ToddySoft Connect ADS - Apache Kafka Connect

A Kafka Connect **source and sink connector** for Beckhoff TwinCAT controllers, built on the ToddySoft Connect ADS driver. It reads tags from TwinCAT 2 and TwinCAT 3 controllers over ADS/AMS - plain, or as Secure ADS (TLS with a client certificate or a pre-shared key) - and publishes every read as one Kafka record; the sink writes record fields to tags.

The connector runs on **your own Kafka Connect workers** near the plant: Apache Kafka's Connect or Confluent Platform, as a service or in a container. It does **not** run in Confluent Cloud: a managed or custom connector there runs in Confluent's network and cannot reach a controller on a plant network.

Compared with the Apache PLC4X connector

Apache PLC4X, the open-source project ToddySoft Connect's drivers descend from, publishes a free Kafka Connect connector. As of PLC4X 1.0.0 the differences are:

	this connector	**Apache PLC4X connector 1.0.0**
Credentials	handed to the driver as such, supplied by Kafka config providers; never in the connection URL, never logged	only what fits into the connection string
Connector configuration in the worker log	never logged	every property logged when a task starts, connection strings included
When tags are read	on a timer, on a value change (with a condition), or by subscription	at a fixed interval
Configuration	one versioned file, shared with the Camel component, checked before start with every problem located	flat connector properties
Several identical PLCs	one batch over a tag set for many connections, spread over tasks	one source per PLC

	this connector	Apache PLC4X connector 1.0.0
Packaging	one plugin per driver, as plugin zip and Confluent Hub component archive	one plugin with every PLC4X driver
Licence and support	commercial licence; support from ToddySoft	Apache License 2.0, free; community support

Requirements

	supported	tested
Apache Kafka (Connect)	3.9.x, 4.0, 4.1, 4.2, 4.3, 4.4	3.9.2, 4.4.0
Confluent Platform	7.9, 8.0, 8.1, 8.2, 8.3, 8.4	8.4
Java on the worker	17 or newer	21, 25

- **Kafka older than 3.9 and Java older than 17 are not supported.** (On Java 11 the worker cannot load the plugin at all; see Troubleshooting.)
- **Self-managed workers only**, standalone or distributed. Every worker that may run a task needs the plugin, the configuration file (if you use one) and the licence.
- **A route from each worker to the controllers** (TCP port 48898; 8016 for Secure ADS).
- **On the controller**: an **AMS route** for each worker's source AMS Net ID, created beforehand - the connector does not create routes (see The source AMS Net ID and the route). For symbolic addresses the PLC program must be running.
- A **ToddySoft licence for ADS** for production use. Without one the driver runs on its bundled trial licence until that expires.

Installation

You receive the plugin twice, in the share you received your licence and downloads from; both hold the same connector:

- `toddysoft-kafka-connect-ads-<version>.zip` - Confluent Hub's component archive format, for Confluent's installer;
- `toddysoft-kafka-connect-ads-<version>-plugin.zip` - a plain plugin directory.

With Confluent's CLI (Confluent Platform)

```
confluent connect plugin install ./toddysoft-kafka-connect-ads-
        1.3.0.zip \
  --plugin-directory /usr/share/confluent-hub-components \
  --worker-configurations /etc/kafka/connect-distributed.properties
        --force
```

The CLI installs from the local file, without internet access, into `<plugin-directory>/toddysoft-kafka-connect-ads` and adds the directory to `plugin.path` in the worker configuration. `--force` accepts the licence agreement it shows; leave it out to read and accept it yourself. It does not run on Windows, and not while logged in to Confluent Cloud. Older Confluent Platform releases have `confluent-hub install <zip> --component-dir <dir> --worker-configs <file> --no-prompt` instead.

By unzipping (any worker)

```
sudo unzip toddysoft-kafka-connect-ads-1.3.0-plugin.zip -d
        /opt/kafka/plugins/
```

`/opt/kafka/plugins` must be listed in the worker's `plugin.path` (`plugin.path=/opt/kafka/plugins`). Restart every worker afterwards.

