ToddySoft Connect S7 Web API - Apache Kafka Connect

Ein Kafka-Connect-**Source- und Sink-Connector** für Siemens-Steuerungen S7-1200 und S7-1500 auf Basis des S7-Web-API-Treibers von ToddySoft Connect. Er liest Variablen über die **SIMATIC Web API** der Steuerung - JSON-RPC über HTTPS, mit den Benutzerkonten der Benutzerverwaltung der Steuerung - und veröffentlicht jeden Lesevorgang als einen Kafka-Datensatz; der Sink schreibt Felder von Datensätzen in Variablen.

Die Web API ist die Verwaltungsschnittstelle der Steuerung, nicht ihr Weg für Prozessdaten: Sie ist je Variable etwa eine Größenordnung langsamer als die S7-Protokolle und kennt keine Änderungsmeldung, jeder Lesevorgang ist also eine Abfrage. Sie erreicht, was die S7-Protokolle nicht erreichen - den Diagnosepuffer, aktive Alarme, den Betriebszustand - und funktioniert, wo die Netzwerkrichtlinie nur HTTPS zulässt. **Für zyklische Prozessdaten verwenden Sie den S7-Connector**; diesen für Diagnose, gelegentliches Lesen und Schreiben und Netze, die nur HTTPS erlauben.

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**
Siemens S7 Web API (HTTPS, Benutzerkonten)	ja	kein Treiber
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

	dieser Connector	Connector von Apache PLC4X 1.0.0
Konfiguration	eine versionierte Datei, gemeinsam mit der Camel-Komponente, vor dem Start geprüft, jeder Fehler mit seiner Stelle	flache Connector-Eigenschaften
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 443, HTTPS).
- **In der Steuerung**: eine S7-1200 ab Firmware V4.5 oder eine S7-1500 ab V2.9 (S7-300 und S7-400 haben keine Web API). Im TIA-Projekt muss der **Webserver** eingeschaltet und die **Web API** aktiviert sein - ein eigener Schalter. Der Connector meldet sich mit einem **Konto der Benutzerverwaltung der Steuerung** an, das die Web-API-Berechtigung besitzt, dazu die Berechtigungen für das, was er tut: Werte lesen und Werte schreiben sind getrennte Berechtigungen (siehe Zugangsdaten). Variablen sind nur sichtbar, wenn sie im TIA-Projekt als erreichbar aus HMI/Web markiert sind.
- Eine **ToddySoft-Lizenz für S7 Web API** für den produktiven Einsatz. Eine Lizenz für den S7-Treiber deckt sie nicht ab. 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-s7-webapi-<version>.zip` - im Komponentenarchiv-Format von Confluent Hub, für Confluents Installationsprogramm;
- `toddysoft-kafka-connect-s7-webapi-<version>-plugin.zip` - ein einfaches Plugin-Verzeichnis.

Mit dem Confluent 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
```

Das CLI installiert aus der lokalen Datei, ohne Internetzugang, nach `<plugin-directory>/toddysoft-kafka-connect-s7-webapi` 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-s7-webapi-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 s7webapi
```

listet `com.toddysoft.connect.integrations.kafka.s7webapi.S7WebApiSourceConnector` und `…S7WebApiSinkConnector` 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**:

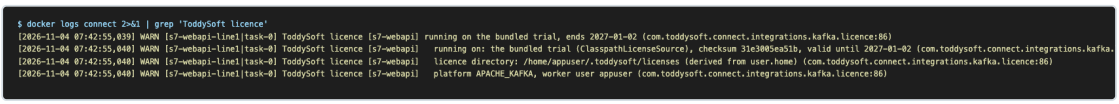
der Worker läuft	die Lizenzdatei gehört nach
als Dienst, Benutzer `kafka`	`~kafka/.toddysoft/licenses/s7-webapi.lic` (zum Beispiel `/home/kafka/.toddysoft/licenses/`)
im Container `apache/kafka` oder `cp-kafka-connect`	ein Host-Verzeichnis, eingebunden unter `/home/appuser/.toddysoft/licenses` (siehe 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 [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
```



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 s7-webapi.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

der Bericht sagt	Bedeutung	was zu tun ist
`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 - auch eine S7-Lizenz	die S7-Web-API-Lizenz verwenden
… `LICENSE_EXPIRED`	sie ist abgelaufen	verlängern
`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 mit ihren Zugangsdaten, 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": "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 } }
  ]
}
```

