ToddySoft Connect UMAS - Apache Kafka Connect

Ein Kafka-Connect-**Source- und Sink-Connector** für Steuerungen Schneider Electric Modicon M340 und M580 auf Basis des UMAS-Treibers von ToddySoft Connect. Er liest Variablen über UMAS, das Protokoll, das EcoStruxure Control Expert selbst verwendet, über die **Symbolnamen des SPS-Programms** - mit ihren Typen aus dem Datenwörterbuch der Steuerung, nicht als nummerierte Register - und veröffentlicht jeden Lesevorgang als einen Kafka-Datensatz; der Sink schreibt Felder von Datensätzen in Variablen.

Der Connector läuft auf **Ihren eigenen Kafka-Connect-Workern** in der Nähe der Anlage: Connect aus Apache Kafka oder Confluent Platform, als Dienst oder im Container. Er läuft **nicht** in Confluent Cloud: Ein verwalteter oder eigener Connector läuft dort im Netz von Confluent und erreicht keine Steuerung in einem Anlagennetz.

Im Vergleich zum Connector von Apache PLC4X

Apache PLC4X, das Open-Source-Projekt, aus dem die Treiber von ToddySoft Connect hervorgegangen sind, veröffentlicht einen freien Kafka-Connect-Connector. Stand PLC4X 1.0.0 sind die Unterschiede:

	dieser Connector	**Connector von Apache PLC4X 1.0.0**
Zugangsdaten	als solche an den Treiber übergeben, geliefert von Config Providern von Kafka; nie in der Verbindungs-URL, nie im Log	nur, was in die Verbindungszeichenfolge passt
Connector-Konfiguration im Log des Workers	wird nie protokolliert	jede Eigenschaft wird beim Start einer Task protokolliert, Verbindungszeichenfolgen eingeschlossen
Wann Variablen gelesen werden	per Timer, bei einer Wertänderung (mit Bedingung) oder per Abonnement	in einem festen Intervall
Konfiguration	eine versionierte Datei, gemeinsam mit der Camel-Komponente, vor dem Start geprüft, jeder Fehler mit seiner Stelle	flache Connector-Eigenschaften

	dieser Connector	Connector von Apache PLC4X 1.0.0
Mehrere gleiche Steuerungen	ein Batch über einen Variablensatz für viele Verbindungen, verteilt auf Tasks	eine Quelle je Steuerung
Paketierung	ein Plugin je Treiber, als Plugin-ZIP und Komponentenarchiv für Confluent Hub	ein Plugin mit allen PLC4X-Treibern
Lizenz und Support	kommerzielle Lizenz; Support durch ToddySoft	Apache License 2.0, frei; Support durch die Community

Voraussetzungen

	unterstützt	getestet
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 auf dem Worker	17 oder neuer	21, 25

- **Kafka vor 3.9 und Java vor 17 werden nicht unterstützt.** (Unter Java 11 kann der Worker das Plugin gar nicht laden; siehe Fehlersuche.)
- **Nur selbst betriebene Worker**, standalone oder verteilt. Jeder Worker, der eine Task ausführen kann, braucht das Plugin, die Konfigurationsdatei (falls Sie eine verwenden) und die Lizenz.
- **Eine Route von jedem Worker zu den Steuerungen** (TCP-Port 502). UMAS teilt sich den Port mit Modbus; eine Steuerung, die Sie bereits über Modbus erreichen, braucht keinen weiteren freigegebenen Port.
- **In der Steuerung**:
  - **Das Datenwörterbuch muss im SPS-Projekt aktiviert sein.** Der Treiber löst Symbolnamen über die Symboltabelle auf, die die Steuerung in ihrem Datenwörterbuch führt; ein Projekt ohne Datenwörterbuch hat keine Symbole herauszugeben. Aktivieren Sie es in EcoStruxure Control Expert unter **Tools → Project Settings → General → PLC embedded data → Data dictionary**, erstellen Sie das Projekt dann neu und laden Sie es in die Steuerung: Die Einstellung wirkt erst mit einem Download.
  - **Das SPS-Programm muss laufen**; ein gestopptes oder leeres Programm hat kein Datenwörterbuch, das sich lesen ließe.
  - **UMAS darf nicht gesperrt sein.** Schneider-Steuerungen erlauben es standardmäßig; manche Sicherheitsrichtlinien eines Standorts schalten es ab.
  - Kein Passwort: siehe Zugangsdaten.
- Eine **ToddySoft-Lizenz für UMAS** für den produktiven Einsatz. Ohne sie arbeitet der Treiber mit seiner mitgelieferten Testlizenz, bis diese abläuft.

