ToddySoft Connect S7 Web API - Apache Kafka Connect

A Kafka Connect **source and sink connector** for Siemens S7-1200 and S7-1500 controllers, built on the ToddySoft Connect S7 Web API driver. It reads tags through the controller's own **SIMATIC Web API** - JSON-RPC over HTTPS, with the user accounts of the controller's user management - and publishes every read as one Kafka record; the sink writes record fields to tags.

The Web API is the controller's management interface, not its process-data path: it is roughly an order of magnitude slower per tag than the S7 protocols and has no change notification, so every read is a poll. It reaches what the S7 protocols cannot - the diagnostic buffer, active alarms, the operating mode - and works where network policy allows only HTTPS. **For cyclic process data use the S7 connector**; use this one for diagnostics, occasional reads and writes, and HTTPS-only networks.

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**
Siemens S7 Web API (HTTPS, user accounts)	yes	no driver
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 443, HTTPS).
- **On the controller**: an S7-1200 from firmware V4.5 or an S7-1500 from V2.9 (S7-300 and S7-400 have no Web API). In the TIA project the **web server** must be switched on and the **Web API** enabled - a separate switch. The connector logs in with an **account of the controller's user management** that holds the Web API right, plus the rights for what it does: reading and writing values are separate rights (see Credentials). Tags are visible only when they are marked accessible from HMI/Web in the TIA project.
- A **ToddySoft licence for S7 Web API** for production use. A licence for the S7 driver does not cover it. 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-s7-webapi-<version>.zip` - Confluent Hub's component archive format, for Confluent's installer;
- `toddysoft-kafka-connect-s7-webapi-<version>-plugin.zip` - a plain plugin directory.

With Confluent's CLI (Confluent Platform)

```
confluent connect plugin install ./toddysoft-kafka-connect-s7-webapi-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-s7-webapi` 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-s7-webapi-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`.

Checking the installation

```
curl -s http://localhost:8083/connector-plugins | grep -i s7webapi
```

lists `com.toddysoft.connect.integrations.kafka.s7webapi.S7WebApiSourceConnector` and `…S7WebApiSinkConnector` 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/s7-webapi.lic` (for example `/home/kafka/.toddysoft/licenses/`)

the worker runs	put the licence file in
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 [s7-webapi]`:

```
ToddySoft licence [s7-webapi] running on the bundled trial, ends 2027-01-02
ToddySoft licence [s7-webapi]   running on: the bundled trial
(ClasspathLicenseSource), checksum 31e3005ea51b, valid until 2027-01-02
ToddySoft licence [s7-webapi]   found and rejected: a licence file
(FileSystemLicenseSource), checksum 9be41f07c2aa: PLATFORM_NOT_LICENSED:
License does not entitle this platform
ToddySoft licence [s7-webapi]   licence directory:
/home/appuser/.toddysoft/licenses (derived from user.home)
ToddySoft licence [s7-webapi]   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 s7-webapi.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

the report says	meaning	what to do
… 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 - an S7 licence among them	use the S7 Web API licence
… `LICENSE_EXPIRED`	it has expired	renew
`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": "s7-webapi://192.168.10.20",
      "credential": { "method": "username-password",
                      "username": "{{line1.username}}", "password": "
        {{line1.password}}" },
      "tags": {
        "temperature": "\"Process\".temperature",
        "counter":     "\"Process\".counter",
        "running":     "\"Process\".running",
        "setpoint":    { "address": "\"Process\".setpoint", "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": 1000 } }
  ]
}
```

- **Connection URL**: `s7-webapi://<host>` (port 443; another port as `s7-webapi://<host>:<port>`). The Web API is HTTPS only; there is no plain-HTTP mode and no transport to choose. The certificate settings are in the next section. `request-timeout-ms` (default 10000) bounds one request; the defaults suit every current S7-1200 and S7-1500 firmware.
- **Addresses** are symbolic only, as TIA Portal shows them: `"DataBlock".member`, `"DataBlock".struct.member`, a PLC tag of the global tag table `"ManualTrigger"`, an array element `"DataBlock".values[2]` (indices of a multi-dimensional array comma-separated in one bracket: `"DataBlock".matrix[1,1]`). A member whose name holds anything but letters, digits and `_` must be quoted: `"OPC_UA_DB"."OPC Data".g_b1`. The controller's declared type applies; a type suffix (`"DataBlock".duration:TIME`) is needed only where the Web API names none. An array or a structure named as a whole is read whole, but never written whole - write its elements or members. An array keeps its declared lower bound: one may start at 0 and the next at 1. There are no absolute `%DB` addresses: addresses are not interchangeable with the S7 connector's. In JSON a `"` inside an address is written `\"`.
- **Diagnostics** are read like tags: `$diag:buffer[0..19]` (the 20 newest diagnostic-buffer entries), `$diag:alarms` (active alarms), `$session:permissions` (the account's rights), `$control:operating-mode:STRING` (the operating mode; see Sink connector before writing it).
- **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 and forwards each change). The Web API has no change notification: the driver serves a subscription by polling, at most every 500 ms.

- **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 controller's certificate

S7-1200 and S7-1500 controllers serve the Web API with a **self-signed certificate** out of the box. The driver verifies it (`tls.verify` , on by default - keep it on) and refuses a certificate it cannot verify. Every record then carries in `errors`

