ToddySoft Connect IEC 61850 Driver - Ignition 8.1

Ignition 8.1 module for IEC 61850 IEDs: protection relays, bay controllers, meters and gateways in substations and distribution networks. The module is an **MMS client**:

- it reads and writes the IED's data attributes;
- it operates its control objects with their own control model;
- it can take values from the IED's **report control blocks** instead of polling them.

The IED's data model is browsed from the IED itself. **No SCD or ICD file is needed.**

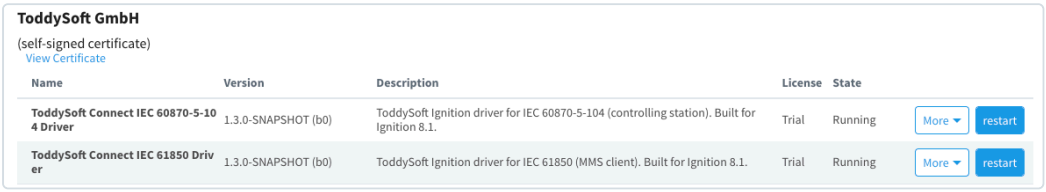
> **What this module does not do.** GOOSE and Sampled Values (the multicast traffic on the process bus) are not offered: this is a client of the IED's MMS server. TLS (IEC 62351) is not supported.

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-61850-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**.



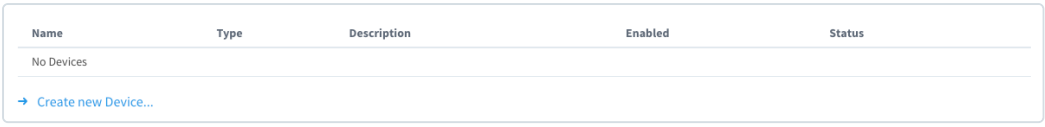
> **No gateway restart is needed on Ignition 8.1.** A module installed or upgraded through **Install or Upgrade a Module...** runs at once. Connected Vision clients and Designers have to restart; Perspective sessions do not. A `.modl` file copied into the gateway's `user-lib/modules` folder instead is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

Before you start: the IED side

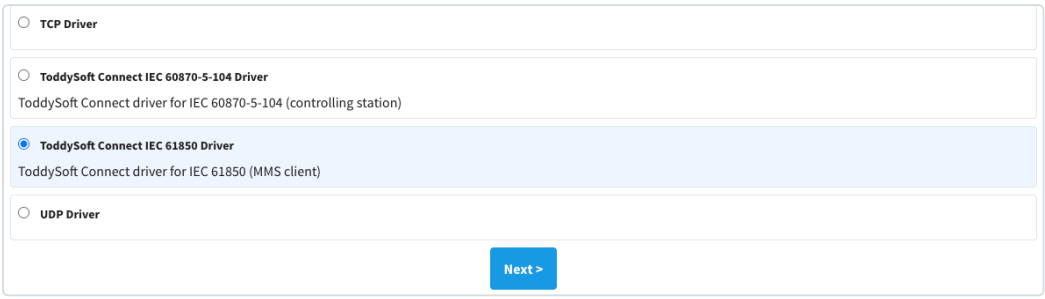
- **The IED must accept an MMS association from the gateway** on TCP port 102. Many IEDs restrict MMS clients by IP address or limit how many may connect at once; check that the gateway is allowed and that a client slot is free.
- **The association settings must match what the IED expects.** Most IEDs accept the defaults this module sends (AP titles, AE qualifiers, selectors, TSAPs). An IED configured otherwise refuses the association; its SCD file, or the IED's own configuration tool, lists the values it wants. See Association.
- **Controls need the IED's permission.** A control object is operated only if the IED allows remote operation (local/remote switch, `Loc` = false) and, where it checks one, the originator.

Creating a Device Connection

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



2. Click **Create new Device...** and select **ToddySoft Connect IEC 61850 Driver** as the device type.



3. Enter a **Name** and at least the IED's **Host**. Every other setting has a working default (see Configuration). Click **Create New Device**.

4. The new device appears in the Device Connections list. Once the association is established it shows **Connected**, and the IED's data model appears below the device in Ignition's tag browser.