Installation

Sie erhalten das Plugin in zwei Formen in der Freigabe, aus der Sie auch Lizenz und Downloads beziehen; beide enthalten denselben Connector:

- `toddysoft-kafka-connect-umas-<version>.zip` - im Komponentenarchiv-Format von Confluent Hub, für Confluents Installationsprogramm;
- `toddysoft-kafka-connect-umas-<version>-plugin.zip` - ein einfaches Plugin-Verzeichnis.

Mit dem Confluent 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
```

Das CLI installiert aus der lokalen Datei, ohne Internetzugang, nach `<plugin-directory>/toddysoft-kafka-connect-umas` und trägt das Verzeichnis in `plugin.path` der Worker-Konfiguration ein. `--force` akzeptiert die angezeigte Lizenzvereinbarung; lassen Sie es weg, um sie selbst zu lesen und zu akzeptieren. Es läuft nicht unter Windows und nicht, solange Sie bei Confluent Cloud angemeldet sind. Ältere Versionen von Confluent Platform haben stattdessen `confluent-hub install <zip> --component-dir <dir> --worker-configs <datei> --no-prompt`.

Durch Entpacken (jeder Worker)

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

`/opt/kafka/plugins` muss in `plugin.path` des Workers stehen (`plugin.path=/opt/kafka/plugins`). Starten Sie danach jeden Worker neu.

Im Container

Binden Sie das entpackte Plugin und das Lizenzverzeichnis in den Worker-Container ein, zum Beispiel mit dem offiziellen Image `apache/kafka` (es läuft als `appuser` mit dem Heimatverzeichnis `/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` läuft ebenfalls als `appuser` mit demselben Heimatverzeichnis; sein vorgegebener Plugin-Pfad ist `/usr/share/confluent-hub-components`.

Die Installation prüfen

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

listet `com.toddysoft.connect.integrations.kafka.umas.UmasSourceConnector` und `…UmasSinkConnector` mit ihrer Version.

Aktualisieren

Entfernen Sie das alte Plugin-Verzeichnis, bevor Sie das neue installieren (das Confluent CLI tut das selbst): Liegen zwei Versionen eines Plugins auf `plugin.path`, lädt der Worker eine beliebige. Starten Sie jeden Worker neu.

Lizenz

Der Treiber liest seine Lizenz aus **`.lic`-Dateien in `~/.toddysoft/licenses/` des Benutzers, unter dem der Worker läuft**:

der Worker läuft	die Lizenzdatei gehört nach
als Dienst, Benutzer `kafka`	`~kafka/.toddysoft/licenses/umas.lic` (zum Beispiel `/home/kafka/.toddysoft/licenses/`)
im Container `apache/kafka` oder `cp-kafka-connect`	ein Host-Verzeichnis, eingebunden unter `/home/appuser/.toddysoft/licenses` (siehe oben)
in einem verteilten Cluster	dasselbe auf **jedem** Worker: Eine Task, die auf einen Worker ohne Lizenz wandert, läuft dort mit der Testlizenz

Der Dateiname ist gleichgültig, nur die Endung `.lic` zählt; eine Datei je Treiber oder eine Lizenz für mehrere Treiber funktioniert beides. Eine neue Lizenz gilt ab dem nächsten Start des Connectors oder der Task (starten Sie den Connector neu). Ohne Lizenz, oder mit einer, die der Treiber ablehnt, gilt die mitgelieferte Testlizenz bis zu ihrem Ablauf; danach lehnt der Treiber seine Verbindungen ab.

Die Lizenz prüfen

Jede Task schreibt beim Start einen **Lizenzbericht** in das Log des Workers, einen Block je Treiber, jede Zeile beginnt mit `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
```



