Kafka Transport Adapter

(*Available starting from version 6.10.0 of FIXEdge C++)*

Overview

The Kafka Transport Adapter (hereinafter Kafka TA) provides the FIXEdge server connectivity to the Kafka distributed streaming platform.

Internally, its implementation uses the Kafka Consumer and Producer APIs. The broker API version must be 0.10.0.0 or higher.

The Kafka TA acts as a FIXEdge plugin and establishes a connection to a Kafka single instance or a Kafka cluster. The adapter supports TLS connectivity.

The Kafka TA transfers FIX messages in raw or serialized formats from the FIXEdge Server to the Kafka platform and vice versa.

Kafka TA performance is close to Kafka's native speed of processing. The average FIX messages reading/writing per second is 30,000 or higher. This number refers to the overall throughput capacity for the FIXEdge server with the Kafka TA (from FIX session to Kafka server).

The Kafka TA automatically reconnects to Kafka if the connection to a message broker was unexpectedly terminated. Reconnection behavior can be configured by parameters TransportLayer.KafkaTA.reconnect.backoff.ms and TransportLayer.KafkaTA.reconnect.backoff.max.ms.

Multiple instances of the Kafka TA can be configured using one library instance, inside one instance of FIXEdge.

During FIXEdge server termination the Kafka TA employs graceful termination, ensuring that message processing operations are executed correctly, completely, and without message loss. When the FIXEdge server fails and nongraceful termination occurs, messages will be retrieved from persistent message storage.

FIFO (First In First Out) ordering is guaranteed, meaning that the order in which messages were sent and received will be maintained upon the termination of the server instance.

Concept

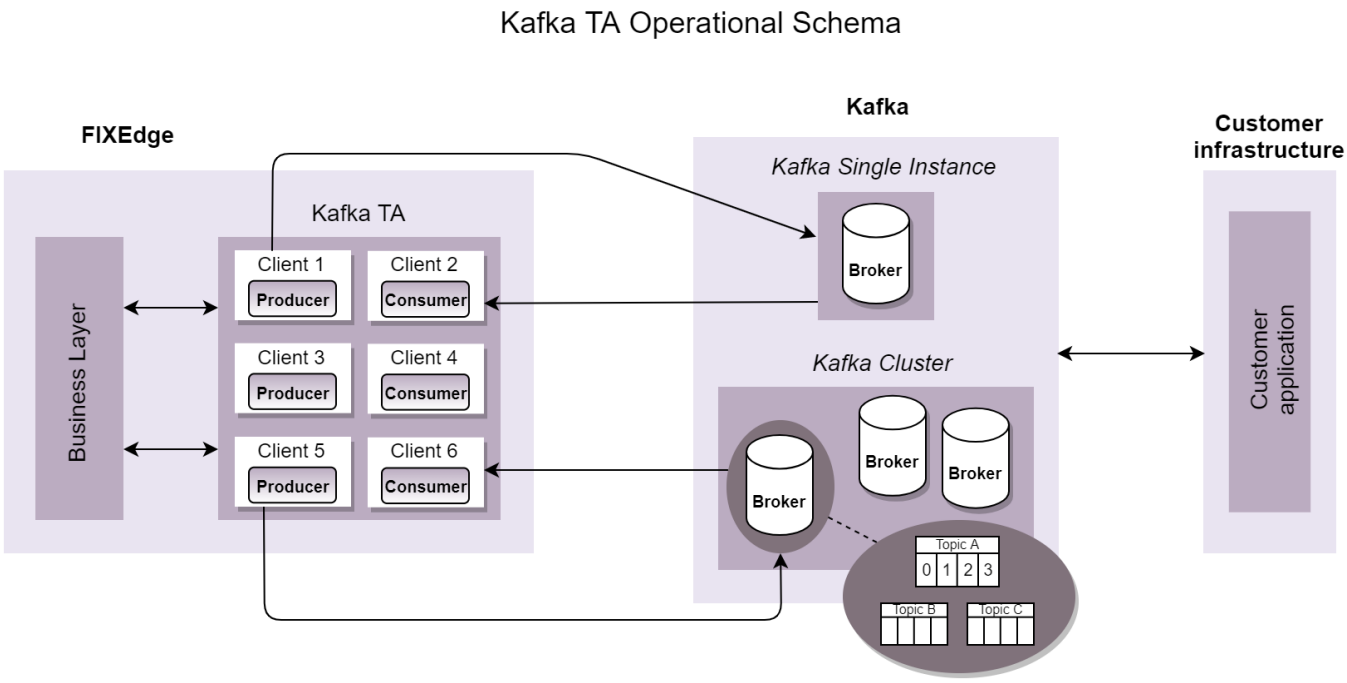
The Kafka TA is intended to communicate FIX messages to other applications using the Kafka streaming platform as middleware.

Communication is provided between the FIXEdge server and a Kafka broker or cluster on the FIXEdge side.

The Kafka TA allows integration with Kafka via Consumer/Producer API. FIXEdge clients, configured either as producers or consumers, push and pull messages to/from brokers housed in Kafka that organize and store messages according to topics. Brokers commit messages to disk storage for a customizable period of time or space, configurable according to the topic. Topics are further broken down into partitions, with each partition acting as a separate commit log. A broker administrator keeps track of consumers reading partitions by assigning offsets, thereby providing guaranteed-ordering. These features facilitate easy scaling and event replay, as well as the building of fault-tolerant systems.

The Kafka TA differentiates from other Transport Adapters in providing custom serialization/deserialization and durable message storage.

The schema below represents an interaction between FIXEdge and a customer's infrastructure via the Kafka solution.