Finding devices on the network

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

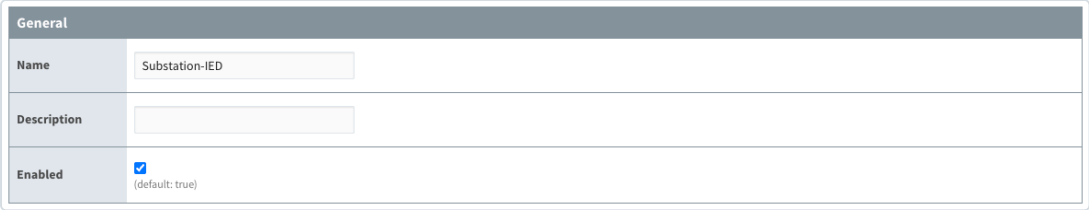
Configuration

The configuration form has these sections, top to bottom:

- **General**
- **Transport Settings**
- **IED Settings**
- **Control Settings**
- **Delivery**
- **Association Settings**
- **Audit Log Settings**
- **Triggered Groups**

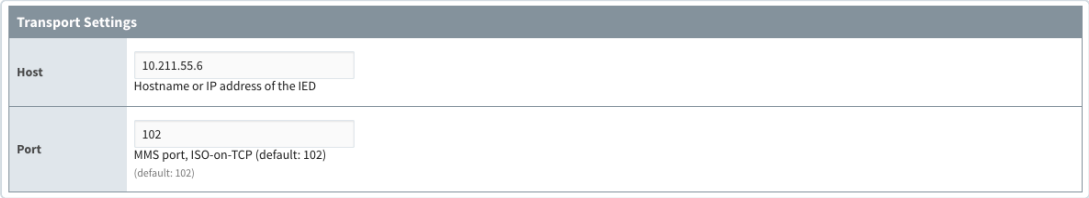
General

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



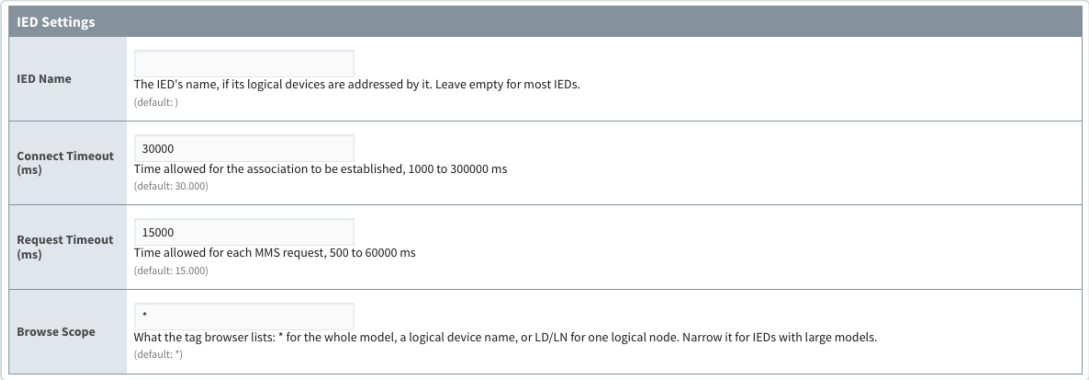
Setting	Description	Default
Name	Unique name for this device connection	
Description	Optional description for this connection	
Enabled	Whether the connection is active	`true`

Transport Settings



Setting	Description	Default
Host	Hostname or IP address of the IED. Required	`localhost`
Port	MMS port (ISO-on-TCP)	`102`

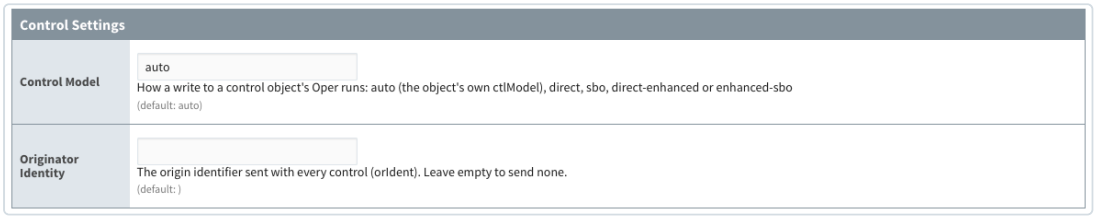
IED Settings



Setting	Description	Default
IED Name	The IED's name, if its logical devices are addressed by it. Leave empty for most IEDs	(empty)
Connect Timeout (ms)	Time allowed for the TCP connect and the association, 1000 to 300000	`30000`
Request Timeout (ms)	Time allowed for each MMS request, 500 to 60000	`15000`
Browse Scope	What the tag browser lists: `*` for the whole model, a logical device name, or `LD/LN` for one logical node	`*`

Browse Scope is worth narrowing on an IED with a large model. The whole model is browsed when the device connects, and on a big relay that can be thousands of attributes in many logical devices.

