Kafka Producer Connector

Publish FrameworX tag values to Apache Kafka topics as JSON messages.

Available from FrameworX 10.1.6. Earlier versions do not include this connector.

- Name: **Kafka**
- Version 1.0.0.0
- Protocol: Kafka (producer)
- Interface: TCP/IP
- Runtime: Windows x64, .NET Framework 4.8 and .NET 10
- Available from: FrameworX 10.1.6
- Configuration:
  - Devices / Protocols

Overview
Prerequisites
Procedure
Channel Configuration
Node Configuration
Points Configuration
Message Format
Delivery Behavior
Station Examples
Platform Support
Troubleshooting
Third-Party Components
Driver Revision History
Related Pages

Overview

The Kafka connector sends FrameworX tag values to an Apache Kafka cluster. Each write to a point publishes one JSON message to a Kafka topic. The connector is a producer only. To read Kafka topics into tags, use the Kafka Consumer Connector. Neither connector creates tags from topics.

This page covers the channel, node and point settings, the message format and delivery behavior. For MQTT brokers, see MQTT Client Connector. For general channel, node and point concepts, see Devices Module.

Communication Driver Information	
Driver name	Kafka
Assembly Name	T.ProtocolDriver.Kafka
Assembly Version	1.0.0.0
Protocol description	Kafka Producer Connector
Available for Linux	False
Direction	Egress only (FrameworX to Kafka)
Tested broker	Apache Kafka 4.3.1

Prerequisites

- FrameworX 10.1.6 or later.
- A Windows x64 computer running the FrameworX runtime. See *Platform Support* below for Windows ARM64.
- The Microsoft Visual C++ 2015-2022 x64 runtime. The FrameworX installer installs it by default. If you clear this component in a Custom install and the computer does not have the runtime, the connector publishes no messages.
- An Apache Kafka broker reachable from the runtime computer on its listener port, for example `9092`.
- The target topics exist on the broker, or the broker creates topics on first use (`auto.create.topics.enable=true`).
- For SASL authentication: a Kafka user with write permission on the target topics.

Procedure

1. In **Devices Channels**, create a channel and select the protocol **Kafka Producer Connector**.
2. Open the channel **Protocol Options** and set the topic strategy, message key and producer settings. See *Channel Configuration*.
3. In **Devices Nodes**, create a node on the channel and set the **Station** fields: broker list, security protocol and credentials. See *Node Configuration*.
4. In **Devices Points**, add one point per tag to publish. Set **Address** to the Kafka topic and **AccessType** to `Write`.
5. Start the runtime. Each time a mapped tag changes, the connector publishes one message.

Channel Configuration

Protocol Options

Field order in the options string: `Type;TopicStrategy;DefaultTopic;KeyStrategy;PayloadFormat;Acks;LingerMs;MessageTimeoutMs;CompressionType;EnableIdempotence;PublishRateMs`

Option	Values	Default	Description
Type	`Producer`	`Producer`	Kafka client role. Read-only. This version supports the producer role only.
TopicStrategy	`PerPoint`, `Fixed`	`PerPoint`	`PerPoint` publishes each point to the topic in its Address. `Fixed` publishes every point of the channel to **DefaultTopic** and ignores the point Address.
DefaultTopic	Topic name	(blank)	Topic used with `Fixed`, and with `PerPoint` for points whose Address is blank.
KeyStrategy	`TagName`, `None`	`TagName`	`TagName` sets the message key to the tag name, so all messages of one tag go to the same partition. To keep their order when the client retries a send, check **EnableIdempotence**. `None` sends messages without a key, and Kafka spreads them across partitions.
PayloadFormat	`JSON`	`JSON`	Message payload format. Read-only. See *Message Format*.
Acks	`All`, `Leader`, `None`	`All`	Broker acknowledgement the producer waits for. `All` waits for all in-sync replicas. `Leader` waits for the partition leader only. `None` does not wait for an acknowledgement.
LingerMs	Integer, 0 or more	`5`	Time in milliseconds the producer waits to group messages into one batch. Higher values raise throughput and add latency. Set it lower than **MessageTimeoutMs**, or the producer does not start.
MessageTimeoutMs	Integer, 1000 or more	`30000`	Delivery timeout in milliseconds. The client reports a message as failed when the broker does not confirm it within this time. The connector raises values below 1000 to 1000.
CompressionType	`None`, `Gzip`, `Snappy`, `Lz4`, `Zstd`	`None`	Compression codec for message batches.
EnableIdempotence	Checked or cleared	Cleared (`false`)	Enables the Kafka idempotent producer, which prevents duplicate messages from producer retries. When checked, the connector uses `Acks=All` regardless of the **Acks** setting.
PublishRateMs	Integer, 50 or more	`500`	Interval in milliseconds between producer cycles. Each cycle passes the messages buffered since the previous cycle to the Kafka client. The connector raises values below 50 to 50.