Basic elements of the Kafka connectivity model

- Producer = publisher, the sender of messages
- Consumer = subscriber, the reader of messages
- Message/event = a single unit of data
- Broker = a server forming a storage layer containing one or more topics
- Topic = ordered collection of events that are organized and stored in a durable way
- Partition = buckets located within topics across which messages are distributed
- Commit log = a record of changes that is committed to disk and can be replayed by consumers
- Guaranteed ordering = a guarantee that a consumer of a given topic-partition will always read that partition's events in exactly the same order as they were written
- Offset = tracking feature that helps consumers remember its position when last reading a partition

Client behavior

When several clients (either one Producer or one Consumer, or both one Producer and one Consumer) are configured for the Kafka TA, all of them establish a connection to the Kafka platform when FIXEdge starts its work.
If one of the clients fails to establish a connection, the remaining clients are not affected by this failure.

Delivery

When the Kafka TA sends a message, it is first put into persistent storage, and then sent through the producer API to a Kafka broker. Once the Kafka platform sends a confirmation that the message was received, the Kafka TA marks the Kafka platform's acknowledgement that the message was delivered in the persistent storage. These two steps implement a delivery guarantee. If the message couldn't be sent it will remain stored in the persistent storage. Once the FIXEdge server starts and Kafka connection is reestablished, any unsent messages in the persistent message storage will be processed before other runtime messages. After delivery, messages remain stored in the persistent message storage.

Message ordering:

- FIX messages sent to Kafka via the Producer API are sent in the same order as the messages that FIXEdge sends to the client.
- FIXEdge fetches FIX messages via the Consumer API in the same order that messages arrive at a particular Kafka topic.
- The Kafka TA services discipline is FIFO (First In First Out) in the scope of one Kafka partition (for messages that come from one client).

The connectivity mechanism preserves FIX messages even if critical situations occur, such as:

- FIXEdge server is terminated
- Kafka TA crashes or does not start due to invalid configuration
- FIX Session is terminated non-gracefully
- message broker is down
- network disorder occurs

The Kafka TA provides an at least once delivery guarantee, (as opposed to only once). This means that messages are guaranteed to be delivered at least once, with the possibility of duplicates.

Persistent Message Storage

Kafka TA persistent FIX messages in text format are stored as a file saved to a disk using memory mapping.

Storage is unlimited but should not exceed disk capacity.

All messages passing through the system are kept in the persistent message storage and are not cleaned up automatically.

Committing to the Kafka platform

The **TransportLayer.KafkaTA.<Session>.Consumer.Commit** parameter provides three strategies that dictate how commit messages (acknowledgements of messages received) are sent to the Kafka platform.

The table below summarizes the best strategies for prioritizing speed and/or reliability in descending order.

Speed	Reliability
1) 'Auto'	1) 'Sync'
2) 'Async'	2) 'Async'
3) 'Sync'	3) 'Auto'

The three strategies are described in more detail in the table below.

TransportLayer. KafkaTA.<Session>. Consumer.Commit	Description

'Auto'	Commit messages are sent by the Kafka TA to the Kafka platform as configured, for example, at least once during a defined period of time. This parameter is the best option if there are no specific requirements as far as reliability. 'Auto' is also the: • Default value • Fastest and most efficient option because the number of requests is reduced • Least reliable method Since commit messages are not sent after every sent message, there is a possibility that duplicate messages will be sent by the Kafka TA.
'Sync'	For every message read by the Kafka TA's consumer, the Kafka TA sends a commit message is to the corresponding broker confirming the value of the last read offset. The Kafka TA does not read the next message without first sending the commit message. If there is a message sending failure, FIXEdge will notice it immediately. This is the most reliable option because Kafka is constantly updated on the consumer's progress. It is also the slowest method.
'Async'	The 'Async' option sends a commit message after each message, however, it does so in an asynchronous manner. Due to this, the Kafka TA does not block reading and delegates sending commit messages to an asynchronous thread. Unsuccessfully executed commits do not stop the process and are written into the log. This option is less reliable than the 'Sync' option but more reliable than the 'Auto' option. The number of lost messages might be smaller than the 'Auto' option. This option is faster than the 'Sync' option but slower than the 'Auto' option.

'Sync' and 'Async' are chosen when there is less traffic, more time for processing, and the user wants to increase the reliability of message sending (avoiding duplicates, etc.)

Message handling

The Kafka TA employs the following messaging handling methods:

- **Sync message handling**
  - Receiving messages in the Kafka TA from the Kafka platform is regarded as *sync* messaging.
  - Messages are not saved to any storage but become available at the FIXEdge Business Layer.
  - Messages arrive at the FIXEdge Business Layer in the same order as they had arrived to the Kafka topic they were fetched from.
- **Async message handling**
  - Sending messages from the Kafka TA to the Kafka platform is regarded as a*sync* messaging.
  - Messages are saved to persistent message storage.

Configuration steps

Given that FIXEdge has already been installed in the user's environment, use the following steps to step up the Kafka TA

1. Enable a transport adapter to be used by FIXEdge:
   In the *'Transport Layer Section'* of the *FIXEdge.properties*, add the Kafka TA to the list of supported adapters:

   ```
   TransportLayer.TransportAdapters = TransportLayer.KafkaTA
   ```

   Note: If you use other transport adapters, just add *TransportLayer.KafkaTA* to the end of the list:

   ```
   TransportLayer.TransportAdapters = ..., TransportLayer.KafkaTA
   ```

2. Configure the Kafka TA by adding the Kafka TA section to the *FIXEdge.properties*:

```
TransportLayer.KafkaTA.Description = Kafka Transport Adaptor
TransportLayer.KafkaTA.DllName = bin/KafkaTA-vc10-MD-x64.dll
TransportLayer.KafkaTA.Sessions = Kafka

TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9092
TransportLayer.KafkaTA.Kafka.FIXVersion = FIX44

TransportLayer.KafkaTA.Kafka.Consumer.Commit = Auto
TransportLayer.KafkaTA.Kafka.Consumer.Topics = outputTopic
TransportLayer.KafkaTA.Kafka.Consumer.group.id = ID

TransportLayer.KafkaTA.Kafka.Producer.Topic = topic
```

Note: Sample settings can be copied to the *FIXEdge.properties* file from the *KafkaTA.properties* file (located in the *doc* folder of the FIXEdge distribution package).