Sie finden ihn mit `grep 'ToddySoft licence' <Worker-Log>` (im Container: `docker logs connect 2>&1 | grep 'ToddySoft licence'`). Eine Lizenz wird über ihre **Prüfsumme** benannt, die ersten 12 Zeichen ihres SHA-256: `shasum -a 256 umas.lic` zeigt, welche Datei gemeint ist.

der Bericht sagt	Bedeutung	was zu tun ist
`licensed` · running on: a licence file	Ihre Lizenz gilt	nichts
`licensed, expires in N days`	sie läuft bald ab	rechtzeitig verlängern
`licensed in its grace period: expired on <date>, refused after <date>`	sie ist abgelaufen; ToddySoft Connect akzeptiert sie noch 30 Tage (Nachfrist)	jetzt verlängern: nach diesem Datum wird die Lizenz abgelehnt
`attention: Support ends in N days (<date>)` / `attention: Support ended on <date>`	Treiberversionen, die nach diesem Datum gebaut wurden, laufen mit dieser Lizenz nicht	den Support verlängern, bevor das Plugin aktualisiert wird
`running on the bundled trial, ends <Datum>` und keine Zeile *found and rejected*	keine Lizenzdatei gefunden	die Datei in das genannte Verzeichnis legen, für den genannten Benutzer
… und `found and rejected: …: PLATFORM_NOT_LICENSED`	die Lizenz deckt Kafka Connect nicht ab	bei ToddySoft eine Lizenz anfordern, die es tut
… `DRIVER_NOT_LICENSED`	es ist die Lizenz eines anderen Treibers	die UMAS-Lizenz verwenden
… `LICENSE_EXPIRED`	sie ist abgelaufen	verlängern

der Bericht sagt	Bedeutung	was zu tun ist
`NOT licensed: …`	keine Lizenz gilt, auch nicht die Testlizenz; der Treiber lehnt jede Verbindung ab	wie oben

Lehnt der Treiber eine Verbindung wegen der Lizenz ab, erscheint derselbe Block einmal, danach wird nur gezählt (`refused N connections since …, licence unchanged` , höchstens alle zehn Minuten), statt das Log zu füllen. Nichts im Log zeigt die Lizenz selbst oder auf wen sie lautet. Die Task schlägt wegen der Lizenz nicht fehl: Der Connector bleibt RUNNING, und seine Datensätze tragen die Ablehnung in `errors` ; die Ursache steht im Bericht.

Die Konfigurationsdatei

Beide Connectoren führen eine **Event-Pump-Konfiguration** von ToddySoft Connect aus (Version 2): die Verbindungen, die Variablen unter Aliasnamen und Batches - welche Variablen zusammen gelesen werden, und wann. Das Verzeichnis `etc/` des Archivs enthält ein Beispiel; ein vollständiges:

```
{
  "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 } }
  ]
}
```

- **Verbindungs-URL**: `umas://<host>` (Port 502). Die Modbus-Unit-ID ist `unit-identifier` (Vorgabe 0, die UMAS üblicherweise verwendet; manche Gateways und

Racks mit mehreren CPUs brauchen eine andere): `umas://192.168.10.20?unit-identifier=1`. Jede Anfrage darf `request-timeout-ms` dauern (Vorgabe 4000).