In a container

Mount the unzipped plugin and the licence directory into the worker container, for example with the official `apache/kafka` image (it runs as `appuser`, home `/home/appuser`):

```
docker run -d --name connect \
  -v /opt/kafka/plugins:/opt/kafka/plugins:ro \
  -v /etc/toddysoft/licenses:/home/appuser/.toddysoft/licenses:ro \
  -v /etc/kafka-connect:/etc/kafka-connect:ro \
  apache/kafka:4.4.0 /opt/kafka/bin/connect-distributed.sh
        /etc/kafka-connect/worker.properties
```

`confluentinc/cp-kafka-connect` runs as `appuser` with the same home; its default plugin path is `/usr/share/confluent-hub-components`.

A worker in a container must name its **source AMS Net ID** in each connection URL: the default, derived from the container's own address, is not one the controller has a route for (see The source AMS Net ID and the route).

Checking the installation

```
curl -s http://localhost:8083/connector-plugins | grep -i ads
```

lists `com.toddysoft.connect.integrations.kafka.ads.AdsSourceConnector` and `…AdsSinkConnector` with their version.

Updating

Remove the old plugin directory before installing the new one (the Confluent CLI does this itself): with two versions of one plugin on `plugin.path`, the worker loads either. Restart every worker.

Licence

The driver reads its licence from **`.lic` files in `~/.toddysoft/licenses/` of the user the worker runs as**:

the worker runs	**put the licence file in**
as a service, user `kafka`	`~kafka/.toddysoft/licenses/ads.lic` (for example `/home/kafka/.toddysoft/licenses/`)
in the `apache/kafka` or `cp-kafka-connect` container	a host directory mounted at `/home/appuser/.toddysoft/licenses` (see above)
in a distributed cluster	the same on **every** worker: a task moved to a worker without it runs on the trial there

The file name does not matter, only the extension `.lic`; one file per driver or one licence covering several drivers both work. A new licence counts from the next connector or task start (restart the connector). Without a licence, or with one the driver rejects, the bundled trial applies until it expires; the driver then refuses its connections.

Checking the licence

Each task writes a **licence report** to the worker log when it starts, one block per driver, every line beginning `ToddySoft licence [ads]`:

```
ToddySoft licence [ads] running on the bundled trial, ends 2027-01-
02
ToddySoft licence [ads]   running on: the bundled trial
(ClasspathLicenseSource), checksum 31e3005ea51b, valid until 2027-
01-02
ToddySoft licence [ads]   found and rejected: a licence file
(FileSystemLicenseSource), checksum 9be41f07c2aa:
PLATFORM_NOT_LICENSED: License does not entitle this platform
ToddySoft licence [ads]   licence directory:
/home/appuser/.toddysoft/licenses (derived from user.home)
ToddySoft licence [ads]   platform APACHE_KAFKA, worker user appuser
```



Find it with `grep 'ToddySoft licence' <worker log>` (in a container: `docker logs connect 2>&1 | grep 'ToddySoft licence'`). A licence is named by its **checksum**, the first 12 characters of its SHA-256: `shasum -a 256 ads.lic` shows which file is meant.

the report says	meaning	what to do
`licensed` · running on: a licence file	your licence applies	nothing
`licensed, expires in N days`	it ends soon	renew in time
`licensed in its grace period: expired on <date>, refused after <date>`	it has ended; ToddySoft Connect still accepts it for 30 days	renew now: after that date the licence is refused
`attention: Support ends in N days (<date>)` / `attention: Support ended on <date>`	driver versions built after that date will not run on this licence	renew the support before updating the plugin
`running on the bundled trial, ends <date>` and no *found and rejected* line	no licence file found	put the file in the directory the report names, for the user it names
… and `found and rejected: …: PLATFORM_NOT_LICENSED`	the licence does not cover Kafka Connect	ask ToddySoft for a licence that does
… `DRIVER_NOT_LICENSED`	it is a licence for another driver	use the ADS licence
… `LICENSE_EXPIRED`	it has expired	renew

the report says	meaning	what to do
`NOT licensed: …`	no licence applies, not even the trial; the driver refuses every connection	as above