```
Error acquiring lease for connection: Unable to create transport instance:
Server certificate not trusted for 192.168.10.20:443. For a device with a
self-signed certificate, pin it with tls.pinned-fingerprints=<sha-256> — the
connect log prints the peer's fingerprint …
```

and the worker log names the controller's `SHA-256 fingerprint: 20:d6:…:54:a3` . Check that fingerprint against the controller and pin it in the URL:

```
s7-webapi://192.168.10.20?tls.pinned-fingerprints=20:d6:…:54:a3
```

A certificate signed by your own CA needs no pin, provided it names the address in the URL: name a trust store on every worker instead (`tls.trust-store=/etc/kafka-connect/plc-ca.p12&tls.trust-store-password=…`). Leave `tls.version` unset: the driver then prefers TLS 1.3 and falls back to TLS 1.2, which older controllers need.

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.FileConfigPr
```

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

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

the controller	`method`	fill in
user management with a Web API account	`username-password`	`username` , `password`

The Web API authenticates every request, so a connection needs its `credential`; without one the controller refuses the login (`Login failed for user 'null'`). `password` must be a reference; `username` may be one or plain text. In the TIA project the account needs the **Web API** right, which permits only the login, and separately the rights to read and to write values. Read `$session:permissions` to see what the account has.

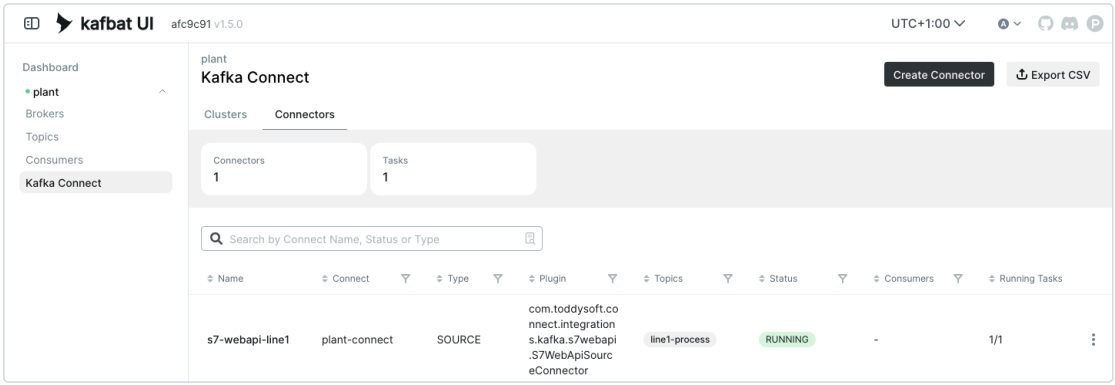
Kafka never logs these connectors' configuration values, and the worker writes the `${file:…}` placeholder, not the value, into its config topic. Restrict the config topic with ACLs all the same.