- **Adressen** sind die Symbolnamen der Variablen, wie Control Expert sie zeigt, mit einem Punktpfad in eine Struktur und einem Index je Array-Dimension: `g_s16`, `Process.temperature`, `MyArray[5]`, `MyMatrix[2][3]`, `MyStruct.Substruct.Field[10]`. Ein ganzes Array oder eine ganze Struktur wird über ihren Namen gelesen. Namen unterscheiden **Groß- und Kleinschreibung**. Registeradressen (`%MW100`) sind keine UMAS-Adressen: Eine lokalisierte Variable wird über ihren Symbolnamen gelesen; einfache Register sind Sache des Modbus-Connectors. Ein Array-Ausschnitt (`MyArray[0..9]`) wird mit `UNSUPPORTED` abgelehnt.
- **Nach einem Download aus Control Expert** erkennt der Treiber das geänderte Programm an seiner CRC, die er alle `status-poll-interval-ms` abfragt (Vorgabe 5000), und lädt die Symboltabelle neu; eine mit dem Download hinzugekommene Variable ist nach dieser Abfrage verfügbar.
- **Auslöser**: `timer` (alle `intervalMillis`), `change` (den Batch lesen, wenn sich eine beobachtete Variable ändert, wahlweise unter einer `condition` wie `oldValue == false && value == true`), `subscription` (der Treiber abonniert und reicht jede Änderung weiter). UMAS kennt keine Änderungsmeldungen auf der Leitung: Der Treiber bedient ein Abonnement durch Abfragen, im Takt von `subscription-poll-interval-ms` in der URL (Vorgabe 500).
- **Zugriff**: `read`-Variablen werden nie geschrieben, `write`-Variablen nie gelesen; ohne `access` beides.
- **Mehrere gleiche Steuerungen**: siehe unten.

Übergeben Sie die Datei dem Connector entweder direkt (`configuration`) oder als Pfad auf **jedem** Worker (`configuration.file`). Eine Konfiguration mit Fehlern weist den Connector ab und nennt jeden Fehler mit seiner Stelle in der Datei und einer Regelkennung.

Zugangsdaten

Es gibt keine einzurichten. UMAS ist, so wie der Treiber es spricht, nicht authentifiziert; eine UMAS-Verbindung trägt daher kein `credential`, und ein dennoch angegebenes wird vor dem Start des Connectors abgewiesen (Regel `driver-credential-unsupported`). Halten Sie die Steuerungen in einem abgetrennten Anlagennetz, das nur die Worker und die Engineering-Stationen erreichen.

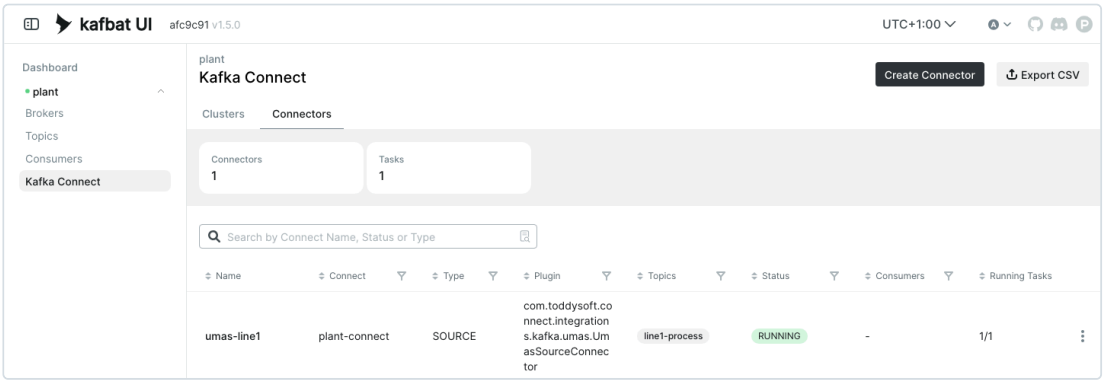
Der übrige Mechanismus der Connectoren für Geheimnisse - ein Verweis `{{name}}` in der Datei, gefüllt von der Connector-Eigenschaft `secrets.<name>`, am besten über einen **Config Provider** von Kafka - wird für UMAS deshalb nicht gebraucht. Kafka protokolliert die Konfigurationswerte dieser Connectoren nie.