3. Configure rules for message routing from the Kafka TA.
   The Kafka TA Client is referred to the Business Layer (BL) by the ClientID name specified in the *FIXEdge.properties* file. For a Kafka Business Layer configuration sample, refer to the Configuration sample sub-section.

4. Restart the FIXEdge server to apply the changes.

Configuring multiple adapter instances

Multiple instances of the Kafka TA can be used during one library instance, inside one instance of FIXEdge.

To run multiple instances of the Kafka TA, each new instance must be assigned a new name and new session names. The name assigned to the new adapter instance must be consistent across parameters relating to that instance. New session names must be unique across all existing instances of the adapter. Topic names **do not** need to be unique.

```
TransportLayer.TransportAdapters = TransportLayer.Kafka1, TransportLayer.Kafka2

TransportLayer.Kafka1.Description = Kafka Transport Adaptor #1
TransportLayer.Kafka1.DllName = bin/KafkaTA-vc10-MD-x64.dll
TransportLayer.Kafka1.Sessions = Kafka_1

TransportLayer.Kafka1.Kafka_1.bootstrap.servers = localhost:9092
TransportLayer.Kafka1.Kafka_1.FIXVersion = FIX44

TransportLayer.Kafka1.Kafka_1.Consumer.Commit = Auto
TransportLayer.Kafka1.Kafka_1.Consumer.Topics = outputTopic
TransportLayer.Kafka1.Kafka_1.Consumer.group.id = ID
TransportLayer.Kafka1.Kafka_1.Producer.Topic = topic


TransportLayer.Kafka2.Description = Kafka Transport Adaptor #2
TransportLayer.Kafka2.DllName = bin/KafkaTA-vc10-MD-x64_2.dll
TransportLayer.Kafka2.Sessions = Kafka_2

TransportLayer.Kafka2.Kafka_2.bootstrap.servers = localhost:9093
TransportLayer.Kafka2.Kafka_2.FIXVersion = FIX44

TransportLayer.Kafka2.Kafka_2.Consumer.Commit = Auto
TransportLayer.Kafka2.Kafka_2.Consumer.Topics = outputTopic
TransportLayer.Kafka2.Kafka_2.Consumer.group.id = ID

TransportLayer.Kafka2.Kafka_2.Producer.Topic = topic
```

In the above example, there 2 separated instances of Kafka adapters: "Kafka1" and "Kafka2". Session names are unique across all Kafka adapters

SSL Configuration Sample

Add the following values to server.properties file in Kafka broker installation:

```
listeners=PLAINTEXT://:9092,SSL://:9093
ssl.keystore.location=D:/SSL/kafka01.keystore.jks
ssl.keystore.password=123456
ssl.key.password=123456
ssl.truststore.location=D:/SSL/kafka.truststore.jks
ssl.truststore.password=123456
ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1
```

Change the broker port to SSL, set security.protocol and ssl.ca.location properties for TA sessions

```
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9093
TransportLayer.KafkaTA.Kafka.security.protocol = SSL
TransportLayer.KafkaTA.Kafka.ssl.ca.location = D:/SSL/root.pem
```

Configuration parameters

Connection to the Kafka platform can be established when the Kafka TA is properly configured with the Kafka IP address.

Property name	Description	Required	Default value	Example
Common properties				
TransportLayer.KafkaTA.Description	Adapter name. Note: It is recommended that this parameter is not empty. This parameter is used in the logging of Kafka events.	N		Kafka Transport Adaptor
TransportLayer.KafkaTA.DllName	Contains path and name of the Kafka adapter DLL.	Y		bin /KafkaTA-vc10-MD-x64.dll
TransportLayer.KafkaTA.Sessions	A comma-separated list of session names. At least one session should be defined. The values from this parameter will be used in the BL description.	Y		
Kafka specific parapameters				
TransportLayer.KafkaTA.reconnect.backoff.ms	Delay after which the Kafka TA starts attempting to re-connect if the connection breaks.	N	100	
TransportLayer.KafkaTA.reconnect.backoff.max.ms	Max delay time after which the Kafka TA stops attempts to re-connect if the connection breaks.	N	10,000	
Session properties				
TransportLayer.KafkaTA.<Session>.FIXVersion	A version of the FIX protocol. Acceptable values: FIX40, FIX41, FIX42, FIX43, FIX44, FIX50, FIX50SP1, FIX50SP2.	N	FIX44	
TransportLayer.KafkaTA.<Session>.ConnectTime	Scheduled time to connect to a Client The value should be in the cron time string format Local time zones will be used	N	If a value is not specified the session is not using a schedule.	

