sends a single point as `M_SP_NA` in the interrogation and as `M_SP_TB` when it changes. Both update the one tag `…/M_SP`. Command points keep their exact type (`C_SC_NA`).

The tags take the type of the point:

IEC 104 type	Point	Ignition type
`M_SP_..`	single point	Boolean
`M_DP_..`	double point	SByte (0 intermediate, 1 off, 2 on, 3 faulty)
`M_ME_NA`, `M_ME_NB`	measured value, normalised / scaled	Int16
`M_ME_NC`	measured value, short float	Float
`M_ST_..`	step position	a folder with two read-only tags: `.step`, SByte (-64 to 63), and `.transient`, Boolean (`true` while the equipment is changing position, e.g. a tap changer moving)
`M_IT_..`	integrated totals	a folder with five read-only tags: `.counter`, Int64 (the station's signed 32-bit reading), `.sequence-number`, SByte (0 to 31), and the Booleans `.carry` (the counter overflowed), `.adjusted` (it was corrected) and `.invalid`

The fields of a step position or of integrated totals carry the point's quality and time stamp; an invalid counter (`.invalid` true) is Bad.

A point the driver cannot type yet (32-bit bitstrings `M_BO_..`, protection-equipment events, packed types) is left out and named once in the gateway log ("Skipping browsed tag ..."); the rest of the tree is not affected.

Addresses

A tag's address is `CASDU/IOA/type`:

```
1/104/M_SP
1/100/M_ME_NB
1/5000/C_SC_NA
```

The fields of a step position or of integrated totals add the field's name: `1/300/M_ST.step`, `1/600/M_IT.counter`.

Two address spaces belong to the module itself:

- `$status:<casdu>/station-failure`: Boolean, read-only, `true` while the connection to the station is lost;
- `$control:<casdu>/test` and `$control:<casdu>/clock-sync`: send a test command or a clock synchronisation to the station.
- `$control:<casdu>/counter-interrogation`: String, sends a counter interrogation (`C_CI_NA`), which asks the station to report, freeze or reset its integrated totals. The value is `<group>/<mode>`:
  - group: `general` (every counter) or `group-1` to `group-4`;
  - mode: `read` (report the current totals; the default, so `general` alone means `general/read`), `freeze` (freeze them without resetting), `freeze-reset` (freeze them and reset the counters) or `reset`.

  For example `general/read` or `group-1/freeze`. The write succeeds once the station has finished its answer; it fails at once when the station refuses the interrogation, and after **t1** when the station does not finish. Any other value is refused before anything is sent (`Bad_TypeMismatch`), and the gateway log names the valid values. Totals are usually frozen on a fixed schedule, for example every 15 minutes: write the tag from a gateway timer script for that. Most stations send their totals only in answer to a counter interrogation, so their `M_IT_..` tags appear after the first one.
- `$control:<casdu>/general-interrogation`: String, sends a general interrogation (`C_IC_NA`): the station sends every value again, changed or not. The value is `general` (or `true`) for the whole station, or `group-1` to `group-16` for one interrogation group. CASDU `65535` asks every station on the connection, for example a gateway and the substations behind it. The write succeeds once the station has finished its answer, so every value has arrived by then; it fails at once when the station refuses the interrogation, and after **t1** when it does not finish. Any other value is refused before anything is sent, and the gateway log names the valid values. An interrogation waits while another one (or a counter interrogation) is still running.

  The tag is a text field in the Designer, because it takes more than yes or no. For operators, put a button in a view whose script writes the value, for example:

```
system.tag.writeBlocking(["
        [default]Station/GeneralInterrogation"], ["general"])
```

Values from reports

Every value comes from the station's reports, as soon as the station sends it:

- **at connect**, the general interrogation delivers every point's current value;
- **after that, every change** the station reports reaches its tag: two changes between two seconds both arrive;
- **the time stamp** is the station's own for the time-tagged types (`..._TB` , with **Time Zone** set), otherwise the time of receipt;
- **the quality** is the station's, mapped in this order (the first that applies wins):

