Red Hat AMQ Streams 2.3

Kafka configuration properties

Use configuration properties to configure Kafka components

Last Updated: 2023-01-16

Red Hat AMQ Streams 2.3 Kafka configuration properties

Use configuration properties to configure Kafka components

Legal Notice

Abstract

Get the most out of how Kafka components operate using Kafka configuration properties.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. BROKER CONFIGURATION PROPERTIES

advertised.listeners

Type: string
Default: null
Importance: high
Dynamic update: per-broker
Listeners to publish to ZooKeeper for clients to use, if different than the **listeners** config property. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, the value for **listeners** will be used. Unlike **listeners**, it is not valid to advertise the 0.0.0.0 meta-address. Also unlike **listeners**, there can be duplicated ports in this property, so that one listener can be configured to advertise another listener's address. This can be useful in some cases where external load balancers are used.

auto.create.topics.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
Enable auto creation of topic on the server.

auto.leader.rebalance.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
Enables auto leader balancing. A background thread checks the distribution of partition leaders at regular intervals, configurable by **leader.imbalance.check.interval.seconds**. If the leader imbalance exceeds **leader.imbalance.per.broker.percentage**, leader rebalance to the preferred leader for partitions is triggered.

background.threads

Type: int
Default: 10
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
The number of threads to use for various background processing tasks.

broker.id

Type: int
Default: -1
Importance: high
Dynamic update: read-only
The broker id for this server. If unset, a unique broker id will be generated.To avoid conflicts between zookeeper generated broker id's and user configured broker id's, generated broker ids start from reserved.broker.max.id + 1.

compression.type

Type: string
Default: producer
Valid Values: [uncompressed, zstd, lz4, snappy, gzip, producer]

Importance: high
Dynamic update: cluster-wide
Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', 'lz4', 'zstd'). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.

control.plane.listener.name

Type: string
Default: null
Importance: high
Dynamic update: read-only
Name of listener used for communication between controller and brokers. Broker will use the control.plane.listener.name to locate the endpoint in listeners list, to listen for connections from the controller. For example, if a broker's config is : listeners = INTERNAL://192.1.1.8:9092, EXTERNAL://10.1.1.5:9093, CONTROLLER://192.1.1.8:9094 listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSL control.plane.listener.name = CONTROLLER On startup, the broker will start listening on "192.1.1.8:9094" with security protocol "SSL". On controller side, when it discovers a broker's published endpoints through zookeeper, it will use the control.plane.listener.name to find the endpoint, which it will use to establish connection to the broker. For example, if the broker's published endpoints on zookeeper are : "endpoints" : ["INTERNAL://broker1.example.com:9092","EXTERNAL://broker1.example.com:9093","CONTROLLER and the controller's config is : listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSL control.plane.listener.name = CONTROLLER then controller will use "broker1.example.com:9094" with security protocol "SSL" to connect to the broker. If not explicitly configured, the default value will be null and there will be no dedicated endpoints for controller connections. If explicitly configured, the value cannot be the same as the value of **inter.broker.listener.name**.

controller.listener.names

Type: string
Default: null
Importance: high
Dynamic update: read-only
A comma-separated list of the names of the listeners used by the controller. This is required if running in KRaft mode. When communicating with the controller quorum, the broker will always use the first listener in this list. Note: The ZK-based controller should not set this configuration.

controller.quorum.election.backoff.max.ms

Type: int
Default: 1000 (1 second)
Importance: high
Dynamic update: read-only
Maximum time in milliseconds before starting new elections. This is used in the binary exponential backoff mechanism that helps prevent gridlocked elections.

controller.quorum.election.timeout.ms

Type: int
Default: 1000 (1 second)
Importance: high
Dynamic update: read-only
Maximum time in milliseconds to wait without being able to fetch from the leader before triggering a new election.

controller.quorum.fetch.timeout.ms

Type: int
Default: 2000 (2 seconds)
Importance: high
Dynamic update: read-only
Maximum time without a successful fetch from the current leader before becoming a candidate and triggering an election for voters; Maximum time without receiving fetch from a majority of the quorum before asking around to see if there's a new epoch for leader.

controller.quorum.voters

Type: list
Default: ""
Valid Values: non-empty list
Importance: high
Dynamic update: read-only
Map of id/endpoint information for the set of voters in a comma-separated list of **{id}@{host}:{port}** entries. For example: **1@localhost:9092,2@localhost:9093,3@localhost:9094**.

delete.topic.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
Enables delete topic. Delete topic through the admin tool will have no effect if this config is turned off.

early.start.listeners

Type: string
Default: null
Importance: high
Dynamic update: read-only
A comma-separated list of listener names which may be started before the authorizer has finished initialization. This is useful when the authorizer is dependent on the cluster itself for bootstrapping, as is the case for the StandardAuthorizer (which stores ACLs in the metadata log.) By default, all listeners included in controller.listener.names will also be early start listeners. A listener should not appear in this list if it accepts external traffic.

leader.imbalance.check.interval.seconds

Type: long
Default: 300
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The frequency with which the partition rebalance check is triggered by the controller.

leader.imbalance.per.broker.percentage

Type: int
Default: 10
Importance: high
Dynamic update: read-only
The ratio of leader imbalance allowed per broker. The controller would trigger a leader balance if it goes above this value per broker. The value is specified in percentage.

listeners

Type: string
Default: PLAINTEXT://:9092
Importance: high
Dynamic update: per-broker
Listener List - Comma-separated list of URIs we will listen on and the listener names. If the listener name is not a security protocol, **listener.security.protocol.map** must also be set. Listener names and port numbers must be unique. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: PLAINTEXT://myhost:9092,SSL://:9091 CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093.

log.dir

Type: string
Default: /tmp/kafka-logs
Importance: high
Dynamic update: read-only
The directory in which the log data is kept (supplemental for log.dirs property).

log.dirs

Type: string
Default: null
Importance: high
Dynamic update: read-only
A comma-separated list of the directories where the log data is stored. If not set, the value in log.dir is used.

log.flush.interval.messages

Type: long
Default: 9223372036854775807
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
The number of messages accumulated on a log partition before messages are flushed to disk.

log.flush.interval.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
The maximum time in ms that a message in any topic is kept in memory before flushed to disk. If not set, the value in log.flush.scheduler.interval.ms is used.

log.flush.offset.checkpoint.interval.ms

Type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
The frequency with which we update the persistent record of the last flush which acts as the log recovery point.

log.flush.scheduler.interval.ms

Type: long
Default: 9223372036854775807
Importance: high
Dynamic update: read-only
The frequency in ms that the log flusher checks whether any log needs to be flushed to disk.

log.flush.start.offset.checkpoint.interval.ms

Type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
The frequency with which we update the persistent record of log start offset.

log.retention.bytes

Type: long
Default: -1
Importance: high
Dynamic update: cluster-wide
The maximum size of the log before deleting it.

log.retention.hours

Type: int
Default: 168
Importance: high
Dynamic update: read-only
The number of hours to keep a log file before deleting it (in hours), tertiary to log.retention.ms property.

log.retention.minutes

Type: int
Default: null
Importance: high
Dynamic update: read-only
The number of minutes to keep a log file before deleting it (in minutes), secondary to log.retention.ms property. If not set, the value in log.retention.hours is used.

log.retention.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
The number of milliseconds to keep a log file before deleting it (in milliseconds), If not set, the value in log.retention.minutes is used. If set to -1, no time limit is applied.

log.roll.hours

Type: int
Default: 168
Valid Values: [1,...]
Importance: high
Dynamic update: read-only

The maximum time before a new log segment is rolled out (in hours), secondary to log.roll.ms property.

log.roll.jitter.hours

Type: int
Default: 0
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
The maximum jitter to subtract from logRollTimeMillis (in hours), secondary to log.roll.jitter.ms property.

log.roll.jitter.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
The maximum jitter to subtract from logRollTimeMillis (in milliseconds). If not set, the value in log.roll.jitter.hours is used.