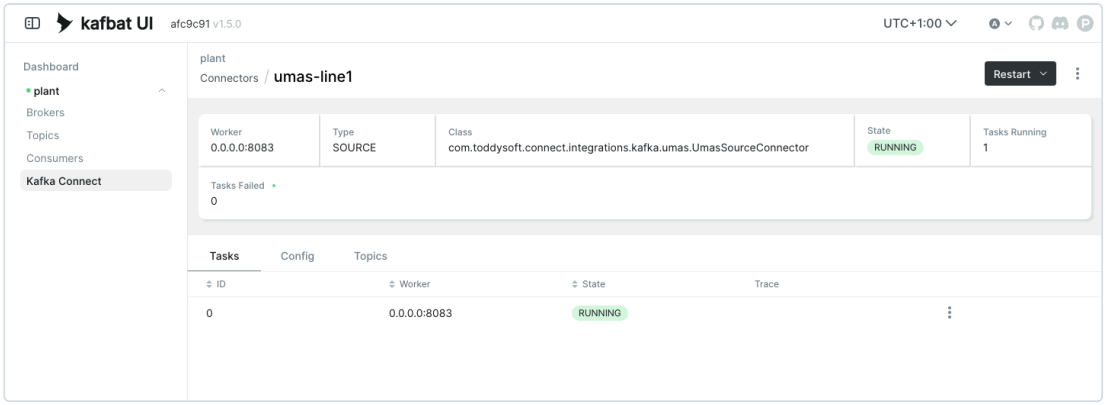
Der Source-Connector

```
{
  "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` legt ihn an.

Eigenschaft	Vorgabe	Bedeutung
`configuration` / `configuration.file`	-	die Konfiguration, direkt oder als Pfad auf jedem Worker; genau eines davon
`batches`	alle	die auszuführenden Batches, durch Kommas getrennt
`batches.topics`	`{}`	JSON-Objekt: Batch zu Topic
`topic.default`	-	Topic jedes Batches, der nicht in `batches.topics` steht
`secrets.<name>`	-	der Wert von `{{name}}`
`tasks.max`	1	Batches (und Verbindungen, siehe unten) werden auf höchstens so viele Tasks verteilt





Die Bildschirmfotos zeigen kafka-ui (kafbat UI), eine freie Weboberfläche für Kafka und Kafka Connect; jedes Werkzeug, das Connectoren und Topics zeigt, eignet sich, unter Confluent Platform zum Beispiel Confluent Control Center. Sie entstanden an einer Modicon M580, die die Variablen `g_b1`, `g_s16` und `g_s32` einmal pro Sekunde liest.

Datensätze

Ein Datensatz je Lesevorgang. Sein Wert ist ein `Struct`:

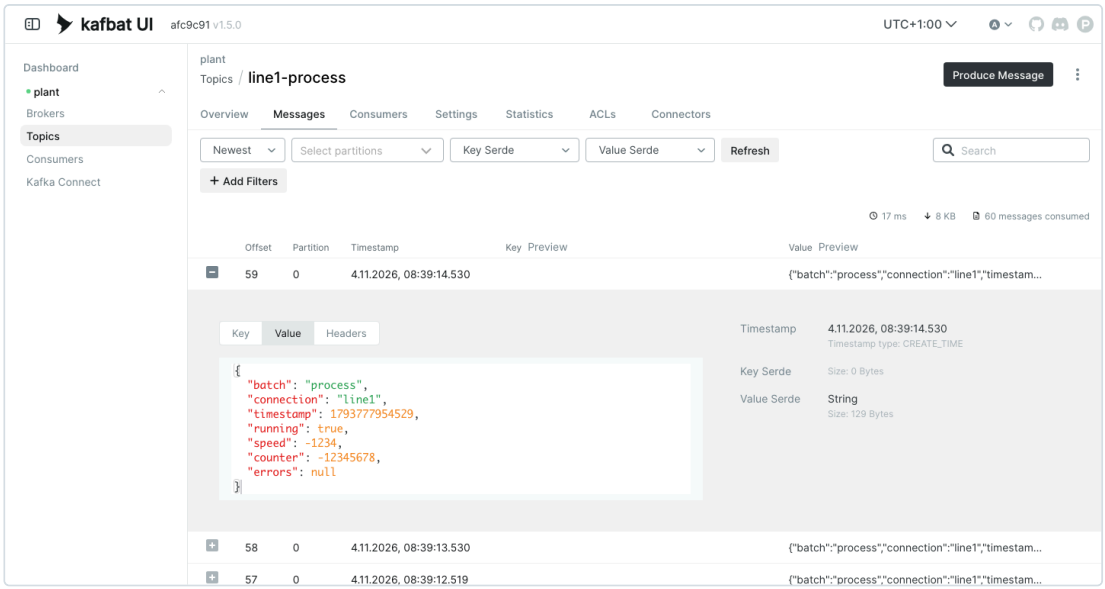
Feld	Typ	
`batch`	STRING	der Batch
`connection`	STRING	die Verbindung
`timestamp`	INT64	der Zeitpunkt des Lesens, Millisekunden seit der Epoche
ein Feld je Variablen-Alias	siehe Typtabelle	`null`, wenn die Variable nicht gelesen werden konnte
`errors`	MAP<STRING,STRING>	Alias zu Grund, für jede Variable, die nicht gelesen werden konnte

Der Datensatz hat **keinen Schlüssel**; seine Source-Partition ist `{batch, connection}`, sein Offset `{timestamp}`.

Typ in der Steuerung	Connect-Typ	
BOOL, EBOOL	BOOLEAN	
BYTE, INT	INT16	BYTE erweitert