A licence refusal on a connection is reported once with the same block, then counted (`refused N connections since …, licence unchanged`, at most every ten minutes) instead of filling the log. Nothing in the log shows the licence itself or whom it is licensed to. The task does not fail on a licence: the connector stays RUNNING and its records carry the refusal in `errors`; the report is where the cause is.

Configuration file

Both connectors run a ToddySoft Connect **event-pump configuration** (version 2): the connections with their credentials, the tags by alias, and batches - which tags are read together, and when. The archive's `etc/` holds a sample; a complete one:

```
{
  "version": 2,
  "connections": [
    {
      "id": "line1",
      "url": "ads:tcp://192.168.10.20?source-ams-net-
        id=192.168.10.5.1.1&target-ams-port=851",
      "tags": {
        "temperature": "MAIN.fTemperature",
        "counter":     "MAIN.nCounter",
        "running":     "GVL.bRunning",
        "setpoint":    { "address": "MAIN.fSetpoint", "access":
        "write" }
      }
    }
  ],
  "batches": [
    { "id": "process", "connectionId": "line1", "tags":
        ["temperature", "counter"],
      "trigger": { "type": "timer", "intervalMillis": 1000 } },
    { "id": "state", "connectionId": "line1", "tags": ["running",
        "counter"],
      "trigger": { "type": "change", "watch": ["running"],
        "pollIntervalMillis": 200 } }
  ]
}
```

- **Connection URL**: `ads:tcp://<host>` (port 48898) for plain ADS, `ads:tls://<host>` or `ads:tls-psk://<host>` (port 8016) for Secure ADS. Parameters:
  - `target-ams-port` - the PLC runtime: `851` for TwinCAT 3 (the default), `801` for TwinCAT 2;
  - `target-ams-net-id` - the controller's AMS Net ID, default `<its IP address>.1.1`; set it when the controller's Net ID does not follow that convention;
  - `source-ams-net-id` - this client's AMS Net ID, as the controller's route names it (see below).
- **Addresses**, symbolic, as TwinCAT shows them: `MAIN.fTemperature`, `GVL.sStatus`, a structure member `MAIN.stMotor.fSpeed`, an array element `MAIN.aValues[0]`, `MAIN.stArray[1].nField` or `MAIN.matrix[2][3]`, a whole array `MAIN.aValues` (read as one list) or a whole structure `MAIN.stMotor`. TwinCAT 2 reports its symbol names in upper case (`MAIN.BSTART`). By index group and offset: `0x4020/0:DINT`, `0x4020/0[0..9]:DINT` (ten DINTs), `0x4020/2.3:BIT`, `0x4020/0:STRING(80)`, `0x4020/0:WSTRING(40)`.
- **Triggers**: `timer` (every `intervalMillis`), `change` (read the batch when a watched tag changes, optionally under a `condition` such as `oldValue == false && value == true`), `subscription` (the driver subscribes with ADS device notifications and forwards each change).
- **Access**: `read` tags are never written, `write` tags are never read; without `access` a tag is both.
- **Several identical PLCs**: see below.

Give the file to the connector either inline (`configuration`) or as a path on **every** worker (`configuration.file`). A configuration with a problem refuses the connector, listing each problem with its location in the file and a rule id.

The source AMS Net ID and the route

A TwinCAT controller answers only a client it has an **AMS route** for, and it recognises the client by its **source AMS Net ID**. Create the route on the controller before the connector connects - in TwinCAT XAE / System Manager (*Routes → Add*), or in `StaticRoutes.xml` - for the Net ID the connection announces and the address of the worker machine. The connector does not register routes. **TwinCAT 2 needs up to about two minutes for a new route** while it restarts its AMS router; TwinCAT 3 applies a route at once.