- **Verbindungs-URL**: `s7-webapi://<host>` (Port 443; ein anderer Port als `s7-webapi://<host>:<port>`). Die Web API spricht nur HTTPS; es gibt keinen Modus mit einfachem HTTP und keinen Transport zur Auswahl. Die Einstellungen zum Zertifikat stehen im nächsten Abschnitt. `request-timeout-ms` (Vorgabe 10000) begrenzt eine Anfrage; die Vorgaben passen zu jeder aktuellen Firmware von S7-1200 und S7-1500.
- **Adressen** sind nur symbolisch, wie TIA Portal sie zeigt: `"DataBlock".member`, `"DataBlock".struct.member`, eine PLC-Variable der globalen Variablentabelle `"ManualTrigger"`, ein Array-Element `"DataBlock".values[2]` (die Indizes eines mehrdimensionalen Arrays durch Kommas getrennt in einer Klammer: `"DataBlock".matrix[1,1]`). Ein Member, dessen Name etwas anderes als Buchstaben, Ziffern und `_` enthält, muss in Anführungszeichen stehen: `"OPC_UA_DB"."OPC Data".g_b1`. Es gilt der in der Steuerung deklarierte Typ; eine Typangabe (`"DataBlock".duration:TIME`) ist nur nötig, wo die Web API keinen nennt. Ein Array oder eine Struktur, als Ganzes benannt, wird als Ganzes gelesen, aber nie als Ganzes geschrieben - schreiben Sie seine Elemente oder ihre Member. Ein Array behält seine deklarierte Untergrenze: Eines kann bei 0 beginnen und das nächste bei 1. Absolute `%DB`-Adressen gibt es nicht: Die Adressen sind nicht mit denen des S7-Connectors austauschbar. In JSON wird ein `"` in einer Adresse als `\"` geschrieben.
- **Diagnose** wird wie Variablen gelesen: `$diag:buffer[0..19]` (die 20 neuesten Einträge des Diagnosepuffers), `$diag:alarms` (aktive Alarme), `$session:permissions` (die Berechtigungen des Kontos), `$control:operating-mode:STRING` (der Betriebszustand; lesen Sie Der Sink-Connector, bevor Sie ihn schreiben).
- **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). Die

Web API kennt keine Änderungsmeldung: Der Treiber bedient ein Abonnement durch Abfragen, höchstens alle 500 ms.
- **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.