log.roll.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
The maximum time before a new log segment is rolled out (in milliseconds). If not set, the value in log.roll.hours is used.

log.segment.bytes

Type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [14,...]
Importance: high
Dynamic update: cluster-wide
The maximum size of a single log file.

log.segment.delete.delay.ms

Type: long
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: cluster-wide
The amount of time to wait before deleting a file from the filesystem.

message.max.bytes

Type: int
Default: 1048588
Valid Values: [0,...]
Importance: high
Dynamic update: cluster-wide
The largest record batch size allowed by Kafka (after compression if compression is enabled). If this is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be

increased so that they can fetch record batches this large. In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case.This can be set per topic with the topic level **max.message.bytes** config.

metadata.log.dir

Type: string
Default: null
Importance: high
Dynamic update: read-only
This configuration determines where we put the metadata log for clusters in KRaft mode. If it is not set, the metadata log is placed in the first log directory from log.dirs.

metadata.log.max.record.bytes.between.snapshots

Type: long
Default: 20971520
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
This is the maximum number of bytes in the log between the latest snapshot and the high-watermark needed before generating a new snapshot.

metadata.log.segment.bytes

Type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [12,...]
Importance: high
Dynamic update: read-only
The maximum size of a single metadata log file.

metadata.log.segment.ms

Type: long
Default: 604800000 (7 days)
Importance: high
Dynamic update: read-only
The maximum time before a new metadata log file is rolled out (in milliseconds).

metadata.max.retention.bytes

Type: long
Default: -1
Importance: high
Dynamic update: read-only
The maximum combined size of the metadata log and snapshots before deleting old snapshots and log files. Since at least one snapshot must exist before any logs can be deleted, this is a soft limit.

metadata.max.retention.ms

Type: long
Default: 604800000 (7 days)
Importance: high
Dynamic update: read-only
The number of milliseconds to keep a metadata log file or snapshot before deleting it. Since at least one snapshot must exist before any logs can be deleted, this is a soft limit.

min.insync.replicas

Type: int
Default: 1
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
When a producer sets acks to "all" (or "-1"), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.

node.id

Type: int
Default: -1
Importance: high
Dynamic update: read-only
The node ID associated with the roles this process is playing when **process.roles** is non-empty. Every node in a KRaft cluster must have a unique **node.id**, this includes broker and controller nodes. This is required configuration when running in KRaft mode.

num.io.threads

Type: int
Default: 8
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
The number of threads that the server uses for processing requests, which may include disk I/O.

num.network.threads

Type: int
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
The number of threads that the server uses for receiving requests from the network and sending responses to the network.

num.recovery.threads.per.data.dir

Type: int
Default: 1
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.

num.replica.alter.log.dirs.threads

Type: int
Default: null

Importance: high
Dynamic update: read-only
The number of threads that can move replicas between log directories, which may include disk I/O.

num.replica.fetchers

Type: int
Default: 1
Importance: high
Dynamic update: cluster-wide
Number of fetcher threads used to replicate records from each source broker. The total number of fetchers on each broker is bound by **num.replica.fetchers** multiplied by the number of brokers in the cluster.Increasing this value can increase the degree of I/O parallelism in the follower and leader broker at the cost of higher CPU and memory utilization.

offset.metadata.max.bytes

Type: int
Default: 4096 (4 kibibytes)
Importance: high
Dynamic update: read-only
The maximum size for a metadata entry associated with an offset commit.

offsets.commit.required.acks

Type: short
Default: -1
Importance: high
Dynamic update: read-only
The required acks before the commit can be accepted. In general, the default (-1) should not be overridden.

offsets.commit.timeout.ms

Type: int
Default: 5000 (5 seconds)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
Offset commit will be delayed until all replicas for the offsets topic receive the commit or this timeout is reached. This is similar to the producer request timeout.

offsets.load.buffer.size

Type: int
Default: 5242880
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
Batch size for reading from the offsets segments when loading offsets into the cache (soft-limit, overridden if records are too large).

offsets.retention.check.interval.ms

Type: long
Default: 600000 (10 minutes)
Valid Values: [1,...]

Importance: high
Dynamic update: read-only
Frequency at which to check for stale offsets.

offsets.retention.minutes

Type: int
Default: 10080
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
After a consumer group loses all its consumers (i.e. becomes empty) its offsets will be kept for this retention period before getting discarded. For standalone consumers (using manual assignment), offsets will be expired after the time of last commit plus this retention period.

offsets.topic.compression.codec

Type: int
Default: 0
Importance: high
Dynamic update: read-only
Compression codec for the offsets topic - compression may be used to achieve "atomic" commits.

offsets.topic.num.partitions

Type: int
Default: 50
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The number of partitions for the offset commit topic (should not change after deployment).

offsets.topic.replication.factor

Type: short
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The replication factor for the offsets topic (set higher to ensure availability). Internal topic creation will fail until the cluster size meets this replication factor requirement.

offsets.topic.segment.bytes

Type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The offsets topic segment bytes should be kept relatively small in order to facilitate faster log compaction and cache loads.

process.roles

Type: list
Default: ""
Valid Values: [broker, controller]

Importance: high
Dynamic update: read-only
The roles that this process plays: 'broker', 'controller', or 'broker,controller' if it is both. This configuration is only applicable for clusters in KRaft (Kafka Raft) mode (instead of ZooKeeper). Leave this config undefined or empty for Zookeeper clusters.

queued.max.requests

Type: int
Default: 500
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The number of queued requests allowed for data-plane, before blocking the network threads.

replica.fetch.min.bytes

Type: int
Default: 1
Importance: high
Dynamic update: read-only
Minimum bytes expected for each fetch response. If not enough bytes, wait up to **replica.fetch.wait.max.ms** (broker config).

replica.fetch.wait.max.ms

Type: int
Default: 500
Importance: high
Dynamic update: read-only
The maximum wait time for each fetcher request issued by follower replicas. This value should always be less than the replica.lag.time.max.ms at all times to prevent frequent shrinking of ISR for low throughput topics.

replica.high.watermark.checkpoint.interval.ms

Type: long
Default: 5000 (5 seconds)
Importance: high
Dynamic update: read-only
The frequency with which the high watermark is saved out to disk.

replica.lag.time.max.ms

Type: long
Default: 30000 (30 seconds)
Importance: high
Dynamic update: read-only
If a follower hasn't sent any fetch requests or hasn't consumed up to the leaders log end offset for at least this time, the leader will remove the follower from isr.

replica.socket.receive.buffer.bytes

Type: int
Default: 65536 (64 kibibytes)
Importance: high
Dynamic update: read-only
The socket receive buffer for network requests.

replica.socket.timeout.ms

Type: int
Default: 30000 (30 seconds)
Importance: high
Dynamic update: read-only
The socket timeout for network requests. Its value should be at least replica.fetch.wait.max.ms.

request.timeout.ms

Type: int
Default: 30000 (30 seconds)
Importance: high
Dynamic update: read-only
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

sasl.mechanism.controller.protocol

Type: string
Default: GSSAPI
Importance: high
Dynamic update: read-only
SASL mechanism used for communication with controllers. Default is GSSAPI.

socket.receive.buffer.bytes

Type: int
Default: 102400 (100 kibibytes)
Importance: high
Dynamic update: read-only
The SO_RCVBUF buffer of the socket server sockets. If the value is -1, the OS default will be used.

socket.request.max.bytes

Type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The maximum number of bytes in a socket request.

socket.send.buffer.bytes

Type: int
Default: 102400 (100 kibibytes)
Importance: high
Dynamic update: read-only
The SO_SNDBUF buffer of the socket server sockets. If the value is -1, the OS default will be used.

transaction.max.timeout.ms

Type: int
Default: 900000 (15 minutes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only

The maximum allowed timeout for transactions. If a client's requested transaction time exceed this, then the broker will return an error in InitProducerIdRequest. This prevents a client from too large of a timeout, which can stall consumers reading from topics included in the transaction.

transaction.state.log.load.buffer.size

Type: int
Default: 5242880
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
Batch size for reading from the transaction log segments when loading producer ids and transactions into the cache (soft-limit, overridden if records are too large).

transaction.state.log.min.isr

Type: int
Default: 2
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
Overridden min.insync.replicas config for the transaction topic.

transaction.state.log.num.partitions

Type: int
Default: 50
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The number of partitions for the transaction topic (should not change after deployment).

transaction.state.log.replication.factor

Type: short
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The replication factor for the transaction topic (set higher to ensure availability). Internal topic creation will fail until the cluster size meets this replication factor requirement.

transaction.state.log.segment.bytes

Type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The transaction topic segment bytes should be kept relatively small in order to facilitate faster log compaction and cache loads.

transactional.id.expiration.ms

Type: int
Default: 604800000 (7 days)
Valid Values: [1,...]