Options string examples:

```
Default values:                Producer;PerPoint;;TagName;JSON;All;5;30000;None;false;500
Fixed topic "telemetry":       Producer;Fixed;telemetry;TagName;JSON;All;5;30000;None;false;500
High throughput (Lz4, leader): Producer;PerPoint;;TagName;JSON;Leader;20;30000;Lz4;false;250
```

Node Configuration

Station Configuration

Station syntax: **<BootstrapServers>;<SecurityProtocol>;<SaslMechanism>;[SaslUsername];[SaslPassword];[SslCaLocation]**

Field	Values	Default	Description
BootstrapServers	`host:port` list	`localhost:9092`	Comma-separated list of brokers, for example `broker1:9092,broker2:9092`. Required. The client discovers the rest of the cluster from these brokers.
SecurityProtocol	`Plaintext`, `Ssl`, `SaslPlaintext`, `SaslSsl`	`Plaintext`	Security of the broker connection. `Ssl` and `SaslSsl` use TLS. `SaslPlaintext` and `SaslSsl` use SASL authentication. Match the broker listener on the port in **BootstrapServers**.
SaslMechanism	`Plain`, `ScramSha256`, `ScramSha512`	`Plain`	SASL mechanism. Used only when **SecurityProtocol** is `SaslPlaintext` or `SaslSsl`.
SaslUsername	Text	(blank)	SASL user name. Used only with `SaslPlaintext` or `SaslSsl`.

SaslPassword	Password	(blank)	SASL password. The Designer stores it encoded. Used only with `SaslPlaintext` or `SaslSsl`.
SslCaLocation	File path	(blank)	Path to a PEM file with the CA certificate, or certificate chain, of the broker, for example `C:\Kafka\certs\ca.pem`. Used with `Ssl` and `SaslSsl`. When blank, the client uses the Trusted Root Certification Authorities store of Windows.

Enter **SaslPassword** in the Designer station editor. The station string separates fields with semicolons, and the Designer encoding keeps a password with a semicolon in one field. A plain-text password with a semicolon, or one starting with #, written to the station string by an import or a script, reaches the broker wrong.

Points Configuration

Address

The point Address is the Kafka **Topic** the connector publishes the tag to, for example `plant.line1.temperature`.

- With **TopicStrategy** `PerPoint`, a blank Address uses the channel **DefaultTopic**.
- With **TopicStrategy** `Fixed`, the connector ignores the Address and publishes every point to **DefaultTopic**.
- The connector skips a point with no topic from either source and logs the tag name.

AccessType

The connector publishes on write. Assign an AccessType with write enabled, for example the predefined `Write` type, which writes when the tag changes its value. Read polling has no effect on this connector. See Devices AccessTypes Reference.

Message Format

Each write produces one Kafka message with a JSON payload:

```
{"tag":"K_PlainCounter","value":265,"quality":192,"timestamp":"2026-10-08T10:11:00.6674091Z"}
```

Field	Content
`tag`	Tag name.
`value`	Tag value at the time of the write, in the matching JSON type, for example a number or a string. The value is `null` when the write carries no value.
`quality`	Tag quality as an integer. `192` is Good.
`timestamp`	Tag timestamp in ISO-8601 UTC, with the `Z` suffix. A tag without a valid timestamp uses the current UTC time.