Without `source-ams-net-id` the driver announces `<the worker's own address>.1.1`. Set it explicitly when that is not the Net ID the route was made for:

- **The worker runs in a container.** The container's own address is not the one the controller sees; the controller sees the machine the container runs on. Announce the Net ID the route was made for, usually that machine's address followed by

`.1.1`. Measured with an `apache/kafka:4.4.0` worker in Docker against a TwinCAT 3 (CX9240):

```
ads:tcp://<plc>?source-ams-net-id=<the address the route was made for>.1.1&target-ams-port=851
```

The default, derived from the container's address, is a Net ID with no route, and the controller does not answer it.

- **The worker machine has several addresses**, for example two in the controller's subnet: the default may be derived from another one than the route names. Set `source-ams-net-id`, and on a worker running directly on the machine bind the connection to the address the route was made for with `tcp.local-address` (`tls.local-address` / `tls-psk.local-address` for Secure ADS).
- **The worker runs on the TwinCAT machine itself.** Both defaults would be the PLC's own Net ID: pick another one, for example `<address>.1.2`, and create a route for it pointing at the machine's own address.

TwinCAT keeps one connection per source AMS Net ID. A second client with the same Net ID - another worker, an Ignition gateway, another ToddySoft product on the same machine - takes the connection away from the first one, and both keep displacing each other (`Connection closed by remote` within milliseconds of connecting). Give each worker, and each other client, its own Net ID and route. Within one plugin, connectors using the same connection share one connection (see Several driver plugins on one worker); two connections with different URLs to the same controller and the same Net ID still compete.

Secure ADS

Secure ADS (TwinCAT 3.1 Build 4024 or newer) is TLS on port 8016, with one of two credentials (see Credentials):

- `ads:tls-psk://` - a **pre-shared key** and identity, as configured in the controller's route. No certificates. TwinCAT derives the key as SHA-256 of the identity (in upper case) followed by the password: `echo -n "KAFKA1MySecret" | shasum -a 256` gives the 64 hex characters the connector needs.
- `ads:tls://` - this client's **certificate**, from a PKCS12 keystore. The controller needs a route for this certificate. If the keystore file does not exist yet, the first connect creates it (a self-signed certificate, owner-readable, protected with the credential's keystore password) and is refused with `UnknownCert`, naming the file, its common name and fingerprint: create the route for it, then restart the connector. Keep the file on a persistent, writable path: a new key needs a new route.

`machine-name` is the name announced in the Secure ADS session, by default the host name without its domain (in a container: the container's host name). With a self-signed client certificate it must equal the certificate's common name, or the controller

answers `CnMismatch`.

The controller's own certificate is verified (`tls.verify`, on by default). A certificate signed by your own CA needs a trust store on every worker (`tls.trust-store=/etc/kafka-connect/plc-ca.p12&tls.trust-store-password=…`). A self-signed controller certificate can be pinned by its SHA-256 fingerprint, which the worker log names when the certificate is refused:

```
ads:tls://192.168.10.20?source-ams-net-
id=192.168.10.5.1.1&tls.pinned-fingerprints=95:62:7b:…:d1
```

`tls.verify=false` accepts any certificate and gives no protection against interception.

Credentials

A credential never goes into the URL or the file itself. Every secret field is a **reference** `{{name}}`, and the connector property `secrets.<name>` supplies the value - best through a Kafka **config provider**, so the value stays out of the connector configuration too. Enable the file provider on every worker:

```
config.providers=file
config.providers.file.class=org.apache.kafka.common.config.provider.Fi
```

and refer to a properties file readable only by the worker's user:

```
"secrets.line1.psk": "${file:/etc/kafka-
        connect/secrets/plc.properties:line1.psk}"
```