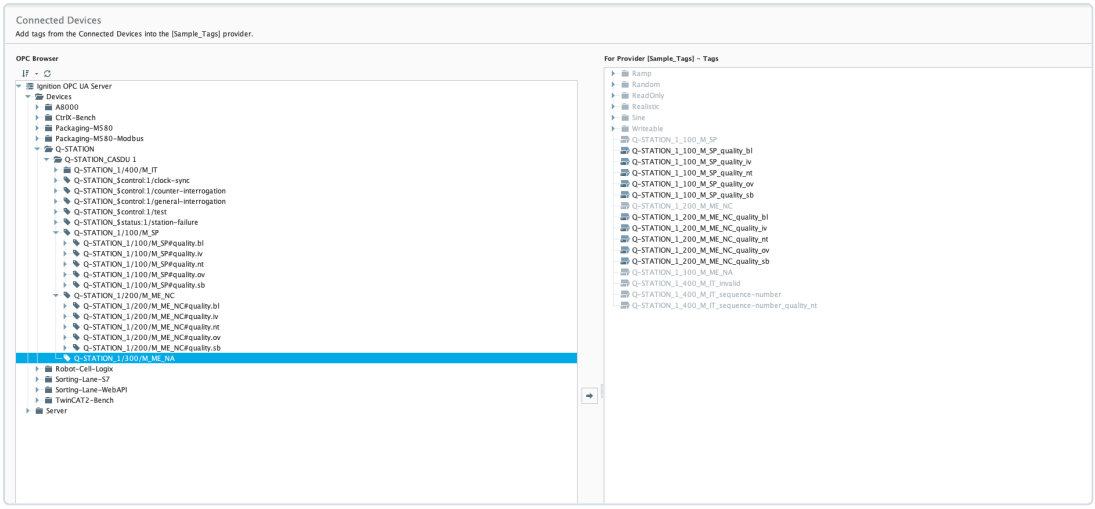
Station quality	Ignition quality
invalid (`IV`)	Bad
overflow (`OV`)	Bad (out of range)
not topical (`NT`)	Uncertain (last usable value)
blocked (`BL`)	Uncertain
substituted (`SB`)	Good (local override)
none	Good

Quality bits as tags

The table shows one quality per tag, the most serious first, so a combination - invalid **and** not topical, substituted **and** blocked - is never visible on the point itself. With **Quality Bits as Tags** on (Station settings, off by default), every point the station reports with quality bits also gets one Boolean tag per bit beneath it in the browse:

Tag	Bit
`<point>#quality.iv`	invalid
`<point>#quality.nt`	not topical
`<point>#quality.sb`	substituted
`<point>#quality.bl`	blocked
`<point>#quality.ov`	overflow

For example `28170/70712/M_SP#quality.sb` . Each is **true** while the station sets the bit, with Good quality and the point's own time stamp, updated together with the point - so each bit can be shown, alarmed and historised on its own. The point's tag keeps its quality as in the table above.





- The bit tags appear in the browse once the station has reported the point, at the latest after the interrogation on connect. A point without quality bits (a normalised value without quality descriptor, `M_ME_ND`) has none. A single point and every field of an integrated total get all five; a bit their type cannot carry (overflow on a single point, all but invalid on an integrated total) stays false.
- Dragged into a tag provider, a bit becomes a tag named `<point>_quality_sb` (a tag name allows neither `#` nor `.`); its OPC item path keeps `#quality.sb`.
- A bit tag on its own is enough: the module subscribes its point for it.
- When the connection is lost, the bit tags keep their last values and turn **Bad** with the point, time-stamped with the loss; `#quality.iv` turns true, because the driver marks every point invalid then. After the reconnect they show the station's bits again.
- They are read-only; nothing about them is ever sent to the station.

The module never interrogates the station again by itself while the connection stands. The station is interrogated only when the connection is made, and again when it is made anew after a loss, or when you write `$control:<casdu>/general-interrogation`; a counter interrogation is sent only when you write `$control:<casdu>/counter-interrogation`.

Losing the station

When the connection to the station is lost:

- `$status:<casdu>/station-failure` turns **true**, with Good quality;
- every other tag turns **Bad** and keeps its last value;
- the device reconnects on its own, with increasing pauses while the station stays away.

When the station is back, the device reconnects, the station is interrogated once, every tag is Good again with current values, and `station-failure` turns **false**.

Command Settings

Write the value to the command point's tag (see Command Points):

- with **Select Before Operate** on, the driver sends SELECT, waits for its confirmation, then EXECUTE;
- with **Wait for Confirmation** on (the default), the write succeeds once the station confirms the command (positive activation confirmation);
- **a refused command fails the write at once**, with the station's reason in the gateway log, for example `the station refused the command 1/4999/SINGLE_COMMAND: unknown information object address`. Ignition shows it as `Bad_NotFound` (unknown address), `Bad_NotSupported` (unknown type) or `Bad_DeviceFailure` (other refusals);
- a command the station does not confirm within **t1** fails; it may still have been executed.