Control Settings



Setting	Description	Default
Control Model	How a write to a control object runs: `auto`, `direct`, `sbo`, `direct-enhanced` or `enhanced-sbo`	`auto`
Originator Identity	The origin identifier (`orIdent`) sent with every control. Empty sends none	(empty)

With `auto` each control object runs the control model the IED declares for it in its `ctlModel` attribute. One IED commonly has direct-operate outputs next to select-before-operate breakers, and `auto` handles both. Set a fixed model only for an IED that reports its `ctlModel` wrongly. See Operating control objects.

Delivery



Setting	Description	Default
Use Report Control Blocks Where Available	Take the values of attributes covered by one of the IED's report control blocks from its reports instead of polling them. Attributes no free block covers stay polled	`false`

Off, every attribute is **polled** at its tag group's rate. On, the module uses the IED's own report control blocks: see Values from reports.

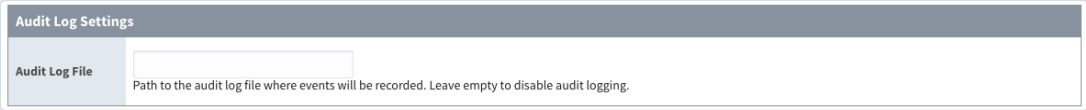
Association Settings

The settings that open the MMS association. **Change them only if the IED's SCD file, or its configuration tool, lists other values.** A mismatch refuses the association, and the device's error names the layer that refused it.



Setting	Description	Default
Local AP Title / **Remote AP Title**	Dotted OIDs; empty omits them	`1.1.1.999` / `1.1.1.999.1`
Local AE Qualifier / **Remote AE Qualifier**	A negative value omits them	`12` / `12`
Local / Remote Session Selector	Hex octets	`0001`
Local / Remote Presentation Selector	Hex octets	`00000001`
Local TSAP / **Remote TSAP**	COTP calling and called TSAP, hex octets	`0001` / `0001`
MMS Max PDU Size	256 to 65535 bytes	`65000`
MMS Max Services Calling / **Called**	1 to 64	`5` / `5`

Audit Log Settings



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

The audit log records protocol-level events and grows quickly, so enable it only while investigating a problem.

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

When the device connects, the module browses the IED's data model and builds the tag tree from it, level by level: **logical device > logical node > data object > data attribute**. For example:

```
SampleIEDDevice1
  MMXU2
    MMXU2.TotW
      MMXU2.TotW.mag.MX
        SampleIEDDevice1/MMXU2.TotW.mag.f.MX
      SampleIEDDevice1/MMXU2.TotW.q.MX
      SampleIEDDevice1/MMXU2.TotW.t.MX
```

Each attribute is a tag of the type the IED declares:

IEC 61850 / MMS type	Ignition type
boolean	Boolean
integer (8 to 64 bit), unsigned	SByte ... Int64, Byte ... UInt64
float32 / float64	Float / Double
bit-string (a quality `q`)	UInt16 and the other unsigned sizes
visible / MMS string	String
UTC time (a time stamp `t`)	DateTime
octet string	ByteString

Access follows the functional constraint (the suffix after the last dot):

- `ST` (status) and `MX` (measurands) are read-only;
- `SP`, `SV`, `CF`, `DC`, `BL`, `SE` are writable;
- `CO` is the control service; see the next section.

A structure the IED declares (a data object, a structured attribute, a report control block) is a folder. An attribute the IED declares as an **array** has one tag per element, at `<attribute>.[0]`, `.[1]`, ...: for example `SampleIEDDevice1/DSCH1.SchdAbsTm.val.SP.[0]`. IEC 61850 has no address for a single element, so the module reads the array once and takes each element from it. Element tags are read-only.

Addresses

A tag's address is the IED's own reference, with the functional constraint at the end:

```
<logical device>/<logical node>.<data object>.<data attribute>.<FC>
SampleIEDDevice1/MMXU2.TotW.mag.f.MX
simpleIOGenericIO/GGIO1.SPCSO1.stVal.ST
```

The functional constraint can be left out where only one fits, but the browse tree always writes it.

Operating control objects

A controllable data object (a switch, a breaker, a binary output) shows its control service as one **tag** per service. The tag is Good and shows the last value written; nothing is read from the IED for it:

- `...Oper.CO` operates the object;
- `...SBOw.CO` selects it with a value (select-before-operate with enhanced security);
- `...Cancel.CO` cancels a selection.