A connection over Secure ADS with a pre-shared key:

```
{
  "id": "line1",
  "url": "ads:tls-psk://192.168.10.20?source-ams-net-
        id=192.168.10.5.1.1&target-ams-port=851",
  "credential": { "method": "pre-shared-key", "pskIdentity":
        "KAFKA1", "pskKey": "{{line1.psk}}" },
  "tags": { "temperature": "MAIN.fTemperature" }
}
```

the connection	`method`	fill in
plain ADS (`ads:tcp://`)	no `credential`	nothing

the connection	`method`	fill in
Secure ADS with a pre-shared key (`ads:tls-psk://`)	`pre-shared-key`	`pskIdentity`, `pskKey` (hexadecimal)
Secure ADS with a client certificate (`ads:tls://`)	`certificate`	`keystore` (a path on every worker), `keystorePassword`

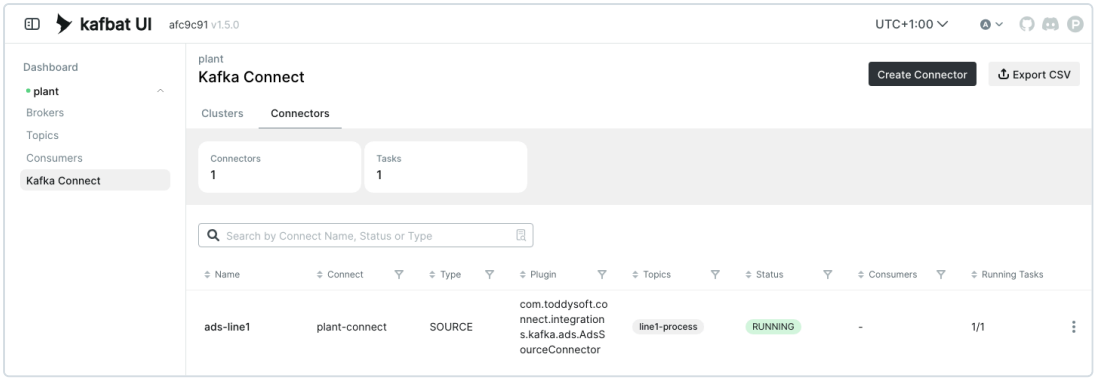
Kafka never logs these connectors' configuration values, and the worker writes the `${file:…}` placeholder, not the value, into its config topic. Restrict the config topic with ACLs all the same.

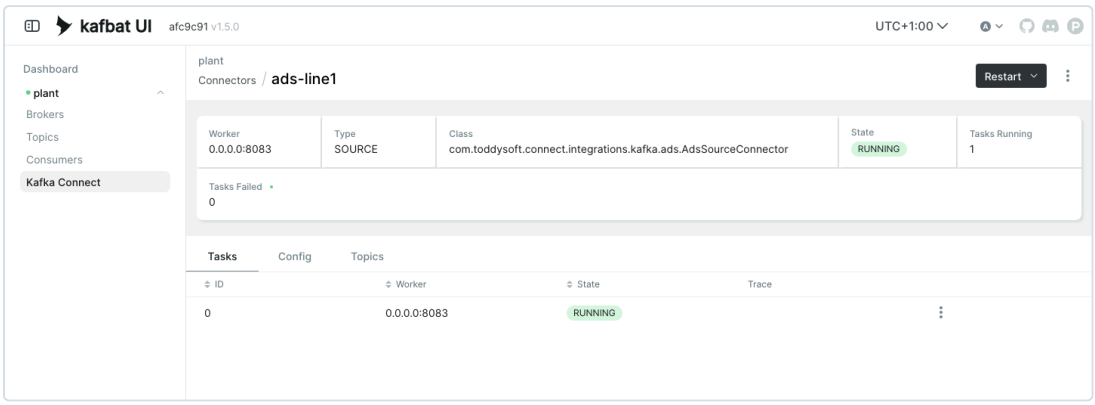
Source connector

```
{
  "name": "ads-line1",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.ads.AdsSourceConnector",
    "configuration.file": "/etc/kafka-connect/line1.toddysoft.json",
    "batches.topics": "{\"process\": \"line1-process\", \"state\":
        \"line1-state\"}",
    "topic.default": "line1-other",
    "tasks.max": "1"
  }
}
```