Importance: high
Dynamic update: read-only
The time in ms that the transaction coordinator will wait without receiving any transaction status updates for the current transaction before expiring its transactional id. This setting also influences producer id expiration - producer ids are expired once this time has elapsed after the last write with the given producer id. Note that producer ids may expire sooner if the last write from the producer id is deleted due to the topic's retention settings.

unclean.leader.election.enable

Type: boolean
Default: false
Importance: high
Dynamic update: cluster-wide
Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss.

zookeeper.connect

Type: string
Default: null
Importance: high
Dynamic update: read-only
Specifies the ZooKeeper connection string in the form **hostname:port** where host and port are the host and port of a ZooKeeper server. To allow connecting through other ZooKeeper nodes when that ZooKeeper machine is down you can also specify multiple hosts in the form **hostname1:port1,hostname2:port2,hostname3:port3**. The server can also have a ZooKeeper chroot path as part of its ZooKeeper connection string which puts its data under some path in the global ZooKeeper namespace. For example to give a chroot path of **/chroot/path** you would give the connection string as **hostname1:port1,hostname2:port2,hostname3:port3/chroot/path**.

zookeeper.connection.timeout.ms

Type: int
Default: null
Importance: high
Dynamic update: read-only
The max time that the client waits to establish a connection to zookeeper. If not set, the value in zookeeper.session.timeout.ms is used.

zookeeper.max.in.flight.requests

Type: int
Default: 10
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
The maximum number of unacknowledged requests the client will send to Zookeeper before blocking.

zookeeper.session.timeout.ms

Type: int
Default: 18000 (18 seconds)
Importance: high
Dynamic update: read-only
Zookeeper session timeout.

zookeeper.set.acl

Type: boolean
Default: false
Importance: high
Dynamic update: read-only
Set client to use secure ACLs.

broker.heartbeat.interval.ms

Type: int
Default: 2000 (2 seconds)
Importance: medium
Dynamic update: read-only
The length of time in milliseconds between broker heartbeats. Used when running in KRaft mode.

broker.id.generation.enable

Type: boolean
Default: true
Importance: medium
Dynamic update: read-only
Enable automatic broker id generation on the server. When enabled the value configured for reserved.broker.max.id should be reviewed.

broker.rack

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Rack of the broker. This will be used in rack aware replication assignment for fault tolerance. Examples: **RACK1**, **us-east-1d**.

broker.session.timeout.ms

Type: int
Default: 9000 (9 seconds)
Importance: medium
Dynamic update: read-only
The length of time in milliseconds that a broker lease lasts if no heartbeats are made. Used when running in KRaft mode.

connections.max.idle.ms

Type: long
Default: 600000 (10 minutes)
Importance: medium
Dynamic update: read-only
Idle connections timeout: the server socket processor threads close the connections that idle more than this.

connections.max.reauth.ms

Type: long
Default: 0
Importance: medium
Dynamic update: read-only

When explicitly set to a positive number (the default is 0, not a positive number), a session lifetime that will not exceed the configured value will be communicated to v2.2.0 or later clients when they authenticate. The broker will disconnect any such connection that is not re-authenticated within the session lifetime and that is then subsequently used for any purpose other than re-authentication. Configuration names can optionally be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.oauthbearer.connections.max.reauth.ms=3600000.

controlled.shutdown.enable

Type: boolean
Default: true
Importance: medium
Dynamic update: read-only
Enable controlled shutdown of the server.

controlled.shutdown.max.retries

Type: int
Default: 3
Importance: medium
Dynamic update: read-only
Controlled shutdown can fail for multiple reasons. This determines the number of retries when such failure happens.

controlled.shutdown.retry.backoff.ms

Type: long
Default: 5000 (5 seconds)
Importance: medium
Dynamic update: read-only
Before each retry, the system needs time to recover from the state that caused the previous failure (Controller fail over, replica lag etc). This config determines the amount of time to wait before retrying.

controller.quorum.append.linger.ms

Type: int
Default: 25
Importance: medium
Dynamic update: read-only
The duration in milliseconds that the leader will wait for writes to accumulate before flushing them to disk.

controller.quorum.request.timeout.ms

Type: int
Default: 2000 (2 seconds)
Importance: medium
Dynamic update: read-only
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

controller.socket.timeout.ms

Type: int
Default: 30000 (30 seconds)
Importance: medium

Dynamic update: read-only
The socket timeout for controller-to-broker channels.

default.replication.factor

Type: int
Default: 1
Importance: medium
Dynamic update: read-only
The default replication factors for automatically created topics.

delegation.token.expiry.time.ms

Type: long
Default: 86400000 (1 day)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
The token validity time in miliseconds before the token needs to be renewed. Default value 1 day.

delegation.token.master.key

Type: password
Default: null
Importance: medium
Dynamic update: read-only
DEPRECATED: An alias for delegation.token.secret.key, which should be used instead of this config.

delegation.token.max.lifetime.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
The token has a maximum lifetime beyond which it cannot be renewed anymore. Default value 7 days.

delegation.token.secret.key

Type: password
Default: null
Importance: medium
Dynamic update: read-only
Secret key to generate and verify delegation tokens. The same key must be configured across all the brokers. If the key is not set or set to empty string, brokers will disable the delegation token support.

delete.records.purgatory.purge.interval.requests

Type: int
Default: 1
Importance: medium
Dynamic update: read-only
The purge interval (in number of requests) of the delete records request purgatory.

fetch.max.bytes

Type: int

Default: 57671680 (55 mebibytes)
Valid Values: [1024,...]
Importance: medium
Dynamic update: read-only
The maximum number of bytes we will return for a fetch request. Must be at least 1024.

fetch.purgatory.purge.interval.requests

Type: int
Default: 1000
Importance: medium
Dynamic update: read-only
The purge interval (in number of requests) of the fetch request purgatory.

group.initial.rebalance.delay.ms

Type: int
Default: 3000 (3 seconds)
Importance: medium
Dynamic update: read-only
The amount of time the group coordinator will wait for more consumers to join a new group before performing the first rebalance. A longer delay means potentially fewer rebalances, but increases the time until processing begins.

group.max.session.timeout.ms

Type: int
Default: 1800000 (30 minutes)
Importance: medium
Dynamic update: read-only
The maximum allowed session timeout for registered consumers. Longer timeouts give consumers more time to process messages in between heartbeats at the cost of a longer time to detect failures.

group.max.size

Type: int
Default: 2147483647
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
The maximum number of consumers that a single consumer group can accommodate.

group.min.session.timeout.ms

Type: int
Default: 6000 (6 seconds)
Importance: medium
Dynamic update: read-only
The minimum allowed session timeout for registered consumers. Shorter timeouts result in quicker failure detection at the cost of more frequent consumer heartbeating, which can overwhelm broker resources.

initial.broker.registration.timeout.ms

Type: int
Default: 60000 (1 minute)
Importance: medium
Dynamic update: read-only

When initially registering with the controller quorum, the number of milliseconds to wait before declaring failure and exiting the broker process.