TransportLayer.KafkaTA.<Session>.DisconnectTime	Scheduled time to disconnect from a Client The value should be in the cron time string format Local time zones will be used	N	If a value is not specified the session is not using a schedule.	
TransportLayer.KafkaTA.<Session>.Serializer	Serializer name. Acceptable values: • Raw • XmlWrapper • JSON Serializer tasks can be set from an external plugin. Format: *TestSerializer:KafkaSerializer,* where: • *TestSerializer* is a serializer Class ID inside the plugin • *KafkaSerializer* is a Plugin ID ID values are set by the Plugin developer.	N	Raw	
TransportLayer.KafkaTA.<Session>.bootstrap.servers	An initial list of brokers.	Y		
Secure connection properties				
TransportLayer.KafkaTA.Kafka.sasl.username	This parameter only applies and is required if using PLAIN authentication.	N		johndoe
TransportLayer.KafkaTA.Kafka.sasl.password	This parameter only applies and is required if using PLAIN authentication.	N		password1234
TransportLayer.KafkaTA.<Session>.ssl.key.location	File or directory path to SSL private key. This parameter only applies and is required if using SSL certificate authentication.	N		D:/SSL/kafka01.pem
TransportLayer.KafkaTA.<Session>.ssl.key.password	This parameter only applies and is required if using SSL certificate authentication.	N		
TransportLayer.KafkaTA.<Session>.ssl.ca.location	File or directory path to CA certificate(s) for verifying the broker's key. The parameter is required if the security protocol is SSL.	N		D:/SSL/root.pem
TransportLayer.KafkaTA.<Session>.sasl.mechanism	Valid values: • (empty) - in this case, neither PLAIN nor GSSAPI authentications are in use • PLAIN - to use PLAIN authentication • GSSAPI - to use GSSAPI authentication	N	If a value is not specified user authentication is not applied.	
TransportLayer.KafkaTA.<Session>.security.protocol	The protocol used to communicate with brokers. Valid values: • PLAINTEXT - using an unsecured connection • SSL - using a secured connection • SASL_PLAINTEXT - using authentication and an unsecured connection • SASL_SSL - using authentication and secured connection	N	PLAINTEXT	
TransportLayer.KafkaTA.Kafka.sasl.kerberos.service.name	Only applies and is required if using GSS_API authentication. In this case, use the Kerberos principal name that Kafka runs as.	N		
Consumer properties				

TransportLayer.KafkaTA.<Session>.Consumer.Commit	Commit mode. Acceptable values: • Auto - automatically, according to time interval expiration • Sync - synchronously, after each received message • Async - asynchronously, after each received message For more information see Sending commit messages to the Kafka platform.	N	Auto	
TransportLayer.KafkaTA.<Session>.Consumer.group.id	A unique string that identifies the consumer group that the given Consumer belongs to. This property is required if the Consumer uses either the group management functionality by using subscribe (Topic) or the Kafka-based offset management strategy. When the "group.id" parameter is not configured and its value is yet to be assigned, the session ID is used as the value. If the group.id value is identical across multiple sessions, and topic names are the same across multiple sessions, only one of the sessions is going to receive and process the data. This scenario is permissible when processing messages in parallel.	N	The default value is equal to the <Session> ID.	
TransportLayer.KafkaTA.<Session>.Consumer.Topics	If a value is not specified, the Consumer is not in use. To use the Consumer, a value must be specified. In this case, use a comma separated list of topics that should be listened by the Consumer.	N		outputTopic
Producer properties				
TransportLayer.KafkaTA.<Session>.Producer.KeyTag	If a value is not specified, then a key tag will not be used to fill a key value. To use this parameter, specify a tag number in a FIX message. The value of the tag will be used as a key when the message is sent to Kafka.	N		
TransportLayer.KafkaTA.<Session>.Producer.Topic	If a value is not specified, the Producer is not in use. To use the producer, specify the topic that the Producer should send FIX messages to.	N		
TransportLayer.KafkaTA.<Session>.Producer.RejectMessageWhileNoConnection	Provides the option to reject all further outgoing messages from FIXEdge from being sent by alerting the Producer when messages cannot be delivered to Kafka	N	false	
TransportLayer.KafkaTA.<Session>.Producer.DisconnectionPeriodThreshold	The timeout period after which unsuccessful attempts at sending will be rejected if TransportLayer.KafkaTA.<Session>.Producer.RejectMessageWhileNoConnection=true. • This parameter is useful when a disconnection happens for a short period of time. If connection is restored during the specified threshold, (for example, 5 seconds), message processing doesn't stop during the connection gap and messages continue to be sent to the Kafka platform broker. • Alternatively, if the disconnection lasts for longer than the specified threshold, the connection is considered broken, the sender is informed that messages cannot be delivered to the Kafka platform, and reject messages are sent back to the sender.	N	0	

Configuration sample

The samples below represent the Kafka TA's configuration with the minimal set of parameters required to run the adaptor:

Kafka TA configuration file:

KafkaTA.properties

```
#------------------------------------------------------------
# Transport Layer Section
#------------------------------------------------------------

#Comma separated list of identifiers of Transport Adapters should be loaded.
TransportLayer.TransportAdapters = TransportLayer.KafkaTA

#------------------------------------------------------------
# The Kafka Transport Adaptor (KafkaTA) configuration file.
#------------------------------------------------------------

# Adaptor's name. Property is required
TransportLayer.KafkaTA.Description = Kafka Transport Adaptor

# Contains path and name of the Kafka adaptor dll. Property is required
TransportLayer.KafkaTA.DllName = bin/KafkaTA-vc10-MD-x64.dll

# List of adaptor sessions
TransportLayer.KafkaTA.Sessions = Kafka

# Following are parameters for each session
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9092
TransportLayer.KafkaTA.Kafka.FIXVersion = FIX44

TransportLayer.KafkaTA.Kafka.Consumer.Commit = Auto
TransportLayer.KafkaTA.Kafka.Consumer.Topics = outputTopic
TransportLayer.KafkaTA.Kafka.Consumer.group.id = ID

TransportLayer.KafkaTA.Kafka.Producer.Topic = topic
```

Kafka Business Layer rules:

BL_Config

```
<FIXEdge>
    <BusinessLayer>

       <Rule>
            <Source>
                <FixSession SenderCompID="SC" TargetCompID="FE"/>
            </Source>
            <Action>
                <Send><Client Name="Kafka"/></Send>
            </Action>
        </Rule>
       <Rule>
            <Source>
                <Client Name="Kafka"/>
            </Source>
            <Action>
                <Send>
                    <FixSession SenderCompID="FE" TargetCompID="SC"/>
                </Send>
            </Action>
        </Rule>

        <DefaultRule>
            <Action>
                <DoNothing/>
            </Action>
        </DefaultRule>

    </BusinessLayer>
</FIXEdge>
```