WORD, UINT, DINT	INT32	vorzeichenlose 16-Bit-Typen erweitert
DWORD, UDINT	INT64	vorzeichenlose 32-Bit-Typen erweitert
REAL	FLOAT32	
STRING	STRING	

Typ in der Steuerung	Connect-Typ	
TIME	INT64	Millisekunden
TIME_OF_DAY (TOD)	INT64	Millisekunden seit Mitternacht
DATE	INT32	Tage seit der Epoche
DATE_AND_TIME (DT)	Timestamp	
eine Struktur (DDT)	STRING	JSON
ein Array	ARRAY of STRING	jedes Element als Text, eine Struktur als JSON
ein Typ, den das Datenwörterbuch nicht beschreibt	BYTES	die Rohbytes der Variablen

Ein Feld erhält seinen Typ beim **ersten Lesen** des Batches, und jedes Feld ist optional: Eine Variable, die beim ersten Lesen nicht gelesen werden konnte, ist danach ein STRING - starten Sie den Connector neu, sobald die Steuerung antwortet.



Konverter und Datenformate

Der Connector erzeugt Connect-Datensätze vom Typ `Struct` mit Schema; der Konverter des Workers bestimmt die Bytes im Topic:

- `org.apache.kafka.connect.json.JsonConverter` mit `value.converter.schemas.enable=true` (Schema und Nutzdaten) oder `false` (nur Nutzdaten);
- `io.confluent.connect.avro.AvroConverter` oder `io.confluent.connect.protobuf.ProtobufConverter` mit Confluents Schema Registry (`value.converter.schema.registry.url=http://registry:8081`), auf einem Worker, der diese Konverter hat (Confluent Platform hat sie).

Setzen Sie den Konverter je Connector (`"value.converter": "…"`) oder für den Worker. Jeder Lesevorgang eines Batches hat dasselbe Schema; eine zum Batch hinzugefügte Variable ändert es.

Der Sink-Connector

Schreibt die Felder jedes Datensatzes in die Variablen einer Verbindung, über den 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"
  }
}
```