inter.broker.listener.name

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Name of listener used for communication between brokers. If this is unset, the listener name is defined by security.inter.broker.protocol. It is an error to set this and security.inter.broker.protocol properties at the same time.

inter.broker.protocol.version

Type: string
Default: 3.3-IV3
Valid Values: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3]
Importance: medium
Dynamic update: read-only
Specify which version of the inter-broker protocol will be used. This is typically bumped after all brokers were upgraded to a new version. Example of some valid values are: 0.8.0, 0.8.1, 0.8.1.1, 0.8.2, 0.8.2.0, 0.8.2.1, 0.9.0.0, 0.9.0.1 Check MetadataVersion for the full list.

log.cleaner.backoff.ms

Type: long
Default: 15000 (15 seconds)
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The amount of time to sleep when there are no logs to clean.

log.cleaner.dedupe.buffer.size

Type: long
Default: 134217728
Importance: medium
Dynamic update: cluster-wide
The total memory used for log deduplication across all cleaner threads.

log.cleaner.delete.retention.ms

Type: long
Default: 86400000 (1 day)
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The amount of time to retain delete tombstone markers for log compacted topics. This setting also gives a bound on the time in which a consumer must complete a read if they begin from offset 0 to ensure that they get a valid snapshot of the final stage (otherwise delete tombstones may be collected before they complete their scan).

log.cleaner.enable

Type: boolean
Default: true
Importance: medium
Dynamic update: read-only
Enable the log cleaner process to run on the server. Should be enabled if using any topics with a cleanup.policy=compact including the internal offsets topic. If disabled those topics will not be compacted and continually grow in size.

log.cleaner.io.buffer.load.factor

Type: double
Default: 0.9
Importance: medium
Dynamic update: cluster-wide
Log cleaner dedupe buffer load factor. The percentage full the dedupe buffer can become. A higher value will allow more log to be cleaned at once but will lead to more hash collisions.

log.cleaner.io.buffer.size

Type: int
Default: 524288
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The total memory used for log cleaner I/O buffers across all cleaner threads.

log.cleaner.io.max.bytes.per.second

Type: double
Default: 1.7976931348623157E308
Importance: medium
Dynamic update: cluster-wide
The log cleaner will be throttled so that the sum of its read and write i/o will be less than this value on average.

log.cleaner.max.compaction.lag.ms

Type: long
Default: 9223372036854775807
Valid Values: [1,...]
Importance: medium
Dynamic update: cluster-wide
The maximum time a message will remain ineligible for compaction in the log. Only applicable for logs that are being compacted.

log.cleaner.min.cleanable.ratio

Type: double
Default: 0.5
Valid Values: [0,...,1]
Importance: medium
Dynamic update: cluster-wide
The minimum ratio of dirty log to total log for a log to eligible for cleaning. If the log.cleaner.max.compaction.lag.ms or the log.cleaner.min.compaction.lag.ms configurations are also specified, then the log compactor considers the log eligible for compaction as soon as either: (i) the

dirty ratio threshold has been met and the log has had dirty (uncompacted) records for at least the log.cleaner.min.compaction.lag.ms duration, or (ii) if the log has had dirty (uncompacted) records for at most the log.cleaner.max.compaction.lag.ms period.

log.cleaner.min.compaction.lag.ms

Type: long
Default: 0
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The minimum time a message will remain uncompacted in the log. Only applicable for logs that are being compacted.

log.cleaner.threads

Type: int
Default: 1
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The number of background threads to use for log cleaning.

log.cleanup.policy

Type: list
Default: delete
Valid Values: [compact, delete]
Importance: medium
Dynamic update: cluster-wide
The default cleanup policy for segments beyond the retention window. A comma separated list of valid policies. Valid policies are: "delete" and "compact".

log.index.interval.bytes

Type: int
Default: 4096 (4 kibibytes)
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The interval with which we add an entry to the offset index.

log.index.size.max.bytes

Type: int
Default: 10485760 (10 mebibytes)
Valid Values: [4,...]
Importance: medium
Dynamic update: cluster-wide
The maximum size in bytes of the offset index.

log.message.format.version

Type: string
Default: 3.0-IV1
Valid Values: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1,

3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3]
Importance: medium
Dynamic update: read-only
Specify the message format version the broker will use to append messages to the logs. The value should be a valid MetadataVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check MetadataVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.

log.message.timestamp.difference.max.ms

Type: long
Default: 9223372036854775807
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If log.message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if log.message.timestamp.type=LogAppendTime.The maximum timestamp difference allowed should be no greater than log.retention.ms to avoid unnecessarily frequent log rolling.

log.message.timestamp.type

Type: string
Default: CreateTime
Valid Values: [CreateTime, LogAppendTime]
Importance: medium
Dynamic update: cluster-wide
Define whether the timestamp in the message is message create time or log append time. The value should be either **CreateTime** or **LogAppendTime**.

log.preallocate

Type: boolean
Default: false
Importance: medium
Dynamic update: cluster-wide
Should pre allocate file when create new segment? If you are using Kafka on Windows, you probably need to set it to true.

log.retention.check.interval.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
The frequency in milliseconds that the log cleaner checks whether any log is eligible for deletion.

max.connection.creation.rate

Type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide

The maximum connection creation rate we allow in the broker at any time. Listener-level limits may also be configured by prefixing the config name with the listener prefix, for example, **listener.name.internal.max.connection.creation.rate**.Broker-wide connection rate limit should be configured based on broker capacity while listener limits should be configured based on application requirements. New connections will be throttled if either the listener or the broker limit is reached, with the exception of inter-broker listener. Connections on the inter-broker listener will be throttled only when the listener-level rate limit is reached.

max.connections

Type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The maximum number of connections we allow in the broker at any time. This limit is applied in addition to any per-ip limits configured using max.connections.per.ip. Listener-level limits may also be configured by prefixing the config name with the listener prefix, for example, **listener.name.internal.max.connections**. Broker-wide limit should be configured based on broker capacity while listener limits should be configured based on application requirements. New connections are blocked if either the listener or broker limit is reached. Connections on the inter-broker listener are permitted even if broker-wide limit is reached. The least recently used connection on another listener will be closed in this case.

max.connections.per.ip

Type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
The maximum number of connections we allow from each ip address. This can be set to 0 if there are overrides configured using max.connections.per.ip.overrides property. New connections from the ip address are dropped if the limit is reached.

max.connections.per.ip.overrides

Type: string
Default: ""
Importance: medium
Dynamic update: cluster-wide
A comma-separated list of per-ip or hostname overrides to the default maximum number of connections. An example value is "hostName:100,127.0.0.1:200".

max.incremental.fetch.session.cache.slots

Type: int
Default: 1000
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
The maximum number of incremental fetch sessions that we will maintain.

num.partitions

Type: int
Default: 1
Valid Values: [1,...]

Importance: medium
Dynamic update: read-only
The default number of log partitions per topic.

password.encoder.old.secret

Type: password
Default: null
Importance: medium
Dynamic update: read-only
The old secret that was used for encoding dynamically configured passwords. This is required only when the secret is updated. If specified, all dynamically encoded passwords are decoded using this old secret and re-encoded using password.encoder.secret when broker starts up.

password.encoder.secret

Type: password
Default: null
Importance: medium
Dynamic update: read-only
The secret used for encoding dynamically configured passwords for this broker.

principal.builder.class

Type: class
Default: org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalBuilder
Importance: medium
Dynamic update: per-broker
The fully qualified name of a class that implements the KafkaPrincipalBuilder interface, which is used to build the KafkaPrincipal object used during authorization. If no principal builder is defined, the default behavior depends on the security protocol in use. For SSL authentication, the principal will be derived using the rules defined by **ssl.principal.mapping.rules** applied on the distinguished name from the client certificate if one is provided; otherwise, if client authentication is not required, the principal name will be ANONYMOUS. For SASL authentication, the principal will be derived using the rules defined by **sasl.kerberos.principal.to.local.rules** if GSSAPI is in use, and the SASL authentication ID for other mechanisms. For PLAINTEXT, the principal will be ANONYMOUS.

producer.purgatory.purge.interval.requests

Type: int
Default: 1000
Importance: medium
Dynamic update: read-only
The purge interval (in number of requests) of the producer request purgatory.

queued.max.request.bytes

Type: long
Default: -1
Importance: medium
Dynamic update: read-only
The number of queued bytes allowed before no more requests are read.

replica.fetch.backoff.ms

Type: int
Default: 1000 (1 second)
Valid Values: [0,...]