Authentication Configuration

SSL certificate authentication

To configure SSL authentication, follow these steps:

1. Make sure the Kafka broker and adaptor are configured for SSL connection
2. Set client authentication as "required" in the **server.properties** file

Example

```
listeners=PLAINTEXT://:9092,SSL://:9093
ssl.keystore.location=D:/SSL/kafka01.keystore.jks
ssl.keystore.password=123456
ssl.key.password=123456
ssl.truststore.location=D:/SSL/kafka.truststore.jks
ssl.truststore.password=123456
ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1
ssl.client.auth = required
```

3. Provide ssl.key details in the **FIXEdge.properties** file

Example

```
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9093
TransportLayer.KafkaTA.Kafka.security.protocol = SSL
TransportLayer.KafkaTA.Kafka.ssl.ca.location = D:/SSL/root.pem
TransportLayer.KafkaTA.Kafka.ssl.key.location = D:/SSL/kafka01.pem
TransportLayer.KafkaTA.Kafka.ssl.key.password = 123456
```

SASL_PLAIN authentication

To configure SASL_PLAIN authentication:

1. Use the following files and corresponding settings to configure the Kafka broker

server.properties file

```
listeners=PLAINTEXT://:9092,SASL_PLAINTEXT://:9093
advertised.listeners=PLAINTEXT://:9092,SASL_PLAINTEXT://:9093
sasl.enabled.mechanisms=PLAIN
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
allow.everyone.if.no.acl.found=true

sasl.jaas.config= \
org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="admin" \
    password="admin-secret" \
    user_admin="admin-secret";
```

2. Use the following file and corresponding settings in to configure the Kafka TA:

FixEdge.properties file

```
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9092
TransportLayer.KafkaTA.Kafka.security.protocol = SASL_PLAINTEXT
TransportLayer.KafkaTA.Kafka.sasl.mechanism=PLAIN
TransportLayer.KafkaTA.Kafka.sasl.username=admin
TransportLayer.KafkaTA.Kafka.sasl.password=admin-secret
```

SASL_SSL authentication

This is a combination of an SSL connection with client authentication and SASL_PLAIN authentication.

To configure SASL_SSL authentication:

1. Use the following files and corresponding settings to configure the Kafka broker

server.properties file

```
listeners=PLAINTEXT://:9092,SASL_SSL://:9093
advertised.listeners=PLAINTEXT://:9092,SASL_SSL://:9093
ssl.keystore.location=D:/SSL/kafka01.keystore.jks
ssl.keystore.password=123456
ssl.key.password=123456
ssl.truststore.location=D:/SSL/kafka.truststore.jks
ssl.truststore.password=123456
ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1
ssl.client.auth = required
sasl.enabled.mechanisms=PLAIN
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
allow.everyone.if.no.acl.found=true

sasl.jaas.config= \
org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="admin" \
    password="admin-secret" \
    user_admin="admin-secret";
```

2. Use the following file and corresponding settings to configure the Kafka TA:

FixEdge.properties file

```
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9093
TransportLayer.KafkaTA.Kafka.security.protocol = SSL
TransportLayer.KafkaTA.Kafka.ssl.ca.location = D:/SSL/root.pem
TransportLayer.KafkaTA.Kafka.ssl.key.location = D:/SSL/kafka01.pem
TransportLayer.KafkaTA.Kafka.ssl.key.password = 123456
TransportLayer.KafkaTA.Kafka.sasl.mechanism=PLAIN
TransportLayer.KafkaTA.Kafka.sasl.username=admin
TransportLayer.KafkaTA.Kafka.sasl.password=admin-secret
```

SASL_GSSAPI authentication

To configure SASL_GSSAPI authentication:

1. Use the following files and corresponding settings to configure the Kafka broker

server.properties file

```
listeners=PLAINTEXT://:9092,SASL_PLAINTEXT://:9093
advertised.listeners=PLAINTEXT://:9092,SASL_PLAINTEXT://:9093
sasl.enabled.mechanisms=GSSAPI
sasl.mechanism.inter.broker.protocol=GSSAPI
sasl.kerberos.service.name=kafka

listener.name.sasl_plaintext.gssapi.sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule
required \
  useKeyTab=true \
  storeKey=true \
  keyTab="D:/SSL/kafka_server.keytab" \
  principal="kafka/kafka_server@example.com";
```

2. Use the following file and corresponding settings to configure the Kafka TA

FixEdge.properties file

```
TransportLayer.KafkaTA.Kafka.bootstrap.servers = localhost:9092
TransportLayer.KafkaTA.Kafka.security.protocol = SASL_GSSAPI
TransportLayer.KafkaTA.Kafka.sasl.mechanism=GSSAPI
TransportLayer.KafkaTA.Kafka.sasl.kerberos.service.name=kafka
```

Logging

Kafka exhaustive logging means that any action with a configuration parameter is logged.

List of logged actions:

- Adapter initialization
- Parsing validation errors
- Configuration parameters
- Sent and received messages

The following levels of logging are used for Kafka:

- ERROR - is always ON
- INFO - can be optionally ON
- WARN - can be optionally ON
- DEBUG - is turned off by default and is used in critical cases

Any configuration event is logged in the "Kafka_TA" log category in the log file via the INFO messages.

The table below specifies the most common logging messages for Kafka.