Each takes the control value, `ctlVal`, of the object's type: a Boolean for a switch. **Write the value to the `Oper` tag**, and the driver does the rest: it builds the control request (control number, originator, time), selects first where the control model needs it, and waits for the command termination where the model has one. The members of `Oper` are not tags of their own, because the driver fills them in.

ctlModel	Control model	What a write to `Oper` does
1	direct with normal security	operates
2	select-before-operate with normal security	selects, then operates
3	direct with enhanced security	operates, then waits for the command termination
4	select-before-operate with enhanced security	selects with the value, operates, waits for the command termination

A refused control fails the write, with the driver's reason as the status. If the IED explains a refusal (blocked by interlocking, already in that position, selected by another client), the explanation is in the message. An object with `ctlModel` 0 (status only) cannot be operated.

Values from reports

With **Use Report Control Blocks Where Available** on, the module asks the IED which of its attributes a report control block covers. Each **covered** attribute is then taken from the reports:

- **it updates as soon as it changes**, not at the tag group's rate;
- **with the IED's own time stamp**: the moment the IED measured or changed it, not when the gateway received it;
- **with every change**: two changes between two polls both reach the tag.

An attribute that **no free report control block covers stays polled** at its tag group's rate. The gateway log names it once, for example:

```
No report covers 'simpleIOGenericIO/GGIO1.AnIn1.t.MX' (UNSUPPORTED); it is polled instead
```

A report control block another client already uses is not free. An IED usually has a few instances of each block for that reason. If none is free, the attributes it covers stay polled, and the log says so.

When the connection is lost, every tag goes **Bad**. After the connection returns, the module enables the blocks again and the values arrive with the next report. A **buffered** report control block may also report that entries were lost while no client was connected; that is the IED's own notice and appears in the gateway log.

Tag Groups

By default all tags are in the default tag group and polled once a second. If different intervals are wanted, create further tag groups with those intervals and assign tags to them.

Each polled attribute is read by its own address, never through the structure it belongs to: an IEC 61850 address ends in its functional constraint, so it is no path into a parent's value. Attributes taken from reports do not depend on the tag group's rate.

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 61850)**. 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.

Cause	Meaning	What to do
`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.iec61850/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 IED is reachable on port 102 and allows the gateway as an MMS client.
- **The association is refused**: the association settings do not match the IED's. Compare them with the IED's SCD file or configuration tool; the device error names the layer that refused it. See Association.
- Review the gateway log: **Status > Diagnostics > Logs**.

The tag tree is empty, or parts are missing:

- Check **Browse Scope**: anything other than `*` lists only that logical device or node.
- An attribute the driver cannot type is left out and named in the gateway log ("Skipping browsed tag ..."); the rest of the tree is not affected. Send that line to ToddySoft.

A control write fails:

- The IED refused it: the status and the gateway log carry the reason. Typical causes are the local/remote switch at *local*, an interlocking condition, the object already in the requested position, or another client holding the selection.
- An object with `ctlModel` 0 (status only) cannot be operated.
- With a fixed **Control Model**, check it matches the object's `ctlModel`; `auto` reads it from the IED.

With reports on, an attribute still updates only at the tag group's rate:

- No free report control block covers it, so it is polled. The gateway log names it once (*No report covers ...*). Either another client holds the block, or the IED's data sets do not include the attribute; the data sets are configured in the IED.

A buffered report control block "reports a buffer overflow":

- The block kept entries while no client was connected and its buffer filled. The IED's notice; the values that follow are current.

Docker related issues

Connections work normally from a container, as long as the container can route to the IED's port 102.

VPN related issues

An IED reached over a VPN needs nothing special from this driver: the connection is an ordinary outbound TCP session to port 102. With reports on, values arrive as the IED sends them, so latency affects only when they arrive, not how many.

Version history

1.3.0 — 2026-11-13

Built on ToddySoft Connect 1.3.0.

- **Fixed: Connect Timeout** bounds the TCP connect too, not only the association.
- **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 61850 MMS client for Ignition 8.1:
  - the IED's data model browsed without an SCD file;
  - reads and writes, typed as the IED declares them;
  - control objects operated with their own control model;
  - values from report control blocks where available;
  - arrays as one tag per element;
  - licence diagnosis.
- **Changed:** a browsed array is one folder, named after the array and its index range (for example `arr[0..9]`), holding its elements; until now the elements sat directly beside the array's neighbours. An array of structures shows a folder per element holding its members (for example `pts[2].x`). Each element keeps its OPC item path, so existing tags keep working.