Eigenschaft	Vorgabe	Bedeutung
`connection`	-	die Verbindung, in die geschrieben wird
`topics`	-	die gelesenen Topics
`max.retries`	3	wie oft ein Schreibvorgang, den die Steuerung abgelehnt hat, wiederholt wird
`retry.backoff.ms`	1000	die erste Wartezeit vor einer Wiederholung, jedes Mal verdoppelt
`retry.backoff.max.ms`	30000	die längste Wartezeit

Ein Datensatz ist ein `Struct` (mit einem Konverter, der ein Schema mitführt, zum Beispiel JSON mit Schema oder Avro); jedes Feld nennt einen Variablen-Alias der Verbindung, sein Typ folgt der Typtabelle. Ein Feld, das die Verbindung nicht kennt, oder eine Variable mit dem Zugriff `read` wird abgewiesen: Der Datensatz geht ungeschrieben an die Fehlerbehandlung von Kafka Connect (eine Dead Letter Queue mit `errors.tolerance=all` und `errors.deadletterqueue.topic.name`). Ein Schreibvorgang, den die Steuerung ablehnt, wird wiederholt und dann ebenso gemeldet. Ein wegen der Lizenz abgelehnter Schreibvorgang wird nicht wiederholt.

Mehrere gleiche Steuerungen

Ein Batch mit `connectionIds` läuft einmal je Verbindung - meist liest er einen Variablensatz, den jede Steuerung hat:

```
{
  "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 } }
  ]
}
```

Jede Presse wird als eigene Einheit gelesen, verteilt auf `min(Einheiten, tasks.max)` Tasks: Mit `tasks.max` 3 in einem Cluster aus drei Workern liest jeder Worker eine Presse. Die Datensätze behalten den Batch `line-speed` und nennen die Verbindung, sodass ein Topic alle Pressen enthält, partitioniert nach Presse.

Im **verteilten Modus** verschiebt Kafka Connect Tasks bei einem Rebalance zwischen Workern; eine verschobene Task verbindet sich von ihrem neuen Worker aus und schreibt dort ihren Lizenzbericht. Die Tasks eines gestoppten Workers wandern erst nach bis zu fünf Minuten (`scheduled.rebalance.max.delay.ms`): so lange wartet Kafka Connect, ob der Worker zurückkommt. Im Standalone-Modus läuft alles auf einem Worker.

Mehrere Treiber-Plugins auf einem Worker

Jeder Treiber ist ein eigenes Plugin, und sie liegen nebeneinander auf einem Worker (`/opt/kafka/plugins/toddysoft-kafka-connect-umas`, `…-modbus`, …). **Verwenden Sie die Connector-Klassen des jeweiligen Treibers** - hier `kafka.umas.UmasSourceConnector`: Ein Worker lädt eine Connector-Klasse nur aus einem Plugin. Die allgemeinen Klassen `PlcSourceConnector` / `PlcSinkConnector` gibt es in jedem Plugin; sie sind nicht zur Verwendung gedacht. Eine Task, die den Treiber ihrer URL nicht findet, startet nicht und nennt die richtige Klasse. Innerhalb eines Plugins teilen sich ein Source- und ein Sink-Connector zur selben Steuerung eine Verbindung.

