

# ps-plugin-KafkaTools

KafkaTools provides the following functionality that allows publishing to and consuming from topics on Kafka Servers.

## Smart Services

- Publish To Kafka
- Consume From Kafka
- Consume From Kafka - JWT Bearer Authorization Grant

## Performance and Scalability Considerations

Because of the nature of Kafka which is designed to handle large volumes of messages, Appian strongly recommends to consider the performance and scalability impacts of consuming large volumes of messages from Kafka into Appian.

To implement this plugin successfully in an application, you must:

1. Use the Transaction Manager to throttle the processing of the messages
2. Only consume messages that are relevant to your use cases. This can be done by creating dedicated Kafka topics for Appian and implementing message filters
3. Conduct performance testing with a production-like volume of messages and with the final process models and rules designed to handle the messages
4. Tune the configuration of the polling intervals, sleeping intervals, number of consumers and the Transaction Manager settings to not overload Appian and to leave resources available for end-user activities

## Installation Instructions

1. Install [Transaction Manager](#)
  - i. Follow instructions to create new job\_type(s) and process model(s) to handle kafaka topics, could be one or multiple depending on requirements
2. For Appian Cloud environments, install the Kafka Tools plugin from the Appian App Market
3. For self-managed environments, copy the *ps-plugin-KafkaTools-X.jar* file into your `$APPIAN_HOME/data/_admin/plugins` directory. The plug-in will be installed and available automatically.
4. Create Third-Party Credentials (explained below)
  - i. name: kafkaTools
  - ii. fields:
    - username
    - password
    - truststorepwd
    - keystorepwd
    - privatekeypwd
    - clientid
    - clientsecret

- tokenurl

5. Give Third-Party Credentials access to KafkaTools plugin

## Appian Secure Credentials Store

This plug-in uses a Third-Party Credentials Store to maintain the credentials for authentication with a Kafka Server. The configuration is available in the Admin Console under Third Party Credentials. The field names to be created are:

- **username:** username to use with the LoginModule. This is only required when using SASL or SASL\_SSL.
- **password:** password to use with the LoginModule. This is only required when using SASL or SASL\_SSL.
- **truststorepwd:** password of the TrustStore containing the Kafka server certificate. This is only required when using SSL or SASL\_SSL.
- **keystorepwd:** password of the KeyStore containing the client private key to use to authenticate with the Kafka server. This is only required when using SSL.
- **privatekeypwd:** password of the private key to use to authenticate with the Kafka server. This is only required when using SSL.
- **clientid:** The client id of the oauth provider to use to authenticate with the Kafka server. This is only required when using OAUTHBEARER.
- **clientsecret:** The client secret of the oauth provider to use to authenticate with the Kafka server. This is only required when using OAUTHBEARER.
- **tokenurl:** The token url of the oauth provider use to authenticate with the Kafka server. This is only required when using OAUTHBEARER.

It is recommended that you mask the value of the passwords. Once the entry is created, the name will be used as a required input parameter to Kafka Tools Smart Service nodes.

**Note:** you must have the plug-in installed prior to creating the credentials. You must add the plug-in to the *Plug-Ins List* to allow KafkaTools to access the credentials.

If your server does not require authentication, you must still create an entry in the Third Party Credentials.

## Smart Services

The nodes for these Smart Services are available under the *Integration Services* group and *Connectivity Services* subgroup of the Appian Process Modeler.

### Publish To Kafka

This Smart Service implements Kafka Producer and sends a single payload to a single Kafka topic per execution.

#### Input Parameters

- **Secure Credentials Store Key:** use the name created in the Admin Console under Third Party Credentials (see above)
- **Servers:** provide a comma-separated list of *server name:port* values
- **Topic:** provide the name of the kafka server topic
- **Key:** text value of the key the payload should be written to

- **Partition:** integer value of the partition the payload should be written to
- **Payload:** text value of the payload
- **Security Protocol:** provide the value of the security protocol to be used with the Kafka Server
- **SASL Mechanism:** if the security protocol is provided as "SASL\_SSL", then this parameter must be provided, use OAUTHBEARER for oauth auth
- **TrustStore:** if the security protocol is provided as SSL or SASL\_SSL, this parameter is an Appian document containing the TrustStore in JKS format
- **KeyStore:** if the security protocol is provided as SSL, this parameter is an Appian document containing the KeyStore in JKS format
- **Scope:** if using a security protocol with a provider that requires a scope, this property can be used to set the scope value for authentication

### Output Parameters

- **success:** returns *true* if smart service completed successfully and *false* if an exception has occurred
- **errorMessage:** returns *null* if smart service completed successfully and the exception message if an exception has occurred

### Known Issues

Currently only supports text based payloads

## Consume From Kafka

This Smart Service implements Kafka Consumer and subscribes to one topic on a Kafka server. It saves the payloads to a database table that is provided in the configuration of the node. The node runs for a configurable number of minutes and iterates between polling the Kafka server and sleeping.