Log type	Event description	Logging message format
ERROR	Client error	*"Session '<session_name>': [Consumer/Producer]: <error_message>"*
	Outgoing message failed to be queued	*"Session '<session_name>': Producer: Error handling the outgoing message: <native error>. Message: <message>"*
	Incoming message failed to be handled	*"Session '<session_name>': Consumer: Error handling the incoming message: <native error>. Message (optional): <message>"*
INFO	Client creation / any config parameter change	
WARN	Kafka TA fails to apply any config parameter	*"Session '<session_name>': <warning_message>"*
	The outgoing message is rejected because the client is not connected to Kafka	*"Session '<session_name>': Producer: The outgoing message is rejected: <message>" is visible in the "Kafka_TA"*
DEBUG	Creation of a Consumer or Producer	*"Session '<session_name>': Producer: Error handling the outgoing message: <native error>. Message: <message>"*
	Adding an outgoing message to the queue	*"Session '<session_name>': Producer: The message is put to the outgoing queue: <message>"*
	Removing an outgoing message from the queue	*"Session '<session_name>': Producer: Sending the message via the producer: <message>"*
	An incoming message is being received	*"Session '<session_name>': Consumer: Receiving the message: <message>"*

Logging setup

The Kafka TA writes messages to the log under the 'Kafka_TA' log category.

By default, the Kafka TA puts log messages into the FIXEdge log. Execute the following steps to configure the Kafka TA logging into a separate file:

1. Open the FIXEdge properties file.
2. Add 'Log.Kafka_TA.File.Name' property with value - path and name of the Kafka TA log file.

```
Log.Kafka_TA.File.Name = Kafka.log
```

3. Restart the FIXEdge server.

When the Kafka TA starts, the following record will be written into the log file (xxx is the Kafka TA version):

```
[NOTE] 20070215-11:20:12.437 - Kafka Adaptor v.xxx started.
```

Additional information about logging configuration can be found there:

- FIXEdge logs format
- How to redirect FIX Antenna and/or FIXEdge logging to Syslog instead of files
- How to divide different categories and severities of log files into different files in the Logging section

Scheduling

(*Available starting from version 6.11.0 of FIXEdge C++)*

The schedule settings regulate work of Kafka TA Producer/Consumer. The schedule can be configured as a cron expression or directly tied up to FIXEdge.

If the cron expression is used for scheduling, the Producer/Consumer starts using Client's connection to the Kafka platform at a scheduled time. At a scheduled disconnection time, the Producer/Consumer stops using the Client's connection to Kafka.

If the schedule settings are directly tied up to FIXEdge, the Producer/Consumer starts using Client's connection to the Kafka platform at the moment when the FIXEdge server starts. When the FIXEdge server stops, the Producer/Consumer stops using the Client's connection to Kafka.

Scheduling is represented by the following properties:

- TransportLayer.KafkaTA.<Session>.ConnectTime
- TransportLayer.KafkaTA.<Session>.DisconnectTime

Refer to the table in the Configuration parameters section above.

Custom serialization

When FIX messages are transferred in raw format (i.e. serialization is not applied), the Kafka TA implies a string containing the tag values has been converted to strings.

The serialized format of communication is fully supported by the Kafka TA. Custom serialization implies the specially designed external code and includes serialization of all Kafka message parts:

- Key
- Value
- Header(-s)

Serialization on sending

When custom serialization is applied and FIXEdge *sends* the message to the particular Kafka topic, this message is serialized via the custom serializer and is sent to a particular Kafka topic via the Producer API. The following DEBUG message is logged in the client log category of the log file:

"The message content:
headers: <headers>,
key: <key>,
partition Id: <partition Id>,
value: <value>"

Serialization on receiving

When custom serialization is applied and the new message *arrives* at the particular Kafka topic, this message is received by FIXEdge via the Consumer API. The following DEBUG message is logged in the client log category of the log file:

"The message content:
headers: <headers>,
key: <key>,
partition Id: <partition Id>,
value: <value>"

The message is deserialized via the custom serializer.

Message key processing

In the case of the *default* serialization, the Kafka TA produces a key using the value of the tag configured in the "KeyTag" parameter. The specified key is passed to the Kafka platform along with the message sent to the particular Kafka topic.

In the case of the *custom* serialization, the custom plugin produces the key that is passed to the Kafka platform along with the message sent to the particular Kafka topic.

Custom partitioning

If custom partitioning is configured, the custom plugin produces the partition ID based on the tag values of a FIX message.

Message Content Wrapping

When FIXEdge sends a message to a particular Kafka topic and the BL Client is configured to work with XMLData (213), two flows are possible:

1. The XMLData (213) field exists in the FIX message and is not empty.
   In this case, the XML message extracted from the XMLData (213) field is sent to the particular Kafka topic via the Producer API.
2. The XMLData (213) field does not exist in the FIX message or is empty.
   In this case, the corresponding error is sent to the Business Layer and nothing is sent to a Kafka topic via the Producer API.

When Kafka sends an XML message to FIXEdge and the BL Client is configured to work with XMLData (213), the FIX message with 35=n (XML message, JSON plain text, etc.) is received by FIXEdge, the XMLData (213) field is filled in with the received XML message, and the XMLDataLen (212) field is filled in with the received message length.

Kafka Adapter Monitoring

Kafka adapter is integrated with FIXICC monitoring feature.