Fehlersuche

Symptom	Ursache	was zu tun ist
das Plugin fehlt in `GET /connector-plugins`	nicht auf `plugin.path`, oder der Worker wurde nicht neu gestartet	`plugin.path` und Verzeichnis prüfen, jeden Worker neu starten
das Plugin fehlt in der Liste, und das Worker-Log zeigt `Failed to discover SourceConnector in …` mit der Ursache `UnsupportedClassVersionError … (class file version 61.0)`	der Worker läuft mit Java 11 oder älter	den Worker mit Java 17 oder neuer betreiben
eine Task schlägt mit `driver-missing` fehl	die allgemeine oder die Klasse eines anderen Treibers	… `kafka.umas.UmasSourceConnector` / `UmasSinkConnector` verwenden
der Connector wird mit Regelkennung und Stelle abgewiesen	die Konfiguration enthält einen Fehler	die Datei an jeder genannten Stelle korrigieren
RUNNING, aber jeder Datensatz hat die Variablen `null` und den Grund in `errors`	die Steuerung lehnt ab oder ist nicht erreichbar	`errors` lesen: eine Adresse, eine abgelehnte Verbindung, eine Lizenz
`errors` nennt `License validation failed`	keine Lizenz gilt	siehe Die Lizenz prüfen
`Connection refused` auf Port 502	Steuerung aus, falsche Adresse oder eine Firewall	die Adresse prüfen und dass TCP 502 von jedem Worker aus offen ist
die TCP-Verbindung kommt zustande, der UMAS-Austausch nicht	ein reines Modbus-Gerät, oder UMAS durch eine Sicherheitsrichtlinie des Standorts abgeschaltet	das Gerät prüfen; UMAS in der Steuerung zulassen lassen
`INVALID_ADDRESS` für eine Variable	das Symbol fehlt im aktuellen Programm, oder ein Tippfehler in Pfad oder Index	den Namen in Control Expert prüfen; Groß- und Kleinschreibung zählen

Symptom	Ursache	was zu tun ist
`INVALID_ADDRESS` für jede Variable, obwohl die Namen stimmen	das Projekt hat kein Datenwörterbuch, oder das Programm läuft nicht	das Datenwörterbuch aktivieren, neu erstellen und laden (siehe Voraussetzungen); das Programm starten
eine mit einem Download aus Control Expert hinzugekommene Variable wird nicht sofort gefunden	die CRC des Programms wurde noch nicht abgefragt	ein `status-poll-interval-ms` abwarten (Vorgabe 5 s)
`%MW…` oder andere Registeradressen werden abgelehnt	UMAS adressiert Variablen über den Symbolnamen	den Symbolnamen der Variablen verwenden, oder für einfache Register den Modbus-Connector
ein Array-Ausschnitt (`[0..9]`) wird mit `UNSUPPORTED` abgelehnt	der Treiber liest nur ganze Arrays oder einzelne Elemente	das Array über seinen Namen lesen, oder die Elemente einzeln
ein `subscription` -Batch verpasst schnelle Änderungen	Abonnements werden abgefragt	`subscription-poll-interval-ms` verkürzen, mit der zusätzlichen Last auf der Steuerung
merkwürdiges Verhalten nach einer Aktualisierung	zwei Kopien des Plugins auf `plugin.path` : Der Worker listet es einmal und protokolliert nichts, man sieht also nicht, welche Kopie läuft	das alte Verzeichnis entfernen, neu starten

Mehr Einzelheiten liefert eine höhere Protokollstufe des Treibers: `curl -X PUT -H 'Content-Type: application/json' --data '{"level":"DEBUG"}' http://localhost:8083/admin/loggers/com.toddysoft` (und danach zurück auf `INFO`).

Versionshistorie

1.3.0 — 2026-11-13

Basiert auf ToddySoft Connect 1.3.0.

- **Behoben:** Ein Schreibzugriff auf einen Array-Bereich (`MyArray[1..8]`) wird wie ein Lesezugriff als nicht unterstützt abgelehnt; bisher wurde er als nicht vorhandene Variable gemeldet.

- **Neu:** der UMAS-Connector als Produkt für selbst betriebene Kafka-Connect-Worker - dieses Handbuch, ein Komponentenarchiv für Confluent Hub neben der Plugin-ZIP, beide von jedem Release geprüft; ein Lizenzbericht je Treiber im Log des Workers; unterstützte Versionen Apache Kafka 3.9.x und 4.0-4.4, Confluent Platform 7.9 und 8.0-8.4.