A command point's tag (and a `$control:` tag) is Good: empty until the first command, then the last value the station accepted. It is never read from the station. It shows the last command sent, not what the station did: read that from the station's feedback point.

Tag Groups

IEC 104 values come from the station's reports, so the tag group's rate does not decide how often a value changes in the device: each change arrives as the station sends it, and the station is never polled.

Activating the licence

Without a licence the module runs in Ignition's trial: two hours at a time, then its devices stop until someone clicks **Reset Trial** in the gateway's banner, as often as needed. A licence for this module comes from ToddySoft by email as an **Ignition licence key** with an **activation token**. The ToddySoft licence for the drivers travels inside that key: activating it on the gateway is all there is to do. There is no licence file to copy and no licence setting in the device form. Keep the key and the token private: together they activate the licence.

Online (the gateway can reach the internet)

1. Install the module first (see Installation).
2. In the gateway's web interface open **Config > System > Licensing**, click **Activate License**, enter the **License Key** and click **Submit**.
3. Click **Activate**. If the gateway asks for an activation token, paste the token from the email: one line, without the `BEGIN` and `END` lines.
4. The licensing page shows the key as activated, and **Config > System > Modules** shows this module as **Activated**.

The gateway asks Inductive Automation's activation service (`api.inductiveautomation.com`, HTTPS) for this; a firewall must allow that one outgoing connection.

Offline (the gateway cannot reach the internet)

Offline activation is a round trip with two files, between the gateway and any computer with internet access. It works for six-character licence keys.

1. On the gateway open **Config > System > Licensing**, click **Activate License**, enter the **License Key** and click **Submit**.
2. Click **Offline Activation**, then **Download Activation Request**. If the gateway asks for an activation token, paste the token from the email. The gateway saves the file `activation_message.txt`.
3. On a computer with internet access open https://links.inductiveautomation.com/activation, choose `activation_message.txt` and save the `license.ipl` file it returns.
4. Back on the gateway's **Offline Activation** page, upload `license.ipl` and click **Activate**.

After activating

No restart is needed. The module takes the licence as soon as Ignition reports the activation, and devices that were refused for want of a licence connect again on their next attempt. The licence status shows **VALID** with the tier and the end of

maintenance (see Licence diagnosis); anything else is explained there.

Renewing or extending a licence

- **When ToddySoft has updated the licence on your existing key** (more drivers, longer maintenance), the gateway has to fetch it again. Online: in **Config > System > Licensing** click **Reactivate** next to the key. Offline: unactivate the key first (below), then activate it offline again. Inductive Automation warns that an offline activation without unactivating first can put the gateway into emergency activation mode.
- **When you received a new key**, activate it and unactivate the old one. Keep exactly **one** activated key covering this module per gateway: with two, Ignition hands the module only one of them (see Troubleshooting).

Moving the licence to another gateway

Unactivate the key on the old gateway, then activate it on the new one. The email says how many gateways a key may be activated on at the same time.

- Online: in **Config > System > Licensing** click the **Unactivate License** icon next to the key and confirm with **Yes, Unactivate**.
- Offline: in **Config > System > Licensing** under **Applied Licenses** click **Unactivate** (the bin icon) and confirm with **Yes, Unactivate**. The gateway saves `unactivation_message.txt`; upload it on https://links.inductiveautomation.com/activation from a computer with internet access.

Ignition Edge: a licence for a third-party module on Ignition Edge is arranged through Inductive Automation. Contact ToddySoft before you order one.

Licence diagnosis

When a device reports a licence problem, the module explains it on the gateway itself. Open **Status > Other > ToddySoft Licence (IEC 60870-5-104)**. A device with a licence problem says so in its status and points there.

The diagnosis shows:

- **which gateway** this is, and its role if it is part of a redundant pair;
- **Ignition's licence** for this module on this gateway: activated, trial running (with its end) or trial expired;
- **what ToddySoft licence data arrived** with Ignition's licence: how many parts, whether they are complete, and a checksum that identifies the licence without revealing it;
- the **driver's own verdict** and the **cause**, in words.