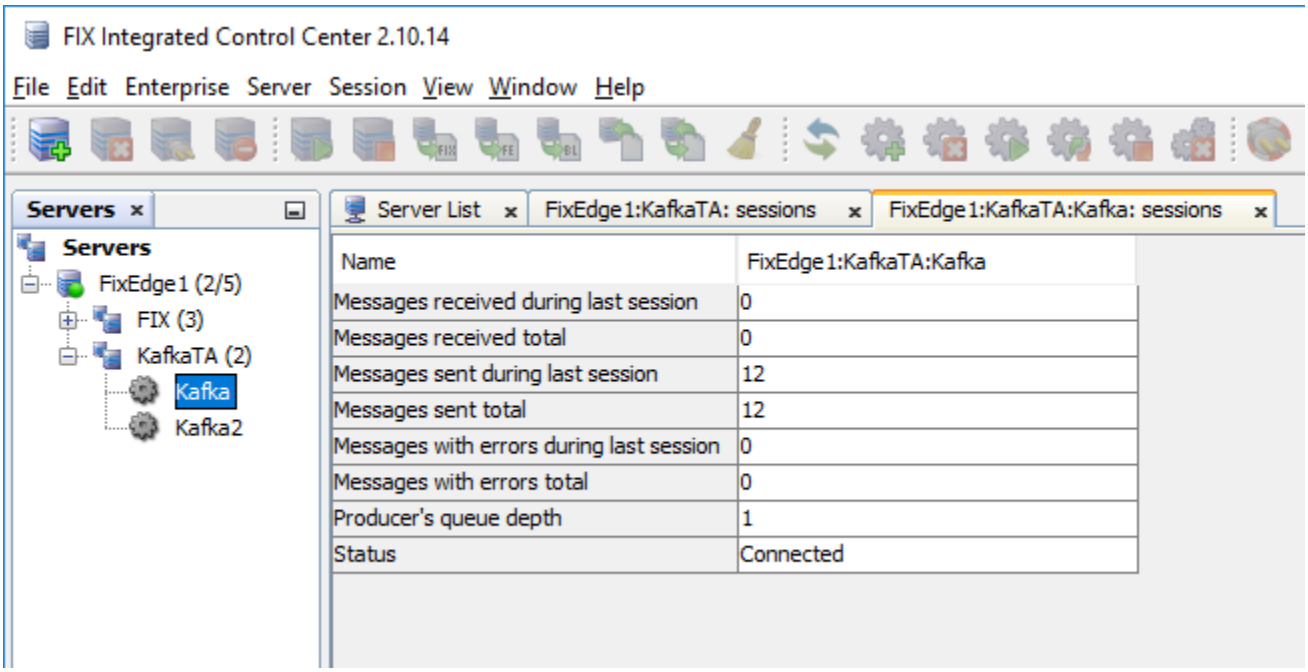
> Information about the adapter's state is sent to FIXICC as a response to the Transport Adapters state request via the FIX Administrative session.

The adapter provides information about the following parameters:

Name shown	Description
Messages received during last session	Number of incoming messages from Kafka handled per client, beginning from the client's current working session
Messages received total	Number of incoming messages from Kafka handled per client, beginning when FIXEdge started
Messages sent during last session	Number of outgoing messages to Kafka handled per client, beginning from the client's current working session
Messages sent total	Number of outgoing messages to Kafka handled per client, beginning when FIXEdge started
Messages with errors during last session	Number of errors related to the Kafka TA, beginning from the client's current working session
Messages with errors total	Number of errors related to the Kafka TA, beginning when FIXEdge started
Producer's queue depth	The size of the queue per producer
Status	Status of the client's connection with Kafka

> The lifetime of the session is defined by parameters: TransportLayer.KafkaTA.<Session>.ConnectTime and TransportLayer.KafkaTA.<Session>.DisconnectTime

The monitoring feature looks like this:



Troubleshooting

Common troubleshooting issues include:

Failure to initialize

If the adapter fails to start up, there is an issue with a configuration parameter.

To resolve this issue, the configuration parameter specified in the error message must be corrected. Check the log for error messages.

Example

In this example, the path specified to the adapter dll is the incorrect one:

```
2020-11-25 15:51:39,019 UTC INFO [FixLayer_Version] 20936 Module 'FixLayer' version 0.2.1.5 was loaded.
2020-11-25 15:51:39,019 UTC INFO [Engine] 20936 Custom AdminApplication is registered.
2020-11-25 15:51:39,024 UTC ERROR [TransportLayer] 20936 Transport Adaptor 'KafkaTA' has failed to initialize:
Error loading DLL 'd:\FIXEdge\bin\KafkaTA-vc10-MD-x64.dll'.
. The specified module could not be found. (Error code = 126)
2020-11-25 15:51:39,024 UTC INFO [TransportLayer] 20936 Module 'TransportLayer' version 0.1.1.5 was loaded.
```

The session wasn't created

If, during start-up, one session doesn't load successfully while others are created successfully, there is an issue with a configuration parameter.

To resolve this issue, the configuration parameter specified in the error message must be corrected. Check the log for error messages.

Example

In this example, two sessions ('Kafka' and 'Kafka2') are configured, and one was not created successfully.

In the first session, the wrong parameter, 'protocol', was used instead of the correct parameter, 'security.protocol', and the session was not created.

In the log file an ERROR message saying, "Failed to set protocol..." appears instead of an INFO message saying, "Session 'Kafka': was created successfully".

```
2020-11-25 16:03:00,882 UTC INFO [Kafka_TA] 7504 process logon for session 'Kafka'
2020-11-25 16:03:00,882 UTC INFO [Kafka_TA] 7504 Session 'Kafka': Is about to be created with parameters:

                    Consumer.Topics = outputTopic
                    Consumer.group.id = ID
                    FIXVersion = FIX44
                    Producer.Topic = topic
                    Serializer = Raw
                    bootstrap.servers = localhost:9092
                    protocol = SSL

2020-11-25 16:03:00,894 UTC ERROR [Kafka_TA] 7504 Failed to set protocol: No such configuration property: "protocol"
2020-11-25 16:03:00,894 UTC INFO [Kafka_TA] 7504 process logon for session 'Kafka2'
2020-11-25 16:03:00,895 UTC INFO [Kafka_TA] 7504 Session 'Kafka2': Is about to be created with parameters:

                    Consumer.Topics = topic
                    Consumer.group.id = ID
                    FIXVersion = FIX44
                    Producer.Topic = outputTopic
                    Serializer = Raw
                    bootstrap.servers = localhost:9092

2020-11-25 16:03:00,922 UTC INFO [CC_Layer] 7504 Client Kafka2 has logged in

2020-11-25 16:03:00,936 UTC INFO [Kafka_TA] 7504 Session 'Kafka2': was created successfully
2020-11-25 16:03:00,936 UTC INFO [Kafka_TA] 7504 Kafka Adaptor v.0.1 started.
```

Adapter not sending or receiving messages

If sessions have been created successfully, but the adapter isn't sending or receiving messages to/from the Kafka server, this issue has most likely occurred due to a problem with the connection.