`curl -X POST -H 'Content-Type: application/json' --data @ads-line1.json http://localhost:8083/connectors` creates it. A connection with a credential needs its `secrets.<name>` here as well (see Credentials).

property	default	meaning
`configuration` / `configuration.file`	-	the configuration, inline or as a path on every worker; exactly one
`batches`	all	the batch ids to run, comma-separated
`batches.topics`	`{}`	JSON object: batch id to topic
`topic.default`	-	topic of every batch not in `batches.topics`
`secrets.<name>`	-	the value of `{{name}}`
`tasks.max`	1	batches (and connections, see below) are spread over at most this many tasks





The screenshots show kafka-ui (kafbat UI), a free web interface for Kafka and Kafka Connect; any tool that shows connectors and topics does, Confluent Control Center on Confluent Platform among them. They were taken against a TwinCAT 3 (CX9240) reading the tags `MAIN.g_b1`, `MAIN.g_s16` and `MAIN.g_s32` once a second.

Records

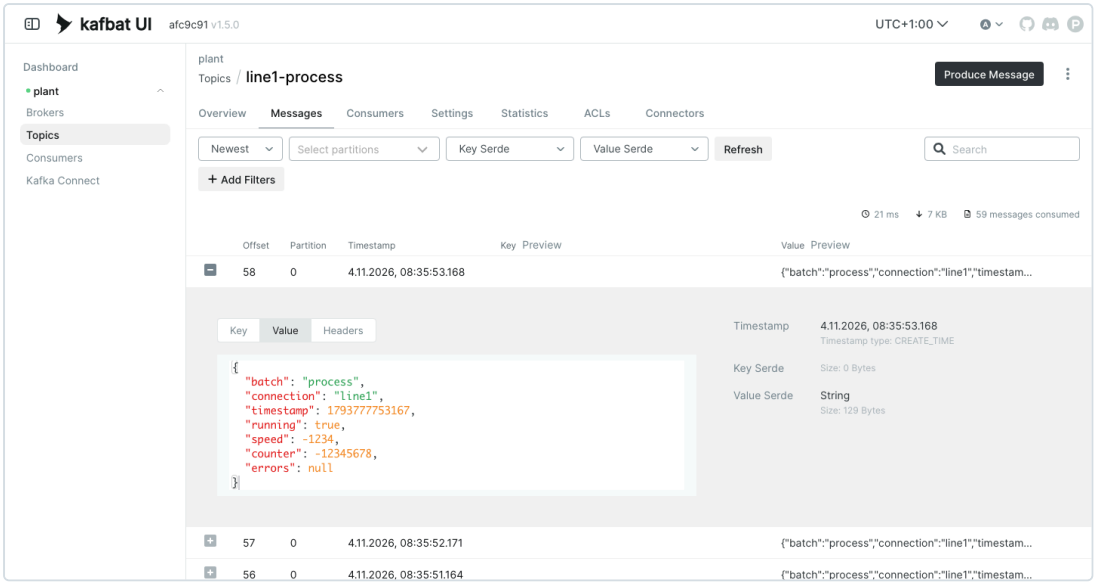
One record per read. Its value is a `Struct`:

field	type	
`batch`	STRING	the batch id
`connection`	STRING	the connection id
`timestamp`	INT64	the time of the read, milliseconds since the epoch
one field per tag alias	see the type table	`null` when the tag could not be read
`errors`	MAP<STRING,STRING>	alias to the reason, for every tag that could not be read

The record has **no key**; its source partition is `{batch, connection}`, its offset `{timestamp}`.

