ToddySoft Connect UMAS - Apache Kafka Connect

A Kafka Connect **source and sink connector** for Schneider Electric Modicon M340 and M580 controllers, built on the ToddySoft Connect UMAS driver. It reads variables over UMAS, the protocol EcoStruxure Control Expert itself uses, by the **symbol names of the PLC program** - with their types from the controller's data dictionary, not as numbered registers - and publishes every read as one Kafka record; the sink writes record fields to variables.

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 502). UMAS shares the port with Modbus, so a controller you already reach over Modbus needs no further port opened.
- **On the controller**:
  - **The data dictionary must be enabled in the PLC project.** The driver resolves symbol names from the symbol table the controller carries in its data dictionary; a project built without it has no symbols to hand out. Enable it in EcoStruxure Control Expert under **Tools → Project Settings → General → PLC embedded data → Data dictionary**, then rebuild the project and download it: the setting takes effect only with a download.
  - **The PLC program must be running**; a stopped or empty program has no data dictionary to read.
  - **UMAS must not be blocked.** Schneider controllers allow it by default; some site security policies disable it.
  - No password: see Credentials.
- A **ToddySoft licence for UMAS** 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-umas-<version>.zip` - Confluent Hub's component archive format, for Confluent's installer;
- `toddysoft-kafka-connect-umas-<version>-plugin.zip` - a plain plugin directory.

With Confluent's CLI (Confluent Platform)

```
confluent connect plugin install ./toddysoft-kafka-connect-umas-
        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-umas` 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-umas-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 umas
```

lists `com.toddysoft.connect.integrations.kafka.umas.UmasSourceConnector` and `…UmasSinkConnector` 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/umas.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 [umas]`:

```
ToddySoft licence [umas] running on the bundled trial, ends 2027-01-02
ToddySoft licence [umas]   running on: the bundled trial
(ClasspathLicenseSource), checksum 31e3005ea51b, valid until 2027-01-
02
ToddySoft licence [umas]   found and rejected: a licence file
(FileSystemLicenseSource), checksum 9be41f07c2aa:
PLATFORM_NOT_LICENSED: License does not entitle this platform
ToddySoft licence [umas]   licence directory:
/home/appuser/.toddysoft/licenses (derived from user.home)
ToddySoft licence [umas]   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 umas.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 UMAS 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, 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": "umas://192.168.10.20",
      "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": 200 } }
  ]
}
```

- **Connection URL**: `umas://<host>` (port 502). The Modbus unit identifier is `unit-identifier` (default 0, which UMAS normally uses; some gateways and multi-CPU

racks need another): `umas://192.168.10.20?unit-identifier=1`. Each request may take `request-timeout-ms` (default 4000).

- **Addresses** are the variables' symbol names as Control Expert shows them, with a dotted path into a structure and an index per array dimension: `g_s16`, `Process.temperature`, `MyArray[5]`, `MyMatrix[2][3]`, `MyStruct.Substruct.Field[10]`. A whole array or structure is read by its name. Names are **case-sensitive**. Register addresses (`%MW100`) are not UMAS addresses: a located variable is read by its symbol name; plain registers are for the Modbus connector. An array range (`MyArray[0..9]`) is refused with `UNSUPPORTED`.
- **After a Control Expert download** the driver notices the changed program by its CRC, which it polls every `status-poll-interval-ms` (default 5000), and reloads the symbol table; a variable added by the download is available after that poll.
- **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). UMAS has no change notification on the wire: the driver serves a subscription by polling, at `subscription-poll-interval-ms` (default 500) in the URL.
- **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.

Credentials

There are none to configure. UMAS as the driver speaks it is unauthenticated, so a UMAS connection carries no `credential`; one given anyway is refused before the connector starts (rule `driver-credential-unsupported`). Keep the controllers on a segmented plant network that only the workers and the engineering stations reach.