If the adapter can't connect to the Kafka server, it will continue to make connection attempts in given intervals until a reason for the error is established. Until this is done, the TA will not be able to send or receive messages. The default level of logging doesn't explain the reason, you need to enable the deeper level of logging.

To establish what the error is, you must enable the **DEBUG** logging as follows:

1. Open the **FIXEdge.properties** file
2. Set the parameter **Log.DebugIsOn** = True

Once this is done, the log will show error messages stating previous connection attempts and the reason for the error. To resolve this issue, correct the configuration issue specified in the error message.

Example

In this example, the wrong server port, 9093 (for SSL connection), is configured instead of the correct one, 9092 (for PLAINTEXT connection).

With **DEBUG** logging enabled, we can see that the adapter is permanently trying to connect to the server, as well as an error message specifying the configuration issue.

```
2020-11-25 16:22:54,533 UTC   DEBUG   [Kafka_TA]  5808  Session 'Kafka': Producer: librdkafka ERROR: [thrd:
app]: rdkafka#producer-2: localhost:9093/bootstrap: Disconnected while requesting ApiVersion: might be caused
by incorrect security.protocol configuration (connecting to a SSL listener?) or broker version is < 0.10 (see
api.version.request) (after 197ms in state APIVERSION_QUERY)
2020-11-25 16:22:54,534 UTC   DEBUG   [Kafka_TA]  10912  Session 'Kafka': Consumer: librdkafka ERROR: [thrd:
localhost:9093/bootstrap]: 1/1 brokers are down
2020-11-25 16:22:54,534 UTC   DEBUG   [Kafka_TA]  22440  Session 'Kafka': Consumer: librdkafka ERROR: [thrd:
app]: rdkafka#consumer-1: localhost:9093/bootstrap: Disconnected while requesting ApiVersion: might be caused
by incorrect security.protocol configuration (connecting to a SSL listener?) or broker version is < 0.10 (see
api.version.request) (after 1086ms in state APIVERSION_QUERY)
2020-11-25 16:22:54,761 UTC   DEBUG   [Kafka_TA]  10912  Session 'Kafka': Consumer: librdkafka ERROR: [thrd:
localhost:9093/bootstrap]: 1/1 brokers are down
2020-11-25 16:22:55,033 UTC   DEBUG   [Kafka_TA]  24160  Session 'Kafka': Producer: librdkafka ERROR: [thrd:
localhost:9093/bootstrap]: 1/1 brokers are down
2020-11-25 16:23:07,261 UTC   DEBUG   [Kafka_TA]  22440  Session 'Kafka': Consumer: librdkafka ERROR: [thrd:
app]: rdkafka#consumer-1: localhost:9093/bootstrap: Disconnected while requesting ApiVersion: might be caused
by incorrect security.protocol configuration (connecting to a SSL listener?) or broker version is < 0.10 (see
api.version.request) (after 1ms in state APIVERSION_QUERY, 4 identical error(s) suppressed)
2020-11-25 16:23:07,261 UTC   DEBUG   [Kafka_TA]  10912  Session 'Kafka': Consumer: librdkafka ERROR: [thrd:
localhost:9093/bootstrap]: 1/1 brokers are down
2020-11-25 16:23:15,441 UTC   DEBUG   [Kafka_TA]  24160  Session 'Kafka': Producer: librdkafka ERROR: [thrd:
localhost:9093/bootstrap]: 1/1 brokers are down
2020-11-25 16:23:15,442 UTC   DEBUG   [Kafka_TA]  5808  Session 'Kafka': Producer: librdkafka ERROR: [thrd:
app]: rdkafka#producer-2: localhost:9093/bootstrap: Disconnected while requesting ApiVersion: might be caused
by incorrect security.protocol configuration (connecting to a SSL listener?) or broker version is < 0.10 (see
api.version.request) (after 1ms in state APIVERSION_QUERY, 4 identical error(s) suppressed)
```

Messages not delivered

If the adapter is receiving messages from the Kafka server, but not delivering them to target sessions, Business Layer rules have either been improperly configured or have not been specified.

In order to see an error message specifying the issue, **DEBUG** logging must be enabled.

To enable **DEBUG** logging:

1. Open the **FIXEdge.properties** file
2. Set the parameter **Log.DebugIsOn** = True

Example

With **DEBUG** logging enabled, we can see that message was received, but the message processing failed because no targets were configured in the BL. Business Layer rules for the 'Kafka2' session have not been specified.

```
2020-11-25 16:35:36,545 UTC DEBUG [Kafka_TA] 21964 Session 'Kafka2': Consumer: Receiving the message: 8=FIX.4.49
=15235=D49=SC56=FE34=252=20201125-16:35:35.936212=4213=test11=BTC/USD_LimitB_GTC55=BTC/USD54=160=20160401-15:15:
58.08038=0.1540=244=630.159=110=078
2020-11-25 16:35:36,545 UTC DEBUG [BL_RoutingTable] 21964 No BL rules found for message with ClientID 'Kafka2',
executing DefaultRule.
2020-11-25 16:35:36,546 UTC DEBUG [CC_Layer] 21964 BL has processed a message. Number of client IDs for
delivery :0. Number or FIX sessions for delivery :0.. Number or sources identifiers for delivery :0.

2020-11-25 16:35:36,546 UTC DEBUG [Kafka_TA] 21964 Session 'Kafka2': Consumer: Message processing failed
```