Das Zertifikat der Steuerung

S7-1200 und S7-1500 liefern die Web API ab Werk mit einem **selbstsignierten Zertifikat** aus. Der Treiber prüft es (`tls.verify`, standardmäßig an - lassen Sie es an) und lehnt ein Zertifikat ab, das er nicht prüfen kann. Jeder Datensatz trägt dann 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 …
```

und das Log des Workers nennt den `SHA-256 fingerprint: 20:d6:…:54:a3` der Steuerung. Vergleichen Sie diesen Fingerabdruck mit der Steuerung und hinterlegen Sie ihn in der URL:

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

Ein Zertifikat, das Ihre eigene CA signiert hat, braucht keinen hinterlegten Fingerabdruck, sofern es die Adresse aus der URL nennt: Nennen Sie stattdessen auf jedem Worker einen Truststore (`tls.trust-store=/etc/kafka-connect/plc-ca.p12&tls.trust-store-password=…`). Lassen Sie `tls.version` weg: Der Treiber bevorzugt dann TLS 1.3 und weicht auf TLS 1.2 aus, das ältere Steuerungen brauchen.

Zugangsdaten

Zugangsdaten stehen nie in der URL oder in der Datei selbst. Jedes geheime Feld ist ein **Verweis** `{{name}}`, und die Connector-Eigenschaft `secrets.<name>` liefert den Wert - am besten über einen **Config Provider** von Kafka, damit der Wert auch nicht in der Connector-Konfiguration steht. Aktivieren Sie den Datei-Provider auf jedem Worker:

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

und verweisen Sie auf eine Properties-Datei, die nur der Benutzer des Workers lesen darf:

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

die Steuerung	`method`	anzugeben
Benutzerverwaltung mit einem Web-API-Konto	`username-password`	`username`, `password`

Die Web API authentifiziert jede Anfrage, eine Verbindung braucht also ihr `credential`; ohne lehnt die Steuerung die Anmeldung ab (`Login failed for user 'null'`). `password` muss ein Verweis sein; `username` darf einer sein oder Klartext. Im TIA-Projekt braucht das Konto die Berechtigung **Web API**, die nur die Anmeldung erlaubt, und getrennt davon die Berechtigungen, Werte zu lesen und zu schreiben. `$session:permissions` zeigt, was das Konto besitzt.

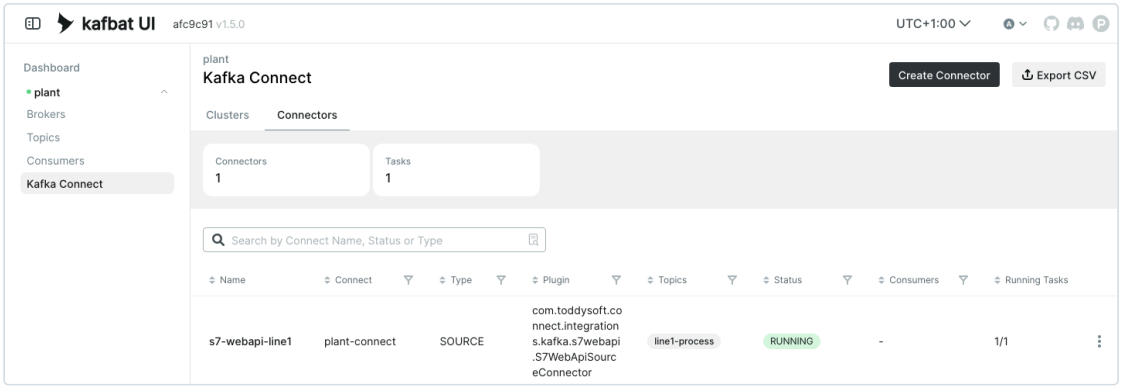
Kafka protokolliert die Konfigurationswerte dieser Connectoren nie, und der Worker schreibt den Platzhalter `${file:…}`, nicht den Wert, in sein Config-Topic. Beschränken Sie das Config-Topic trotzdem mit ACLs.

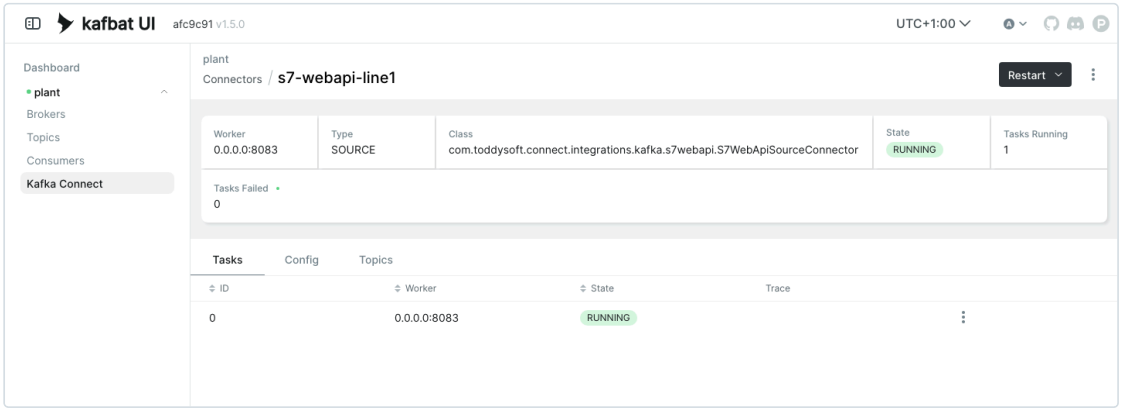
Der Source-Connector

```
{
  "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` 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 S7-1200, die über ihre Web API die Variablen `"TestData".g_b1`, `g_s16` und `g_s32` einmal pro Sekunde liest.