Cause	Meaning	What to do
`IGNITION_UNLICENSED`	Ignition applies no activated licence to this module on this gateway (trial running or expired). ToddySoft licence data only arrives with an activated licence.	Activate a licence key that covers this module on this gateway. In a redundant pair every node needs its own activation.
`NO_DATA_FOR_DRIVER`	The activated key carries no ToddySoft licence for this driver.	Send the report to ToddySoft.
`DATA_INCOMPLETE`	The licence data arrived incomplete or damaged.	Send the report to ToddySoft; the licence has to be delivered again.
`DATA_REJECTED`	The driver refused the delivered licence; the reason is shown.	Send the report to ToddySoft.
`RUNNING_ON_BUNDLED_TRIAL`	A licence was delivered but not accepted, and the driver runs on its built-in trial until that ends.	Send the report to ToddySoft.
`NOT_CHECKED`	Licence data arrived, but the driver gave no verdict - for instance because it could not be loaded.	Send the report to ToddySoft.
`VALID`	Licensed; tier and expiry are shown.	Nothing.
`NOT_YET_KNOWN`	The first seconds after start-up, before Ignition reports a licence state.	Wait a moment.

Sending a report to support. The diagnosis comes as a plain-text report, the one thing to send with any licence question. The page shows it ready to select and copy. The same text is served at `http://<gateway>:<port>/data/com.toddysoft.connect.iec60870/licence-report` to a logged-in user. It contains no licence data, no keys, no credentials and no customer details. In a redundant pair, send the report from both nodes.

The gateway log carries the same report once each time the licence situation changes.

Troubleshooting

Licensing problems after a new or renewed licence key was activated — `DATA_INCOMPLETE` with "duplicate fragment index" in the licence diagnosis:

- **Two licence keys covering this module are activated on the gateway at the same time**, typically a renewed or upgraded key activated alongside the old one. Ignition hands the module the licence data of both, in numbered parts that carry the same numbers; the module cannot tell which parts belong together, keeps one of each pair, and the licence it reassembles is unreadable. Depending on the parts it keeps, devices are refused or run on the module's bundled trial until that ends.
- Keep exactly **one** activated licence key that covers this module per gateway: deactivate the old one under **Platform > Licensing**, then restart the module (or the gateway). In a redundant pair, check every node.

The module will not load, or the log shows `ClassNotFoundException` / `NoClassDefFoundError`:

- Ignition's **OPC UA** module is missing or disabled. This driver is an OPC UA device driver and cannot load without it. Install or re-enable it under **Config > System > Modules** and restart the gateway; see Installation.

The device does not connect:

- Check that the station is reachable on port 2404 and lists the gateway as an allowed master.
- Review the gateway log: **Status > Diagnostics > Logs**.

Connected, but the tag tree is empty:

- The station has not answered the interrogation. Check **Interrogate on Connect** is on, and **Interrogation CASDU**: a station that answers only its own common address refuses the global address `65535`; enter its CASDU.
- The gateway log names an interrogation the station refused.

Frames cannot be parsed, or the connection drops repeatedly:

- The field sizes do not match the station's: **CASDU Size**, **IOA Size**, **Originator Size**. IEC 104 does not negotiate them; take them from the station's configuration.

A command point is missing from the tree:

- Command points must be named under **Command Points**; a station never announces them.

A command fails:

- The station refused it; the gateway log carries the reason (unknown address, unknown type, ...). Check the entry under **Command Points** against the station's configuration.
- With **Select Before Operate** on, the station must support select-before-operate for that point.
- A command not confirmed within **t1** fails; a station that never confirms commands needs **Wait for Confirmation** off.

Values have no source time, or the wrong one:

- Set **Time Zone** to the station's own zone: IEC 104 time tags carry no offset.

Docker related issues

Connections work normally from a container, as long as the container can route to the station's port 2404 and the station accepts the address the connection arrives from.

VPN related issues

A station reached over a VPN needs nothing special from this driver: the connection is an ordinary outbound TCP session. The station must list the address the connection arrives from as an allowed master.

Version history

1.3.0 — 2026-11-13

Built on ToddySoft Connect 1.3.0.

- **Fixed: t0 Connect Timeout** bounds the TCP connect; before, a fixed default applied.
- **Changed:** the module requires Ignition 8.1.33 or later (Java 17). On older 8.1 releases, which run Java 11, it could be installed but never loaded; Ignition now refuses the install. See Installation.
- **New:** installing or upgrading the module through the web interface needs no gateway restart on Ignition 8.1; Vision clients and Designers restart. See Installation.
- **New:** how to activate the licence key, online and offline, renew it and move it to another gateway. See Activating the licence.
- **New:** this module and its manual. IEC 60870-5-104 controlling station for Ignition 8.1:
  - every value from the station's reports, never polled;
  - the station's points browsed after the interrogation on connect;
  - commands through named command points, with select-before-operate and confirmation;
  - station failure reported, tags Bad while the station is away, automatic recovery;

- licence diagnosis.