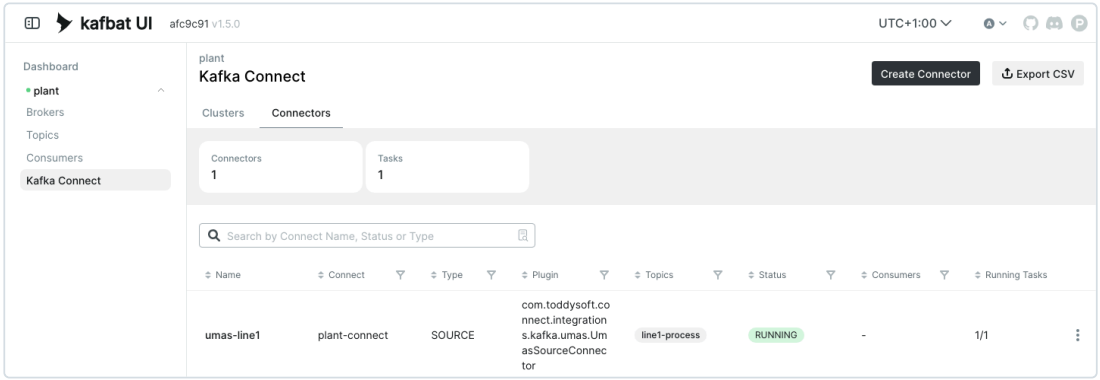
The connectors' other secrets mechanism - a reference `{{name}}` in the file, filled by the connector property `secrets.<name>`, best through a Kafka **config provider** - is therefore not needed for UMAS. Kafka never logs these connectors' configuration values.