### Input Parameters

- **Servers:** provide a comma-separated list of *server name:port* values
- **Datasource Name:** provide the value that is specified in the *Data Source* dropdown in any of the Data Stores in Designer
- **TransactionTableName:** name of the database table where the Kafka records are saved into. Set to the default value when using the Transaction Manager to process the Kafka records
- **Security Protocol:** provide the value of the security protocol to be used with the Kafka Server
- **Secure Credentials Store Key:** use the name created in the Admin Console under Third Party Credentials (see above)
- **SASL Mechanism:** if the security protocol is provided as "SASL\_SSL", then this parameter must be provided
- **TrustStore:** if the security protocol is provided as SSL or SASL\_SSL, this parameter is an Appian document containing the TrustStore in JKS format
- **KeyStore:** if the security protocol is provided as SSL, this parameter is an Appian document containing the KeyStore in JKS format
- **Runtime In Minutes:** number of minutes that the Smart Service run for. Any value above 58 minutes will be defaulted to 58 minutes as Smart Service nodes timeout after 60 minutes and result in a process error
- **Group Id:** provide the value for the Group Id to be used to subscribe to the Kafka topic
- **NumConsumers:** provide the number of consumers to use to consume records
- **Topics:** a list of Topics to subscribe to on the Kafka server

- **Job Type Id:** provide the integer of the transaction job type these messages will be assigned to. Refer to the Transaction Manager documentation for more details
- **Polling Interval in Ms:** provide the number of milliseconds for the Kafka Consumer to poll a topic per iteration
- **Sleeping Interval In Ms:** provide the number of milliseconds for the thread to sleep between polling iterations
- **Session Timeout:** provide the number of milliseconds for the session timeout (default value is 30000)
- **Auto Offset Reset Config:** provide the auto offset reset config (default value is latest)
- **Deserializer Class Name:** full name of the class to be used with Kafka Consumer (default is text-based deserializer)
- **avroSchemaRegistryUrl:** URL for the API endpoint for AVRO schema registry. If certificate is needed, it must be contained in truststore and keystore configuration. If specified, all topics configured to decode (consume) must be in AVRO format (no mixing of topics formats). To receive mixed format topics, 2 PM nodes needs to be configured - one for String based topics and one for Avro based topics.
- **Message Filter:** jsonpath filter used to filter out message before processing in Appian. JsonPath documentation can be [found here](#). Use [this site](#) for testing.
- **Scope:** if using a security protocol with a provider that requires a scope, this property can be used to set the scope value for authentication

### Output Parameters

- **success:** returns *true* if smart service completed successfully and *false* if an exception has occurred
- **errorMessage:** returns *null* if smart service completed successfully and the exception message if an exception has occurred

## Consume From Kafka - JWT Bearer Authorization Grant

This Smart Service implements a Kafka Consumer that authenticates using the OAuth 2.0 JWT Bearer Authorization Grant as defined in RFC 7523. The service generates a signed JWT assertion and exchanges it with the configured authorization server for an access token using the grant type `urn:ietf:params:oauth:grant-type:jwt-bearer`. The retrieved access token is then used to authenticate against the Kafka broker via SASL/OAUTHBEARER. The Smart Service subscribes to a single Kafka topic and persists the consumed payloads into a database table defined in the node configuration. The node executes for a configurable duration, alternating between polling the Kafka broker and sleeping.

### SSL / TLS Trust Configuration

This Smart Service relies exclusively on the default JVM trust configuration for SSL/TLS validation. TLS validation is performed using the standard Java TrustStore (typically the `cacerts` file bundled with the JVM). As a result, the service can establish SSL/TLS connections only with Kafka clusters whose server certificates are issued by a public Certificate Authority included in the default Java TrustStore.