controller type	Connect type	
BOOL, BIT	BOOLEAN	BIT: a bit member of a structure
SINT	INT8	
BYTE, USINT, INT	INT16	unsigned 8-bit types widened
WORD, UINT, DINT	INT32	unsigned 16-bit types widened
DWORD, UDINT, LINT	INT64	unsigned 32-bit types widened
LWORD, ULINT	Decimal (scale 0)	unsigned 64-bit, exact
REAL	FLOAT32	
LREAL	FLOAT64	
STRING, WSTRING	STRING	
TIME, LTIME	INT64	milliseconds
TIME_OF_DAY, LTIME_OF_DAY	INT64	milliseconds since midnight
DATE, LDATE	INT32	days since the epoch
DATE_AND_TIME, LDATE_AND_TIME	Timestamp	
a structure or function block	STRING	JSON
an array	ARRAY of STRING	each element as text, a structure as JSON

A field takes its type from the batch's **first read**, and every field is optional: a tag that could not be read on the first read is a STRING from then on - restart the connector once the controller answers.



Converters and data formats

The connector produces Connect `Struct` records with a schema; the worker's converter decides the bytes on the topic:

- `org.apache.kafka.connect.json.JsonConverter` with `value.converter.schemas.enable=true` (schema and payload) or `false` (payload only);
- `io.confluent.connect.avro.AvroConverter` or `io.confluent.connect.protobuf.ProtobufConverter` with Confluent's Schema Registry (`value.converter.schema.registry.url=http://registry:8081`), on a worker that has those converters (Confluent Platform has them).

Set the converter per connector (`"value.converter": "…"`) or for the worker. Every read of a batch has the same schema; a tag added to the batch changes it.

Sink connector

Writes the fields of each record to the tags of one connection, by alias:

```
{
  "name": "ads-line1-setpoints",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.ads.AdsSinkConnector",
    "configuration.file": "/etc/kafka-connect/line1.toddysoft.json",
    "connection": "line1",
    "topics": "line1-setpoints"
  }
}
```

property	default	meaning
`connection`	-	the connection id written to
`topics`	-	the topics read
`max.retries`	3	how often a write the controller failed is tried again
`retry.backoff.ms`	1000	the first wait before a retry, doubled each time
`retry.backoff.max.ms`	30000	the longest wait

A record is a `Struct` (with a converter that carries a schema, for example JSON with schemas or Avro); each field names a tag alias of the connection, and its type follows the type table. A field the connection does not define, or a tag whose access is `read`, is refused: the record goes to Kafka Connect's error reporting (a dead letter queue with `errors.tolerance=all` and `errors.deadletterqueue.topic.name`) unwritten. A write the controller fails is retried, then reported the same way. A write refused for a licence reason is not retried.

Several identical PLCs

A batch with `connectionIds` runs once per connection - usually reading a tag set every PLC has:

```
{
  "version": 2,
  "tagSets": { "press": { "speed": "MAIN.fSpeed", "count":
        "MAIN.nCount" } },
  "connections": [
    { "id": "press-1", "url": "ads:tcp://192.168.10.31?source-ams-
        net-id=192.168.10.5.1.1", "tagSets": ["press"] },
    { "id": "press-2", "url": "ads:tcp://192.168.10.32?source-ams-
        net-id=192.168.10.5.1.1", "tagSets": ["press"] },
    { "id": "press-3", "url": "ads:tcp://192.168.10.33?source-ams-
        net-id=192.168.10.5.1.1", "tagSets": ["press"] }
  ],
  "batches": [
    { "id": "line-speed", "connectionIds": ["press-1", "press-2",
        "press-3"], "tags": ["speed", "count"],
      "trigger": { "type": "timer", "intervalMillis": 500 } }
  ]
}
```

Each press's read is a unit of its own, spread over `min(units, tasks.max)` tasks: with `tasks.max` 3 on a three-worker cluster each worker reads one press. The records keep the batch id `line-speed` and name the connection, so one topic holds

every press, partitioned by press. One source AMS Net ID towards several controllers is fine: TwinCAT keeps one connection per Net ID on each controller.