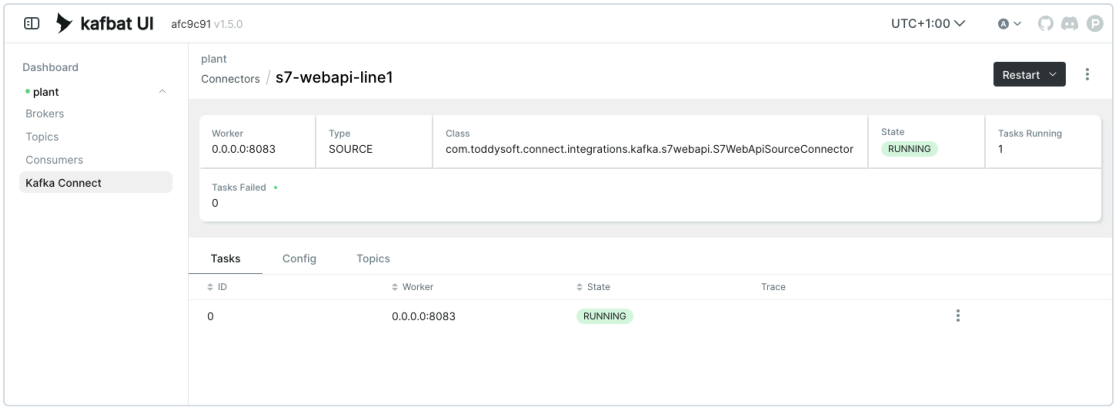
Source connector

```
{
  "name": "s7-webapi-line1",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.s7webapi.S7WebApiSourceConnector",
    "configuration.file": "/etc/kafka-connect/line1.toddysoft.json",
    "batches.topics": "{\"process\": \"line1-process\", \"state\": \"line1-
        state\"}",
    "topic.default": "line1-other",
    "secrets.line1.username": "${file:/etc/kafka-
        connect/secrets/plc.properties:line1.username}",
    "secrets.line1.password": "${file:/etc/kafka-
        connect/secrets/plc.properties:line1.password}",
    "tasks.max": "1"
  }
}
```

`curl -X POST -H 'Content-Type: application/json' --data @s7-webapi-line1.json http://localhost:8083/connectors` creates it.

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 an S7-1200 through its Web API, reading the tags `"TestData".g_b1`, `g_s16` and `g_s32` once a second.

Records

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

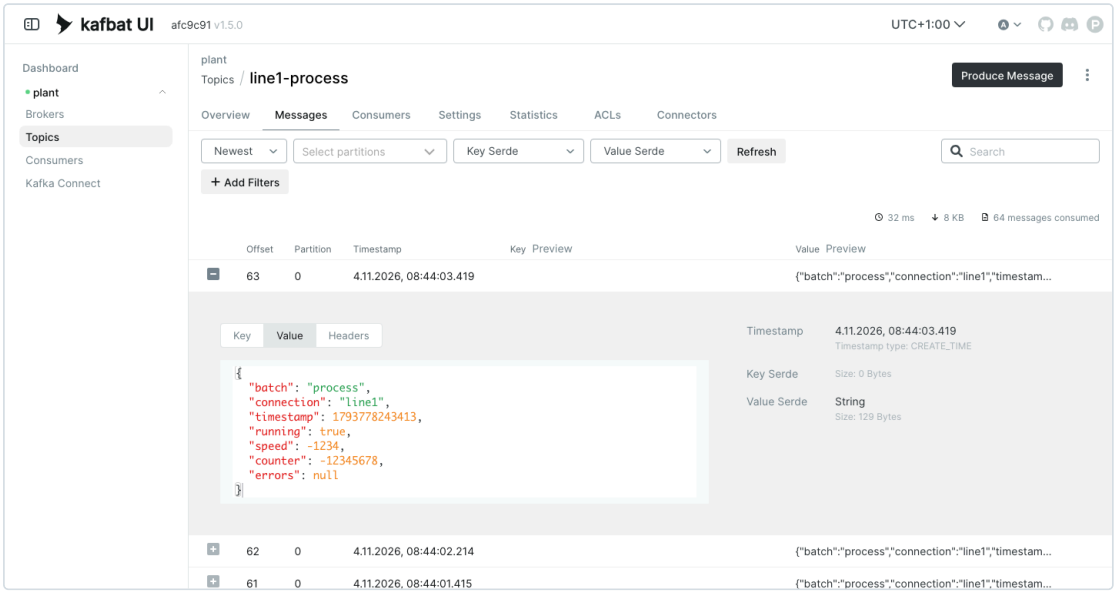
field	type	
`batch`	STRING	the batch id
`connection`	STRING	the connection id
`timestamp`	INT64	the time of the read, milliseconds since the epoch
one field per tag alias	see the type table	`null` when the tag could not be read
`errors`	MAP<STRING,STRING>	alias to the reason, for every tag that could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

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

controller type	Connect type	
BOOL	BOOLEAN	
SINT	INT8	

controller type	Connect type	
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	
CHAR, WCHAR, STRING, WSTRING	STRING	
TIME, LTIME	INT64	milliseconds
TIME_OF_DAY	INT64	milliseconds since midnight
DATE	INT32	days since the epoch
a structure, DTL and DATE_AND_TIME included	STRING	JSON; the Web API transfers them only member by member
an array, `$diag:buffer`, `$diag:alarms`	ARRAY of STRING	each element as text, a structure as JSON
a member the Web API names no type for (a member of a structure it cannot list)	typed by its value: a number INT32 or INT64	add a type suffix where it matters