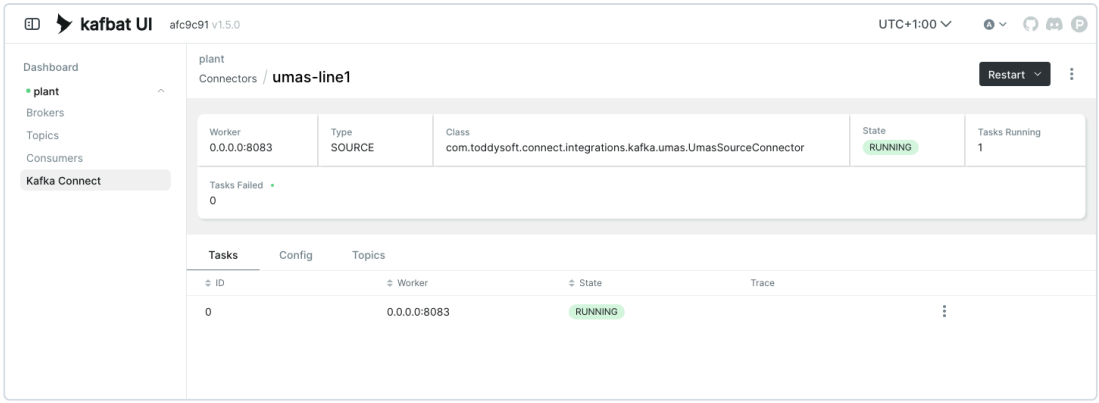
Source connector

```
{
  "name": "umas-line1",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.umas.UmasSourceConnector",
    "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 @umas-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 a Modicon M580 reading the variables `g_b1`, `g_s16` and `g_s32` once a second.

Records

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

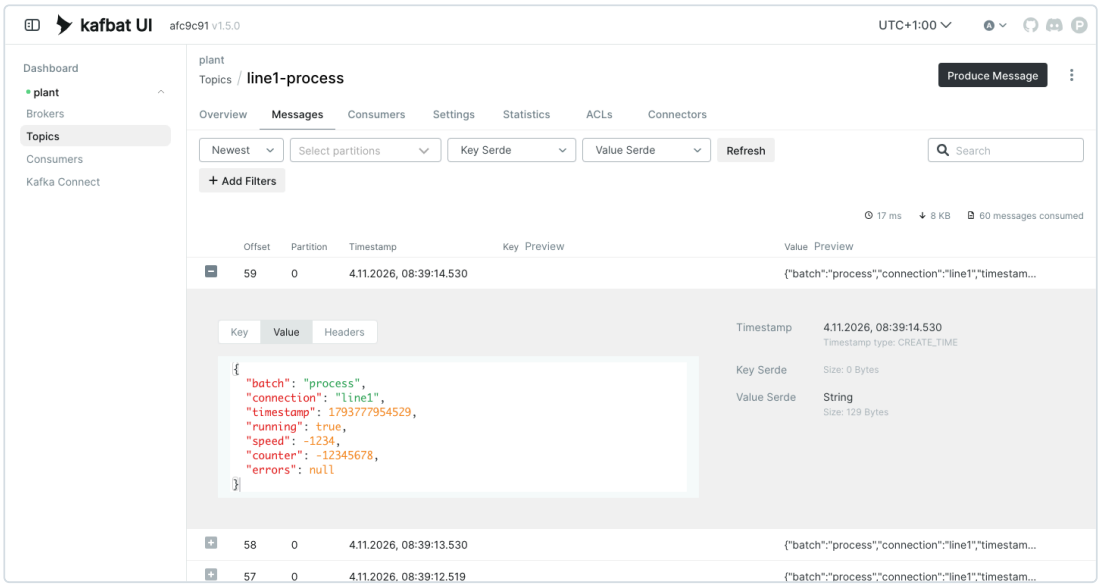
field	type	
`batch`	STRING	the batch id
`connection`	STRING	the connection id
`timestamp`	INT64	the time of the read, milliseconds since the epoch
one field per tag alias	see the type table	`null` when the tag could not be read
`errors`	MAP<STRING,STRING>	alias to the reason, for every tag that could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

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

controller type	Connect type	
BOOL, EBOOL	BOOLEAN	
BYTE, INT	INT16	BYTE widened
WORD, UINT, DINT	INT32	unsigned 16-bit types widened
DWORD, UDINT	INT64	unsigned 32-bit types widened
REAL	FLOAT32	
STRING	STRING	

controller type	Connect type	
TIME	INT64	milliseconds
TIME_OF_DAY (TOD)	INT64	milliseconds since midnight
DATE	INT32	days since the epoch
DATE_AND_TIME (DT)	Timestamp	
a structure (DDT)	STRING	JSON
an array	ARRAY of STRING	each element as text, a structure as JSON
a type the data dictionary does not describe	BYTES	the variable's raw bytes

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": "umas-line1-setpoints",
  "config": {
    "connector.class":
        "com.toddysoft.connect.integrations.kafka.umas.UmasSinkConnector",
    "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": "Press.speed", "count":
        "Press.count" } },
  "connections": [
    { "id": "press-1", "url": "umas://192.168.10.31", "tagSets":
        ["press"] },
    { "id": "press-2", "url": "umas://192.168.10.32", "tagSets":
        ["press"] },
    { "id": "press-3", "url": "umas://192.168.10.33", "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.

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-umas`, `…-modbus`, …). **Use each driver's own connector classes** - `…kafka.umas.UmasSourceConnector` 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.umas.UmasSourceConnector` / `UmasSinkConnector`
the connector is refused with a rule id and a location	the configuration has a problem	fix the file at each location named
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 connection, a licence
`errors` mentions `License validation failed`	no licence applies	see Checking the licence
`Connection refused` on port 502	controller off, wrong address, or a firewall	check the address and that TCP 502 is open from every worker
the TCP connection is made, the UMAS exchange is not	a plain Modbus device, or UMAS disabled by a site security policy	check the device; have UMAS allowed on the controller
`INVALID_ADDRESS` for one tag	the symbol is not in the current program, or a typo in the path or index	check the name in Control Expert; names are case-sensitive

symptom	cause	what to do
`INVALID_ADDRESS` for every tag, although the names are right	the project has no data dictionary, or the program is not running	enable the data dictionary, rebuild and download (see Requirements); start the program
a variable added by a Control Expert download is not found at once	the program CRC has not been polled yet	wait one `status-poll-interval-ms` (default 5 s)
`%MW…` or other register addresses refused	UMAS addresses variables by symbol name	use the variable's symbol name, or the Modbus connector for plain registers
an array range (`[0..9]`) refused with `UNSUPPORTED`	the driver reads a whole array or single elements only	read the array by its name, or the elements one by one
a `subscription` batch misses fast changes	subscriptions are polled	shorten `subscription-poll-interval-ms` , accepting the extra load on the controller
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.

- **Fixed:** writing an array range (`MyArray[1..8]`) is refused as unsupported, as reading one is; it was reported as a variable that does not exist.

- **New:** the UMAS 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.