- **Message key:** the tag name with **KeyStrategy** `TagName`, no key with `None`.
- **Kafka record timestamp:** the time of the `timestamp` field, in milliseconds. A topic with `message.timestamp.type=LogAppendTime` replaces it with the broker time.
- The connector captures the value when the write occurs, so each message carries the value of its own write.

Delivery Behavior

- **No store-and-forward.** The connector keeps messages in memory only and drops a message the broker does not confirm. The connector does not resend a message after the Kafka client reports it as failed. Without **EnableIdempotence**, a client retry after a lost broker acknowledgement writes the message twice.
- **Retries.** The Kafka client retries sends and reconnects to the brokers on its own. When a message has no confirmation after **MessageTimeoutMs**, the connector logs `Kafka delivery failed` and drops the message.
- **Buffer limit.** While the producer is unavailable, for example because of an invalid configuration, the connector buffers up to 100,000 messages per node. Above this limit it drops the oldest messages and logs the total dropped.
- **Configuration changes.** When the node Station changes, the connector rebuilds the producer with the new settings.
- **Runtime stop.** On stop, the connector gives the Kafka client up to 5 seconds per node to deliver the messages it holds. The connector discards the messages still in its own buffer, for example writes made since the last producer cycle.

Station Examples

1. Local Apache Kafka, no authentication, no TLS

Field	Value
BootstrapServers	`localhost:9092`
SecurityProtocol	`Plaintext`
SaslMechanism	`Plain` (not used)
SaslUsername	(blank)
SaslPassword	(blank)
SslCaLocation	(blank)

Station string:

```
localhost:9092;Plaintext;Plain;;;
```

Channel Protocol Options: the default values. Point: Address `plant.line1.temperature`, AccessType `Write`.

2. Apache Kafka with SASL_SSL, SCRAM-SHA-256 and a private CA file

Field	Value
BootstrapServers	`kafka01.plant.local:9093`
SecurityProtocol	`SaslSsl`
SaslMechanism	`ScramSha256`
SaslUsername	`fxproducer`
SaslPassword	The password of `fxproducer`, entered in the Designer station editor
SslCaLocation	`C:\Kafka\certs\ca.pem`

The broker listener on port `9093` uses SASL_SSL with `SCRAM-SHA-256`, and `ca.pem` holds the CA certificate in PEM format. The broker certificate lists `kafka01.plant.local` as a host name. Create the SCRAM user on the broker with `kafka-configs` before you start the runtime.

Platform Support

Platform	Supported
Windows x64, .NET Framework 4.8 runtime	Yes
Windows x64, .NET 10 runtime	Yes
Windows ARM64, .NET Framework 4.8 runtime	Yes, under x64 emulation
Windows ARM64, .NET 10 runtime started with the x64 `dotnet` host	Yes, under x64 emulation
Windows ARM64, .NET 10 runtime with the default ARM64 `dotnet` host	No
Linux, any architecture	No

The connector depends on the native Kafka client library for Windows x64. On a host where this library does not load, the channel publishes no messages.

Troubleshooting

The connector writes these messages to the Trace Window (see Runtime Diagnostics Reference). Check Devices in the Trace Window settings. Messages starting with `Kafka error` on are Debug messages and appear only when you also check Debug. For rows with a quoted message, the first column shows the start of the message.