Importance: medium
Dynamic update: read-only
The amount of time to sleep when fetch partition error occurs.

replica.fetch.max.bytes

Type: int
Default: 1048576 (1 mebibyte)
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
The number of bytes of messages to attempt to fetch for each partition. This is not an absolute maximum, if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that progress can be made. The maximum record batch size accepted by the broker is defined via **message.max.bytes** (broker config) or **max.message.bytes** (topic config).

replica.fetch.response.max.bytes

Type: int
Default: 10485760 (10 mebibytes)
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
Maximum bytes expected for the entire fetch response. Records are fetched in batches, and if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that progress can be made. As such, this is not an absolute maximum. The maximum record batch size accepted by the broker is defined via **message.max.bytes** (broker config) or **max.message.bytes** (topic config).

replica.selector.class

Type: string
Default: null
Importance: medium
Dynamic update: read-only
The fully qualified class name that implements ReplicaSelector. This is used by the broker to find the preferred read replica. By default, we use an implementation that returns the leader.

reserved.broker.max.id

Type: int
Default: 1000
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
Max number that can be used for a broker.id.

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.

sasl.enabled.mechanisms

Type: list
Default: GSSAPI
Importance: medium
Dynamic update: per-broker
The list of SASL mechanisms enabled in the Kafka server. The list may contain any mechanism for which a security provider is available. Only GSSAPI is enabled by default.

sasl.jaas.config

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here. The format for the value is: **loginModuleClass controlFlag (optionName=optionValue)*;**. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;.

sasl.kerberos.kinit.cmd

Type: string
Default: /usr/bin/kinit
Importance: medium
Dynamic update: per-broker
Kerberos kinit command path.

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: medium
Dynamic update: per-broker
Login thread sleep time between refresh attempts.

sasl.kerberos.principal.to.local.rules

Type: list
Default: DEFAULT
Importance: medium
Dynamic update: per-broker
A list of rules for mapping from principal names to short names (typically operating system usernames). The rules are evaluated in order and the first rule that matches a principal name is used to map it to a short name. Any later rules in the list are ignored. By default, principal names of the form {username}/{hostname}@{REALM} are mapped to {username}. For more details on the format please see security authorization and acls. Note that this configuration is ignored if an extension of KafkaPrincipalBuilder is provided by the **principal.builder.class** configuration.

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

sasl.kerberos.ticket.renew.jitter

Type: double
Default: 0.05
Importance: medium
Dynamic update: per-broker
Percentage of random jitter added to the renewal time.

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: medium
Dynamic update: per-broker
Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler.

sasl.login.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Importance: medium
Dynamic update: per-broker
The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Importance: medium
Dynamic update: per-broker

The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.factor

Type: double
Default: 0.8
Importance: medium
Dynamic update: per-broker
Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.jitter

Type: double
Default: 0.05
Importance: medium
Dynamic update: per-broker
The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.mechanism.inter.broker.protocol

Type: string
Default: GSSAPI
Importance: medium
Dynamic update: per-broker
SASL mechanism used for inter-broker communication. Default is GSSAPI.

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
Dynamic update: read-only
The OAuth/OIDC provider URL from which the provider's JWKS (JSON Web Key Set) can be retrieved. The URL can be HTTP(S)-based or file-based. If the URL is HTTP(S)-based, the JWKS data will be retrieved from the OAuth/OIDC provider via the configured URL on broker startup. All then-current keys will be cached on the broker for incoming requests. If an authentication request is received for a JWT that includes a "kid" header claim value that isn't yet in the cache, the JWKS endpoint will be queried again on demand. However, the broker polls the URL every sasl.oauthbearer.jwks.endpoint.refresh.ms milliseconds to refresh the cache with any forthcoming keys before any JWT requests that include them are received. If the URL is file-based, the broker will load the JWKS file from a configured location on startup. In the event that the JWT includes a "kid" header value that isn't in the JWKS file, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
Dynamic update: read-only
The URL for the OAuth/OIDC identity provider. If the URL is HTTP(S)-based, it is the issuer's token

endpoint URL to which requests will be made to login based on the configuration in sasl.jaas.config. If the URL is file-based, it specifies a file containing an access token (in JWT serialized form) issued by the OAuth/OIDC identity provider to use for authorization.

sasl.server.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
The fully qualified name of a SASL server callback handler class that implements the AuthenticateCallbackHandler interface. Server callback handlers must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.plain.sasl.server.callback.handler.class=com.example.CustomPlainCallbackHandler

sasl.server.max.receive.size

Type: int
Default: 524288
Importance: medium
Dynamic update: read-only
The maximum receive size allowed before and during initial SASL authentication. Default receive size is 512KB. GSSAPI limits requests to 64K, but we allow upto 512KB by default for custom SASL mechanisms. In practice, PLAIN, SCRAM and OAUTH mechanisms can use much smaller limits.

security.inter.broker.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Dynamic update: read-only
Security protocol used to communicate between brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL. It is an error to set this and inter.broker.listener.name properties at the same time.

socket.connection.setup.timeout.max.ms

Type: long
Default: 30000 (30 seconds)
Importance: medium
Dynamic update: read-only
The maximum amount of time the client will wait for the socket connection to be established. The connection setup timeout will increase exponentially for each consecutive connection failure up to this maximum. To avoid connection storms, a randomization factor of 0.2 will be applied to the timeout resulting in a random range between 20% below and 20% above the computed value.

socket.connection.setup.timeout.ms

Type: long
Default: 10000 (10 seconds)
Importance: medium
Dynamic update: read-only
The amount of time the client will wait for the socket connection to be established. If the connection is not built before the timeout elapses, clients will close the socket channel.

socket.listen.backlog.size

Type: int
Default: 50
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
The maximum number of pending connections on the socket. In Linux, you may also need to configure **somaxconn** and **tcp_max_syn_backlog** kernel parameters accordingly to make the configuration takes effect.

ssl.cipher.suites

Type: list
Default: ""
Importance: medium
Dynamic update: per-broker
A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.

ssl.client.auth

Type: string
Default: none
Valid Values: [required, requested, none]
Importance: medium
Dynamic update: per-broker
Configures kafka broker to request client authentication. The following settings are common:

- **ssl.client.auth=required** If set to required client authentication is required.
- **ssl.client.auth=requested** This means client authentication is optional. unlike required, if this option is set client can choose not to provide authentication information about itself
- **ssl.client.auth=none** This means client authentication is not needed.

ssl.enabled.protocols

Type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
Dynamic update: per-broker
The list of protocols enabled for SSL connections. The default is 'TLSv1.2,TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. With the default value for Java 11, clients and servers will prefer TLSv1.3 if both support it and fallback to TLSv1.2 otherwise (assuming both support at least TLSv1.2). This default should be fine for most cases. Also see the config documentation for **ssl.protocol**.

ssl.key.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
The password of the private key in the key store file or the PEM key specified in `ssl.keystore.key'.

ssl.keymanager.algorithm

Type: string
Default: SunX509

Importance: medium
Dynamic update: per-broker
The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
Certificate chain in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with a list of X.509 certificates.

ssl.keystore.key

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
Private key in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with PKCS#8 keys. If the key is encrypted, key password must be specified using 'ssl.key.password'.

ssl.keystore.location

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
The location of the key store file. This is optional for client and can be used for two-way authentication for client.

ssl.keystore.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
The store password for the key store file. This is optional for client and only needed if 'ssl.keystore.location' is configured. Key store password is not supported for PEM format.