Datensätze

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

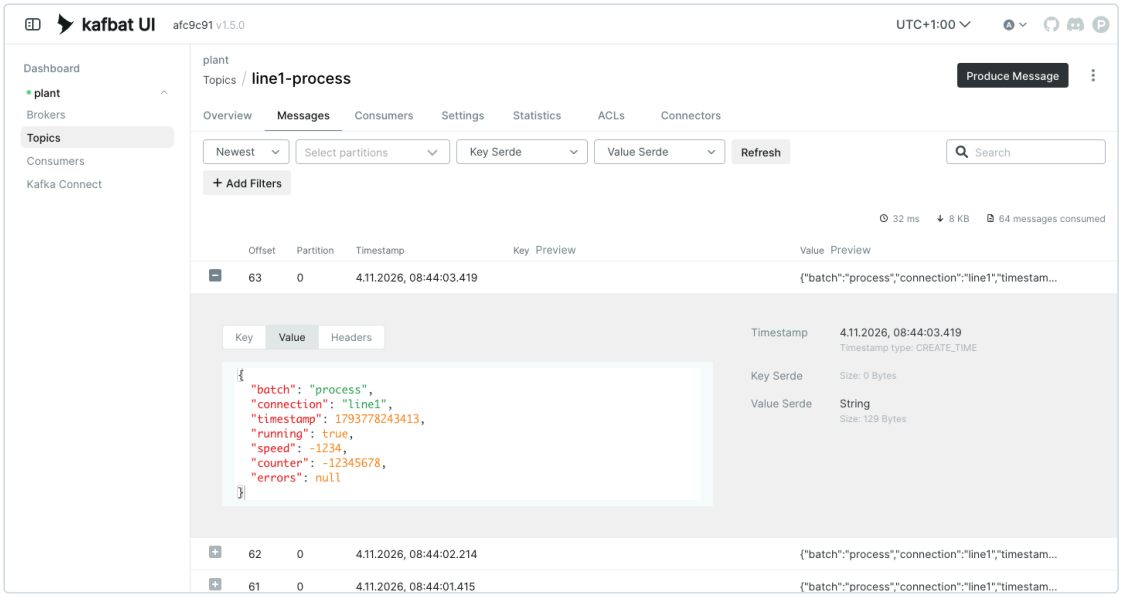
Feld	Typ	
`batch`	STRING	der Batch
`connection`	STRING	die Verbindung
`timestamp`	INT64	der Zeitpunkt des Lesens, Millisekunden seit der Epoche
ein Feld je Variablen-Alias	siehe Typtabelle	`null`, wenn die Variable nicht gelesen werden konnte
`errors`	MAP<STRING,STRING>	Alias zu Grund, für jede Variable, 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	BOOLEAN	
SINT	INT8	

Typ in der Steuerung	Connect-Typ	
BYTE, USINT, INT	INT16	vorzeichenlose 8-Bit-Typen erweitert
WORD, UINT, DINT	INT32	vorzeichenlose 16-Bit-Typen erweitert
DWORD, UDINT, LINT	INT64	vorzeichenlose 32-Bit-Typen erweitert
LWORD, ULINT	Decimal (Skala 0)	vorzeichenlos 64 Bit, exakt
REAL	FLOAT32	
LREAL	FLOAT64	
CHAR, WCHAR, STRING, WSTRING	STRING	
TIME, LTIME	INT64	Millisekunden
TIME_OF_DAY	INT64	Millisekunden seit Mitternacht
DATE	INT32	Tage seit der Epoche
eine Struktur, auch DTL und DATE_AND_TIME	STRING	JSON; die Web API überträgt sie nur Member für Member
ein Array, `$diag:buffer`, `$diag:alarms`	ARRAY of STRING	jedes Element als Text, eine Struktur als JSON
ein Member, für den die Web API keinen Typ nennt (Member einer Struktur, die sie nicht auflisten kann)	nach seinem Wert: eine Zahl INT32 oder INT64	wo es darauf ankommt, eine Typangabe ergänzen