### Appian Secure Credentials Store (for this smart service only)

This Smart Service uses a Third-Party Credentials Store to maintain the credentials for authentication with a Kafka Server. The configuration is available in the Admin Console under Third Party Credentials. The field names to be created are:

- `jwt.bearer.auth.grant.avro.registry.username:` (OPTIONAL) Username used for HTTP Basic Authentication when connecting to the Avro Schema Registry. This property must be provided together with `jwt.bearer.auth.grant.avro.registry.password`. It is independent from the OAuth 2.0 JWT authentication used for Kafka broker access.

- `jwt.bearer.auth.grant.avro.registry.password`: (OPTIONAL) Password used for HTTP Basic Authentication when connecting to the Avro Schema Registry. This value is paired with `jwt.bearer.auth.grant.avro.registry.username`. This credential is only used for Schema Registry access and is not related to the OAuth 2.0 JWT Bearer Authorization Grant used for Kafka authentication.
- `jwt.bearer.auth.grant.tokenuri`: OAuth 2.0 token endpoint URL. The Smart Service sends the signed JWT assertion to this endpoint using the grant type: `urn:ietf:params:oauth:grant-type:jwt-bearer`
- `jwt.bearer.auth.grant.privatekey`: Private key used to sign the JWT assertion in accordance with RFC 7523. Supported formats (PEM, unencrypted): PKCS#1 RSA and PKCS#8. The key must be RSA, unencrypted, and correspond to the public key registered with the authorization server and match the algorithm used to sign the JWT (RS256).
- `jwt.bearer.auth.grant.privatekeyid`: Key identifier (kid) included in the JWT header. Used by the authorization server to identify the correct public key for signature validation when multiple keys are registered for the client.
- `jwt.bearer.auth.grant.clientid`: OAuth 2.0 Client Identifier representing the service account. This value is used as the `iss` (issuer) claim in the JWT.

It is recommended that you mask the value of the passwords. Once the entry is created, the name will be used as a required input parameter to Kafka Tools Smart Service nodes.

**Note:** you must have the plug-in installed prior to creating the credentials. You must add the plug-in to the *Plug-Ins List* to allow KafkaTools to access the credentials.

If your server does not require authentication, you must still create an entry in the Third Party Credentials.

### Input Parameters

- **Servers:** provide a comma-separated list of `server name:port` values
- **Datasource Name:** provide the value that is specified in the *Data Source* dropdown in any of the Data Stores in Designer
- **TransactionTableName:** name of the database table where the Kafka records are saved into. Set to the default value when using the Transaction Manager to process the Kafka records
- **Secure Credentials Store Key:** use the name created in the Admin Console under Third Party Credentials (see above)
- **Runtime In Minutes:** number of minutes that the Smart Service runs for. Any value above 58 minutes will be defaulted to 58 minutes as Smart Service nodes timeout after 60 minutes and result in a process error
- **Group Id:** provide the value for the Group Id to be used to subscribe to the Kafka topic
- **NumConsumers:** provide the number of consumers to use to consume records
- **Topics:** a list of exact topic names to subscribe to on the Kafka server. Note: You must provide either Topics OR TopicPattern, but not both.
- **TopicPattern:** a regular expression (regex) string used to subscribe to multiple topics that match the pattern (e.g., `^corp.payments.*`). Note: You must provide either Topics OR TopicPattern, but not both. Providing both will result in a validation error.
- **AvroSchemaRegistryUrl:** URL for the API endpoint for AVRO schema registry. If specified, all topics configured to decode (consume) must be in AVRO format.
- **Job Type Id:** provide the integer of the transaction job type these messages will be assigned to. Refer to the Transaction Manager documentation for more details
- **Polling Interval in Ms:** provide the number of milliseconds for the Kafka Consumer to poll a topic per iteration
- **Sleeping Interval in Ms:** provide the number of milliseconds for the thread to sleep between polling iterations
- **Session Timeout:** provide the number of milliseconds for the session timeout (default value is 30000)
- **Auto Offset Reset Config:** provide the auto offset reset config (default value is latest)

- **Message Filter:** jsonpath filter used to filter out message before processing in Appian. JsonPath documentation can be [found here](#). Use [this site](#) for testing.

### Output Parameters

- **success:** returns *true* if smart service completed successfully and *false* if an exception has occurred
- **errorMessage:** returns *null* if smart service completed successfully and the exception message if an exception has occurred

### Payload Handler Process Models

These models will be configured and assigned through the transaction manager job types. See documentation for Transaction Manager [Transaction Manager](#)