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
Dynamic update: per-broker
The file format of the key store file. This is optional for client. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

ssl.protocol

Type: string
Default: TLSv1.3
Importance: medium
Dynamic update: per-broker
The SSL protocol used to generate the SSLContext. The default is 'TLSv1.3' when running with Java

11 or newer, 'TLSv1.2' otherwise. This value should be fine for most use cases. Allowed values in recent JVMs are 'TLSv1.2' and 'TLSv1.3'. 'TLS', 'TLSv1.1', 'SSL', 'SSLv2' and 'SSLv3' may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities. With the default value for this config and 'ssl.enabled.protocols', clients will downgrade to 'TLSv1.2' if the server does not support 'TLSv1.3'. If this config is set to 'TLSv1.2', clients will not use 'TLSv1.3' even if it is one of the values in ssl.enabled.protocols and the server only supports 'TLSv1.3'.

ssl.provider

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: medium
Dynamic update: per-broker
The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.

ssl.truststore.certificates

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
Trusted certificates in the format specified by 'ssl.truststore.type'. Default SSL engine factory supports only PEM format with X.509 certificates.

ssl.truststore.location

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
The location of the trust store file.

ssl.truststore.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
The password for the trust store file. If a password is not set, trust store file configured will still be used, but integrity checking is disabled. Trust store password is not supported for PEM format.

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
Dynamic update: per-broker

The file format of the trust store file. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

zookeeper.clientCnxnSocket

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Typically set to **org.apache.zookeeper.ClientCnxnSocketNetty** when using TLS connectivity to ZooKeeper. Overrides any explicit value set via the same-named **zookeeper.clientCnxnSocket** system property.

zookeeper.ssl.client.enable

Type: boolean
Default: false
Importance: medium
Dynamic update: read-only
Set client to use TLS when connecting to ZooKeeper. An explicit value overrides any value set via the **zookeeper.client.secure** system property (note the different name). Defaults to false if neither is set; when true, **zookeeper.clientCnxnSocket** must be set (typically to **org.apache.zookeeper.ClientCnxnSocketNetty**); other values to set may include **zookeeper.ssl.cipher.suites**, **zookeeper.ssl.crl.enable**, **zookeeper.ssl.enabled.protocols**, **zookeeper.ssl.endpoint.identification.algorithm**, **zookeeper.ssl.keystore.location**, **zookeeper.ssl.keystore.password**, **zookeeper.ssl.keystore.type**, **zookeeper.ssl.ocsp.enable**, **zookeeper.ssl.protocol**, **zookeeper.ssl.truststore.location**, **zookeeper.ssl.truststore.password**, **zookeeper.ssl.truststore.type**.

zookeeper.ssl.keystore.location

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Keystore location when using a client-side certificate with TLS connectivity to ZooKeeper. Overrides any explicit value set via the **zookeeper.ssl.keyStore.location** system property (note the camelCase).

zookeeper.ssl.keystore.password

Type: password
Default: null
Importance: medium
Dynamic update: read-only
Keystore password when using a client-side certificate with TLS connectivity to ZooKeeper. Overrides any explicit value set via the **zookeeper.ssl.keyStore.password** system property (note the camelCase). Note that ZooKeeper does not support a key password different from the keystore password, so be sure to set the key password in the keystore to be identical to the keystore password; otherwise the connection attempt to Zookeeper will fail.

zookeeper.ssl.keystore.type

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Keystore type when using a client-side certificate with TLS connectivity to ZooKeeper. Overrides any

explicit value set via the **zookeeper.ssl.keyStore.type** system property (note the camelCase). The default value of **null** means the type will be auto-detected based on the filename extension of the keystore.

zookeeper.ssl.truststore.location

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Truststore location when using TLS connectivity to ZooKeeper. Overrides any explicit value set via the **zookeeper.ssl.trustStore.location** system property (note the camelCase).

zookeeper.ssl.truststore.password

Type: password
Default: null
Importance: medium
Dynamic update: read-only
Truststore password when using TLS connectivity to ZooKeeper. Overrides any explicit value set via the **zookeeper.ssl.trustStore.password** system property (note the camelCase).

zookeeper.ssl.truststore.type

Type: string
Default: null
Importance: medium
Dynamic update: read-only
Truststore type when using TLS connectivity to ZooKeeper. Overrides any explicit value set via the **zookeeper.ssl.trustStore.type** system property (note the camelCase). The default value of **null** means the type will be auto-detected based on the filename extension of the truststore.

alter.config.policy.class.name

Type: class
Default: null
Importance: low
Dynamic update: read-only
The alter configs policy class that should be used for validation. The class should implement the **org.apache.kafka.server.policy.AlterConfigPolicy** interface.

alter.log.dirs.replication.quota.window.num

Type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The number of samples to retain in memory for alter log dirs replication quotas.

alter.log.dirs.replication.quota.window.size.seconds

Type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The time span of each sample for alter log dirs replication quotas.

authorizer.class.name

Type: string
Default: ""
Valid Values: non-null string
Importance: low
Dynamic update: read-only
The fully qualified name of a class that implements **org.apache.kafka.server.authorizer.Authorizer** interface, which is used by the broker for authorization.

client.quota.callback.class

Type: class
Default: null
Importance: low
Dynamic update: read-only
The fully qualified name of a class that implements the ClientQuotaCallback interface, which is used to determine quota limits applied to client requests. By default, the <user> and <client-id> quotas that are stored in ZooKeeper are applied. For any given request, the most specific quota that matches the user principal of the session and the client-id of the request is applied.

connection.failed.authentication.delay.ms

Type: int
Default: 100
Valid Values: [0,...]
Importance: low
Dynamic update: read-only
Connection close delay on failed authentication: this is the time (in milliseconds) by which connection close will be delayed on authentication failure. This must be configured to be less than connections.max.idle.ms to prevent connection timeout.

controller.quorum.retry.backoff.ms

Type: int
Default: 20
Importance: low
Dynamic update: read-only
The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

controller.quota.window.num

Type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The number of samples to retain in memory for controller mutation quotas.

controller.quota.window.size.seconds

Type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The time span of each sample for controller mutations quotas.

create.topic.policy.class.name

Type: class
Default: null
Importance: low
Dynamic update: read-only
The create topic policy class that should be used for validation. The class should implement the **org.apache.kafka.server.policy.CreateTopicPolicy** interface.

delegation.token.expiry.check.interval.ms

Type: long
Default: 3600000 (1 hour)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
Scan interval to remove expired delegation tokens.

kafka.metrics.polling.interval.secs

Type: int
Default: 10
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The metrics polling interval (in seconds) which can be used in kafka.metrics.reporters implementations.

kafka.metrics.reporters

Type: list
Default: ""
Importance: low
Dynamic update: read-only
A list of classes to use as Yammer metrics custom reporters. The reporters should implement **kafka.metrics.KafkaMetricsReporter** trait. If a client wants to expose JMX operations on a custom reporter, the custom reporter needs to additionally implement an MBean trait that extends **kafka.metrics.KafkaMetricsReporterMBean** trait so that the registered MBean is compliant with the standard MBean convention.

listener.security.protocol.map

Type: string
Default:
PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
Importance: low
Dynamic update: per-broker
Map between listener names and security protocols. This must be defined for the same security protocol to be usable in more than one port or IP. For example, internal and external traffic can be separated even if SSL is required for both. Concretely, the user could define listeners with names INTERNAL and EXTERNAL and this property as: **INTERNAL:SSL,EXTERNAL:SSL**. As shown, key and value are separated by a colon and map entries are separated by commas. Each listener name should only appear once in the map. Different security (SSL and SASL) settings can be configured for each listener by adding a normalised prefix (the listener name is lowercased) to the config name. For example, to set a different keystore for the INTERNAL listener, a config with name **listener.name.internal.ssl.keystore.location** would be set. If the config for the listener name is not

set, the config will fallback to the generic config (i.e. **ssl.keystore.location**). Note that in KRaft a default mapping from the listener names defined by **controller.listener.names** to PLAINTEXT is assumed if no explicit mapping is provided and no other security protocol is in use.