Symptom	Likely Cause	Resolution
`Kafka: BootstrapServers cannot be empty`, the node does not start. After a Station edit: `Kafka: BootstrapServers is empty for node`	The first Station field is blank.	Enter at least one broker as `host:port`.
`Kafka: no topic resolved for tag`	The point Address is blank and the channel **DefaultTopic** is blank.	Set the point Address, or set **DefaultTopic** on the channel.
`Kafka: failed to build producer`	The client rejected the configuration, for example an **SslCaLocation** file missing or unreadable, or a **LingerMs** not lower than **MessageTimeoutMs**. Or the native Kafka client library did not load, see the Windows ARM64 and Visual C++ rows.	Check the reason at the end of the message. The connector retries on each producer cycle, so a corrected Station or CA file takes effect without a restart. After a Protocol Options change, restart the runtime.
`Kafka error on '<BootstrapServers>'`, repeated (Debug)	Broker unreachable, wrong port, TLS or SASL settings different from the broker listener, or wrong credentials. The client keeps retrying.	Confirm the runtime computer reaches the broker port. Match **SecurityProtocol** and **SaslMechanism** to the listener on the broker port. Check the user name and password.
`Kafka delivery failed: topic '<topic>'`	The broker did not confirm the message within **MessageTimeoutMs**, or it rejected the message. Common causes: broker unreachable, TLS or SASL settings different from the broker listener, wrong credentials, topic missing with automatic topic creation disabled, or no write permission on the topic.	Check Debug in the Trace Window settings to see the connection reason. Create the topic, or grant the Kafka user write permission on it. The connector does not resend messages reported as failed.
`Kafka produce rejected: topic '<topic>'`	The client send queue is full because tags change faster than the broker accepts messages, or one message is larger than the client message size limit.	Lower the change rate of the mapped tags, or set a **CompressionType**. Check the length of string tag values.
`Kafka fatal error on '<BootstrapServers>'`	The client reported an unrecoverable error, for example from the idempotent producer.	Read the reason in the message. Correct the cause, then restart the runtime.
`Kafka: node '<node>' pending queue exceeded 100000`	The producer is not available, so messages accumulate. The connector drops the oldest.	Fix the producer error logged before this message.
TLS handshake fails with `certificate verify failed` in a `Kafka error on` message (Debug), CA file correct	The broker certificate does not list the host name used in **BootstrapServers**. The client checks the host name of the broker certificate.	Use the host name from the broker certificate in **BootstrapServers**, or issue a broker certificate with this host name in its Subject Alternative Name.
TLS handshake fails with a broker certificate trusted by other clients	The TLS library rejects certificates with RSA keys shorter than 2048 bits. It also does not read a system `openssl.cnf` file or the `OPENSSL_CONF` variable.	Use a broker certificate with an RSA key of 2048 bits or more. Set the CA file in **SslCaLocation** instead of an OpenSSL configuration file.
`Kafka: failed to build producer` with a native library load error, on Windows ARM64 with the .NET 10 runtime	The default ARM64 `dotnet` host does not load the x64 Kafka client library.	Use the .NET Framework 4.8 runtime, or start the .NET 10 runtime with the x64 `dotnet` host.
`Kafka: failed to build producer` with a native library load error, on any runtime	The Visual C++ 2015-2022 x64 runtime is missing.	Install the Microsoft Visual C++ 2015-2022 Redistributable (x64).
No message for a point, no error	The point AccessType has write disabled, or the tag value did not change.	Assign the `Write` AccessType, or another type with write enabled.

Third-Party Components

The connector includes the following components, installed in the `Protocols` folder of the FrameworX installation:

Component	Version
Confluent.Kafka (.NET client)	2.15.1
librdkafka	2.15.1
OpenSSL	3.5.9 LTS

libcurl	8.23.0
zlib	1.3.2
Zstandard (zstd)	1.5.7

License notices for these components ship in `<FrameworX installation>\Protocols\Kafka-ThirdPartyNotices.txt`.

Driver Revision History

Kafka Revision History	
Version	**Notes**
1.0.0.0	Initial release in FrameworX 10.1.6. Producer only, JSON payload.

Related Pages

- Kafka Consumer Connector
- Devices Module
- Devices Module Reference
- Devices AccessTypes Reference
- Runtime Diagnostics Reference
- MQTT Client Connector
- IT-Cloud and AI Connectors

In this section...