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": "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}"
  }
}
```

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. Das Konto braucht die Berechtigung, Werte zu schreiben, und eine Variable, die das TIA-Projekt als schreibgeschützt markiert, wird unabhängig von den Berechtigungen des Kontos abgelehnt.

Ein Schreiben von `$control:operating-mode` (`run` / `stop`) stoppt oder startet die CPU. Die Steuerung stellt jede solche Anforderung in eine Warteschlange und führt jede aus, auch eine wiederholte; ihre Bestätigung bedeutet *eingereiht*, nicht *ausgeführt*. Eine Wiederholung ist also eine zweite Ausführung: Geben Sie einem Sink, der den Betriebszustand schreibt, `max.retries` 0, und prüfen Sie eine Änderung durch Lesen des Zustands, nicht an der Bestätigung.

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": "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 } }
  ]
}
```

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-s7-webapi` , `…-s7` , …). **Verwenden Sie die Connector-Klassen des jeweiligen Treibers** - hier `…kafka.s7webapi.S7WebApiSourceConnector` :

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.s7webapi.S7WebApiSourceConnector` / `S7WebApiSinkConnector` verwenden
der Connector wird mit Regelkennung und Stelle abgewiesen	die Konfiguration enthält einen Fehler	die Datei an jeder genannten Stelle korrigieren
… die Regelkennung ist `credential-reference-unresolved`	ein `{{name}}` ohne seine Eigenschaft `secrets.<name>`	die Eigenschaft im Connector ergänzen
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 Anmeldung, ein Zertifikat, eine Lizenz
`errors` nennt `License validation failed`	keine Lizenz gilt	siehe Die Lizenz prüfen
`errors` sagt `Server certificate not trusted for <host>:443`	das selbstsignierte Zertifikat der Steuerung	den Fingerabdruck aus dem Log des Workers hinterlegen (siehe Das Zertifikat der Steuerung)
`Login failed for user '…'`	falsches Passwort, oder dem Konto fehlt die Web-API-Berechtigung - die Steuerung antwortet auf beides gleich	Passwort und Berechtigungen des Kontos im TIA-Projekt prüfen (siehe Zugangsdaten)

Symptom	Ursache	was zu tun ist
`Login failed for user 'null'`	keine Zugangsdaten haben den Treiber erreicht: Die Verbindung hat kein `credential`	es ergänzen (siehe Zugangsdaten)
die Anmeldung gelingt, aber Variablen werden mit `ACCESS_DENIED` abgelehnt	dem Konto fehlt die Berechtigung, Werte zu lesen oder zu schreiben	sie im TIA-Projekt vergeben; `$session:permissions` zeigt die Berechtigungen des Kontos
`errors` nennt HTTP `404`	der Webserver läuft, aber die Web API ist nicht aktiviert	die Web API im TIA-Projekt aktivieren
`errors` nennt HTTP `403`	dem Konto fehlt die Web-API-Berechtigung, oder die Firmware hat keine Web API	Firmware (S7-1200 V4.5, S7-1500 V2.9) und Konto prüfen
eine Variable wird mit `NOT_FOUND` abgelehnt	keine solche Variable, oder sie ist nicht aus HMI/Web erreichbar	den Namen und das HMI/Web-Attribut der Variablen im TIA-Projekt prüfen
eine Variable wird mit `INVALID_ADDRESS` abgelehnt	meist ein Member-Name mit Leerzeichen oder Bindestrich ohne Anführungszeichen, oder Indizes als `[1][1]` geschrieben	den Member in Anführungszeichen setzen; `[1,1]` schreiben
das Schreiben eines Arrays oder einer Struktur wird mit `UNSUPPORTED` abgelehnt	die Web API schreibt ein Aggregat nie als Ganzes	seine Elemente oder ihre Member schreiben
der TLS-Handshake scheitert an einer älteren Steuerung	`tls.version` auf TLS 1.3 festgelegt	`tls.version` aus der URL entfernen
das Lesen ist langsam	die Web API ist je Variable etwa zehnmal langsamer als die S7-Protokolle	für zyklische Prozessdaten den S7-Connector verwenden

Symptom	Ursache	was zu tun ist
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.

- **Neu:** der S7-Web-API-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.