log.message.downconversion.enable

Type: boolean
Default: true
Importance: low
Dynamic update: cluster-wide
This configuration controls whether down-conversion of message formats is enabled to satisfy consume requests. When set to **false**, broker will not perform down-conversion for consumers expecting an older message format. The broker responds with **UNSUPPORTED_VERSION** error for consume requests from such older clients. This configurationdoes not apply to any message format conversion that might be required for replication to followers.

metadata.max.idle.interval.ms

Type: int
Default: 500
Valid Values: [0,...]
Importance: low
Dynamic update: read-only
This configuration controls how often the active controller should write no-op records to the metadata partition. If the value is 0, no-op records are not appended to the metadata partition. The default value is 500.

metric.reporters

Type: list
Default: ""
Importance: low
Dynamic update: cluster-wide
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string
Default: INFO
Importance: low
Dynamic update: read-only
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)

Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The window of time a metrics sample is computed over.

password.encoder.cipher.algorithm

Type: string
Default: AES/CBC/PKCS5Padding
Importance: low
Dynamic update: read-only
The Cipher algorithm used for encoding dynamically configured passwords.

password.encoder.iterations

Type: int
Default: 4096
Valid Values: [1024,...]
Importance: low
Dynamic update: read-only
The iteration count used for encoding dynamically configured passwords.

password.encoder.key.length

Type: int
Default: 128
Valid Values: [8,...]
Importance: low
Dynamic update: read-only
The key length used for encoding dynamically configured passwords.

password.encoder.keyfactory.algorithm

Type: string
Default: null
Importance: low
Dynamic update: read-only
The SecretKeyFactory algorithm used for encoding dynamically configured passwords. Default is PBKDF2WithHmacSHA512 if available and PBKDF2WithHmacSHA1 otherwise.

quota.window.num

Type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The number of samples to retain in memory for client quotas.

quota.window.size.seconds

Type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The time span of each sample for client quotas.

replication.quota.window.num

Type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The number of samples to retain in memory for replication quotas.

replication.quota.window.size.seconds

Type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The time span of each sample for replication quotas.

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the external authentication provider connection timeout. Currently applies only to OAUTHBEARER.

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the external authentication provider read timeout. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the maximum wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the initial wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum

wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
Dynamic update: read-only
The (optional) value in seconds to allow for differences between the time of the OAuth/OIDC identity provider and the broker.

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
Dynamic update: read-only
The (optional) comma-delimited setting for the broker to use to verify that the JWT was issued for one of the expected audiences. The JWT will be inspected for the standard OAuth "aud" claim and if this value is set, the broker will match the value from JWT's "aud" claim to see if there is an exact match. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
Dynamic update: read-only
The (optional) setting for the broker to use to verify that the JWT was created by the expected issuer. The JWT will be inspected for the standard OAuth "iss" claim and if this value is set, the broker will match it exactly against what is in the JWT's "iss" claim. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the broker to wait between refreshing its JWKS (JSON Web Key Set) cache that contains the keys to verify the signature of the JWT.

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the maximum wait between attempts to retrieve the JWKS (JSON Web Key Set) from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms

Type: long

Default: 100
Importance: low
Dynamic update: read-only
The (optional) value in milliseconds for the initial wait between JWKS (JSON Web Key Set) retrieval attempts from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
Dynamic update: read-only
The OAuth claim for the scope is often named "scope", but this (optional) setting can provide a different name to use for the scope included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
Dynamic update: read-only
The OAuth claim for the subject is often named "sub", but this (optional) setting can provide a different name to use for the subject included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

security.providers

Type: string
Default: null
Importance: low
Dynamic update: read-only
A list of configurable creator classes each returning a provider implementing security algorithms. These classes should implement the **org.apache.kafka.common.security.auth.SecurityProviderCreator** interface.

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
Dynamic update: per-broker
The endpoint identification algorithm to validate server hostname using server certificate.

ssl.engine.factory.class

Type: class
Default: null
Importance: low
Dynamic update: per-broker
The class of type org.apache.kafka.common.security.auth.SslEngineFactory to provide SSLEngine objects. Default value is org.apache.kafka.common.security.ssl.DefaultSslEngineFactory.

ssl.principal.mapping.rules

Type: string
Default: DEFAULT
Importance: low
Dynamic update: read-only
A list of rules for mapping from distinguished name from the client certificate to short name. The rules are evaluated in order and the first rule that matches a principal name is used to map it to a short name. Any later rules in the list are ignored. By default, distinguished name of the X.500 certificate will be the principal. For more details on the format please see security authorization and acls. Note that this configuration is ignored if an extension of KafkaPrincipalBuilder is provided by the **principal.builder.class** configuration.

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
Dynamic update: per-broker
The SecureRandom PRNG implementation to use for SSL cryptography operations.

transaction.abort.timed.out.transaction.cleanup.interval.ms

Type: int
Default: 10000 (10 seconds)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The interval at which to rollback transactions that have timed out.

transaction.remove.expired.transaction.cleanup.interval.ms

Type: int
Default: 3600000 (1 hour)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
The interval at which to remove transactions that have expired due to **transactional.id.expiration.ms** passing.

zookeeper.ssl.cipher.suites

Type: list
Default: null
Importance: low
Dynamic update: read-only
Specifies the enabled cipher suites to be used in ZooKeeper TLS negotiation (csv). Overrides any explicit value set via the **zookeeper.ssl.ciphersuites** system property (note the single word "ciphersuites"). The default value of **null** means the list of enabled cipher suites is determined by the Java runtime being used.

zookeeper.ssl.crl.enable

Type: boolean
Default: false
Importance: low
Dynamic update: read-only
Specifies whether to enable Certificate Revocation List in the ZooKeeper TLS protocols. Overrides any explicit value set via the **zookeeper.ssl.crl** system property (note the shorter name).

zookeeper.ssl.enabled.protocols

Type: list
Default: null
Importance: low
Dynamic update: read-only
Specifies the enabled protocol(s) in ZooKeeper TLS negotiation (csv). Overrides any explicit value set via the **zookeeper.ssl.enabledProtocols** system property (note the camelCase). The default value of **null** means the enabled protocol will be the value of the **zookeeper.ssl.protocol** configuration property.

zookeeper.ssl.endpoint.identification.algorithm

Type: string
Default: HTTPS
Importance: low
Dynamic update: read-only
Specifies whether to enable hostname verification in the ZooKeeper TLS negotiation process, with (case-insensitively) "https" meaning ZooKeeper hostname verification is enabled and an explicit blank value meaning it is disabled (disabling it is only recommended for testing purposes). An explicit value overrides any "true" or "false" value set via the **zookeeper.ssl.hostnameVerification** system property (note the different name and values; true implies https and false implies blank).

zookeeper.ssl.ocsp.enable

Type: boolean
Default: false
Importance: low
Dynamic update: read-only
Specifies whether to enable Online Certificate Status Protocol in the ZooKeeper TLS protocols. Overrides any explicit value set via the **zookeeper.ssl.ocsp** system property (note the shorter name).

zookeeper.ssl.protocol

Type: string
Default: TLSv1.2
Importance: low
Dynamic update: read-only
Specifies the protocol to be used in ZooKeeper TLS negotiation. An explicit value overrides any value set via the same-named **zookeeper.ssl.protocol** system property.