In **distributed mode** Kafka Connect moves tasks between workers on a rebalance; a moved task connects from its new worker and writes its licence report there. A stopped worker's tasks move only after up to five minutes (`scheduled.rebalance.max.delay.ms`): Kafka Connect waits that long for the worker to come back. Each controller therefore needs a route for every worker that may run its task. Standalone mode runs everything on one worker.

Several driver plugins on one worker

Each driver is its own plugin, and they sit side by side on one worker (`/opt/kafka/plugins/toddysoft-kafka-connect-ads`, `…-s7`, …). **Use each driver's own connector classes** - `…kafka.ads.AdsSourceConnector` here: a worker loads a connector class from one plugin only. The generic `PlcSourceConnector` / `PlcSinkConnector` exist in every plugin and are not meant to be used; a task that finds itself without the driver its URL names refuses to start and names the class to use. Within one plugin, a source and a sink to the same controller share one connection - which on ADS also means they do not displace each other.

Troubleshooting

symptom	cause	what to do
the plugin is not in `GET /connector-plugins`	not on `plugin.path`, or the worker was not restarted	check `plugin.path` and the directory, restart every worker
the plugin is not listed, and the worker log shows `Failed to discover SourceConnector in …` caused by `UnsupportedClassVersionError … (class file version 61.0)`	the worker runs on Java 11 or older	run the worker on Java 17 or newer
a task fails with `driver-missing`	the generic or another driver's connector class	use `…kafka.ads.AdsSourceConnector` / `AdsSinkConnector`
the connector is refused with a rule id and a location	the configuration has a problem	fix the file at each location named

symptom	cause	what to do
RUNNING, but every record has the tags `null` and the reason in `errors`	the controller refuses or cannot be reached	read `errors`: an address, a missing route, a certificate, a licence
`errors` mentions `License validation failed`	no licence applies	see Checking the licence
TwinCAT 3: every connect times out; TwinCAT 2: `Transport is closed`	the controller has no route for the source AMS Net ID	check `source-ams-net-id` (always set it in a container) and create the route, see The source AMS Net ID and the route
TwinCAT 2: the route was just created and connecting still fails	the controller restarts its AMS router, for up to about two minutes	wait, and look again after two minutes
`Connection closed by remote` within milliseconds of connecting, connections keep dropping	another client uses the same source AMS Net ID towards this controller	give each worker and each other client its own Net ID and route
a tag `null` with a not-found reason in `errors`	the symbol is not in the running program, or the program is not running	copy the name from TwinCAT (TwinCAT 2: in upper case), put the PLC in RUN
`UnknownCert`	the controller has no route for this client's certificate (a keystore just created, or replaced)	create the route for the certificate, see Secure ADS
`CnMismatch`	the announced machine name differs from the client certificate's common name	set `machine-name` to the common name

symptom	cause	what to do
the TLS handshake refuses the controller's certificate	not signed by a CA the worker trusts	name a trust store, or pin the fingerprint the log names (see Secure ADS)
`ads:tls-psk` connections refused	identity or key differ from the controller's route	compute the key again from the identity in upper case and the password
odd behaviour after an update	two copies of the plugin on `plugin.path`: the worker lists it once and logs nothing, so you cannot tell which copy runs	remove the old directory, restart

Raise the driver's log level for more detail: `curl -X PUT -H 'Content-Type: application/json' --data '{"level":"DEBUG"}' http://localhost:8083/admin/loggers/com.toddysoft` (and back to `INFO` afterwards).

Version history

1.3.0 — 2026-11-13

Built on ToddySoft Connect 1.3.0.

- **New:** the ADS connector as a product for self-managed Kafka Connect workers - this manual, a Confluent Hub component archive next to the plugin zip, both checked by every release; a licence report in the worker log per driver; supported versions Apache Kafka 3.9.x and 4.0-4.4, Confluent Platform 7.9 and 8.0-8.4.