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": "s7-webapi-line1-setpoints",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.s7webapi.S7WebApiSinkConnector",
    "configuration.file": "/etc/kafka-connect/line1.toddysoft.json",
    "connection": "line1",
    "topics": "line1-setpoints",
    "secrets.line1.username": "${file:/etc/kafka-
        connect/secrets/plc.properties:line1.username}",
    "secrets.line1.password": "${file:/etc/kafka-
        connect/secrets/plc.properties:line1.password}"
  }
}
```

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. The account needs the right to write values, and a tag the TIA project marks read-only is refused whatever the account's rights.

Writing `$control:operating-mode` (`run` / `stop`) stops or starts the CPU. The controller queues every such request and carries out each one, a repeated one too; its acknowledgement means *queued*, not *done*. A retry is therefore a second execution: give a sink that writes the operating mode `max.retries` 0, and confirm a change by reading the mode, not by the acknowledgement.

Several identical PLCs

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

```
{
  "version": 2,
  "tagSets": { "press": { "speed": "\"Press\".speed", "count":
        "\"Press\".count" } },
  "connections": [
    { "id": "press-1", "url": "s7-webapi://192.168.10.31", "tagSets":
        ["press"],
      "credential": { "method": "username-password", "username": "
        {{press.username}}", "password": "{{press.password}}" } },
    { "id": "press-2", "url": "s7-webapi://192.168.10.32", "tagSets":
        ["press"],
      "credential": { "method": "username-password", "username": "
        {{press.username}}", "password": "{{press.password}}" } },
    { "id": "press-3", "url": "s7-webapi://192.168.10.33", "tagSets":
        ["press"],
      "credential": { "method": "username-password", "username": "
        {{press.username}}", "password": "{{press.password}}" } }
  ],
  "batches": [
    { "id": "line-speed", "connectionIds": ["press-1", "press-2", "press-3"],
      "tags": ["speed", "count"],
      "trigger": { "type": "timer", "intervalMillis": 1000 } }
  ]
}
```

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.

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. 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-s7-webapi`, `…-s7`, …). **Use each driver's own connector classes** - `…kafka.s7webapi.S7WebApiSourceConnector` 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.

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.s7webapi.S7WebApiSourceConnector` / `S7WebApiSinkConnector`
the connector is refused with a rule id and a location	the configuration has a problem	fix the file at each location named
… the rule id is `credential-reference-unresolved`	a `{{name}}` without its `secrets.<name>` property	add the property to the connector
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 refused login, a certificate, a licence
`errors` mentions `License validation failed`	no licence applies	see Checking the licence
`errors` says `Server certificate not trusted for <host>:443`	the controller's self-signed certificate	pin the fingerprint from the worker log (see The controller's certificate)

symptom	cause	what to do
`Login failed for user '…'`	wrong password, or the account lacks the Web API right - the controller answers both alike	check the password and the account's rights in the TIA project (see Credentials)
`Login failed for user 'null'`	no credential reached the driver: the connection has no `credential`	add it (see Credentials)
the login works, but tags are refused with `ACCESS_DENIED`	the account lacks the right to read or write values	grant it in the TIA project; `$session:permissions` shows the account's rights
`errors` mentions HTTP `404`	the web server runs, but the Web API is not enabled	enable the Web API in the TIA project
`errors` mentions HTTP `403`	the account lacks the Web API right, or the firmware has no Web API	check the firmware (S7-1200 V4.5, S7-1500 V2.9) and the account
a tag is refused with `NOT_FOUND`	no such variable, or it is not accessible from HMI/Web	check the name and the tag's HMI/Web attribute in the TIA project
a tag is refused with `INVALID_ADDRESS`	usually an unquoted member name with a space or a hyphen, or indices written `[1][1]`	quote the member; write `[1,1]`
a write to an array or a structure is refused with `UNSUPPORTED`	the Web API never writes an aggregate whole	write its elements or members
the TLS handshake fails on an older controller	`tls.version` pinned to TLS 1.3	remove `tls.version` from the URL

symptom	cause	what to do
reads are slow	the Web API is roughly ten times slower per tag than the S7 protocols	use the S7 connector for cyclic process data
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 S7 Web API 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.