CHAPTER 2. TOPIC CONFIGURATION PROPERTIES

cleanup.policy

Type: list
Default: delete
Valid Values: [compact, delete]
Server Default Property: log.cleanup.policy
Importance: medium
This config designates the retention policy to use on log segments. The "delete" policy (which is the default) will discard old segments when their retention time or size limit has been reached. The "compact" policy will enable log compaction, which retains the latest value for each key. It is also possible to specify both policies in a comma-separated list (e.g. "delete,compact"). In this case, old segments will be discarded per the retention time and size configuration, while retained segments will be compacted.

compression.type

Type: string
Default: producer
Valid Values: [uncompressed, zstd, lz4, snappy, gzip, producer]
Server Default Property: compression.type
Importance: medium
Specify the final compression type for a given topic. This configuration accepts the standard compression codecs ('gzip', 'snappy', 'lz4', 'zstd'). It additionally accepts 'uncompressed' which is equivalent to no compression; and 'producer' which means retain the original compression codec set by the producer.

delete.retention.ms

Type: long
Default: 86400000 (1 day)
Valid Values: [0,...]
Server Default Property: log.cleaner.delete.retention.ms
Importance: medium
The amount of time to retain delete tombstone markers for log compacted topics. This setting also gives a bound on the time in which a consumer must complete a read if they begin from offset 0 to ensure that they get a valid snapshot of the final stage (otherwise delete tombstones may be collected before they complete their scan).

file.delete.delay.ms

Type: long
Default: 60000 (1 minute)
Valid Values: [0,...]
Server Default Property: log.segment.delete.delay.ms
Importance: medium
The time to wait before deleting a file from the filesystem.

flush.messages

Type: long
Default: 9223372036854775807
Valid Values: [1,...]
Server Default Property: log.flush.interval.messages
Importance: medium
This setting allows specifying an interval at which we will force an fsync of data written to the log. For

example if this was set to 1 we would fsync after every message; if it were 5 we would fsync after every five messages. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient. This setting can be overridden on a per-topic basis (see the per-topic configuration section).

flush.ms

Type: long
Default: 9223372036854775807
Valid Values: [0,...]
Server Default Property: log.flush.interval.ms
Importance: medium
This setting allows specifying a time interval at which we will force an fsync of data written to the log. For example if this was set to 1000 we would fsync after 1000 ms had passed. In general we recommend you not set this and use replication for durability and allow the operating system's background flush capabilities as it is more efficient.

follower.replication.throttled.replicas

Type: list
Default: ""
Valid Values: [partitionId]:[brokerId],[partitionId]:[brokerId],...
Server Default Property: follower.replication.throttled.replicas
Importance: medium
A list of replicas for which log replication should be throttled on the follower side. The list should describe a set of replicas in the form [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... or alternatively the wildcard '*' can be used to throttle all replicas for this topic.

index.interval.bytes

Type: int
Default: 4096 (4 kibibytes)
Valid Values: [0,...]
Server Default Property: log.index.interval.bytes
Importance: medium
This setting controls how frequently Kafka adds an index entry to its offset index. The default setting ensures that we index a message roughly every 4096 bytes. More indexing allows reads to jump closer to the exact position in the log but makes the index larger. You probably don't need to change this.

leader.replication.throttled.replicas

Type: list
Default: ""
Valid Values: [partitionId]:[brokerId],[partitionId]:[brokerId],...
Server Default Property: leader.replication.throttled.replicas
Importance: medium
A list of replicas for which log replication should be throttled on the leader side. The list should describe a set of replicas in the form [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... or alternatively the wildcard '*' can be used to throttle all replicas for this topic.

max.compaction.lag.ms

Type: long
Default: 9223372036854775807
Valid Values: [1,...]
Server Default Property: log.cleaner.max.compaction.lag.ms
Importance: medium

The maximum time a message will remain ineligible for compaction in the log. Only applicable for logs that are being compacted.

max.message.bytes

Type: int
Default: 1048588
Valid Values: [0,...]
Server Default Property: message.max.bytes
Importance: medium
The largest record batch size allowed by Kafka (after compression if compression is enabled). If this is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be increased so that they can fetch record batches this large. In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case.

message.format.version

Type: string
Default: 3.0-IV1
Valid Values: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3]
Server Default Property: log.message.format.version
Importance: medium
[DEPRECATED] Specify the message format version the broker will use to append messages to the logs. The value of this config is always assumed to be **3.0** if **inter.broker.protocol.version** is 3.0 or higher (the actual config value is ignored). Otherwise, the value should be a valid ApiVersion. Some examples are: 0.10.0, 1.1, 2.8, 3.0. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or equal than the specified version. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.

message.timestamp.difference.max.ms

Type: long
Default: 9223372036854775807
Valid Values: [0,...]
Server Default Property: log.message.timestamp.difference.max.ms
Importance: medium
The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if message.timestamp.type=LogAppendTime.

message.timestamp.type

Type: string
Default: CreateTime
Valid Values: [CreateTime, LogAppendTime]
Server Default Property: log.message.timestamp.type
Importance: medium
Define whether the timestamp in the message is message create time or log append time. The value should be either **CreateTime** or **LogAppendTime**.

min.cleanable.dirty.ratio

Type: double
Default: 0.5
Valid Values: [0,...,1]
Server Default Property: log.cleaner.min.cleanable.ratio
Importance: medium
This configuration controls how frequently the log compactor will attempt to clean the log (assuming log compaction is enabled). By default we will avoid cleaning a log where more than 50% of the log has been compacted. This ratio bounds the maximum space wasted in the log by duplicates (at 50% at most 50% of the log could be duplicates). A higher ratio will mean fewer, more efficient cleanings but will mean more wasted space in the log. If the max.compaction.lag.ms or the min.compaction.lag.ms configurations are also specified, then the log compactor considers the log to be eligible for compaction as soon as either: (i) the dirty ratio threshold has been met and the log has had dirty (uncompacted) records for at least the min.compaction.lag.ms duration, or (ii) if the log has had dirty (uncompacted) records for at most the max.compaction.lag.ms period.

min.compaction.lag.ms

Type: long
Default: 0
Valid Values: [0,...]
Server Default Property: log.cleaner.min.compaction.lag.ms
Importance: medium
The minimum time a message will remain uncompacted in the log. Only applicable for logs that are being compacted.

min.insync.replicas

Type: int
Default: 1
Valid Values: [1,...]
Server Default Property: min.insync.replicas
Importance: medium
When a producer sets acks to "all" (or "-1"), this configuration specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, **min.insync.replicas** and **acks** allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set **min.insync.replicas** to 2, and produce with **acks** of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.

preallocate

Type: boolean
Default: false
Server Default Property: log.preallocate
Importance: medium
True if we should preallocate the file on disk when creating a new log segment.

retention.bytes

Type: long
Default: -1
Server Default Property: log.retention.bytes
Importance: medium
This configuration controls the maximum size a partition (which consists of log segments) can grow to before we will discard old log segments to free up space if we are using the "delete" retention

policy. By default there is no size limit only a time limit. Since this limit is enforced at the partition level, multiply it by the number of partitions to compute the topic retention in bytes.

retention.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [-1,...]
Server Default Property: log.retention.ms
Importance: medium
This configuration controls the maximum time we will retain a log before we will discard old log segments to free up space if we are using the "delete" retention policy. This represents an SLA on how soon consumers must read their data. If set to -1, no time limit is applied.

segment.bytes

Type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [14,...]
Server Default Property: log.segment.bytes
Importance: medium
This configuration controls the segment file size for the log. Retention and cleaning is always done a file at a time so a larger segment size means fewer files but less granular control over retention.

segment.index.bytes

Type: int
Default: 10485760 (10 mebibytes)
Valid Values: [4,...]
Server Default Property: log.index.size.max.bytes
Importance: medium
This configuration controls the size of the index that maps offsets to file positions. We preallocate this index file and shrink it only after log rolls. You generally should not need to change this setting.

segment.jitter.ms

Type: long
Default: 0
Valid Values: [0,...]
Server Default Property: log.roll.jitter.ms
Importance: medium
The maximum random jitter subtracted from the scheduled segment roll time to avoid thundering herds of segment rolling.

segment.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [1,...]
Server Default Property: log.roll.ms
Importance: medium
This configuration controls the period of time after which Kafka will force the log to roll even if the segment file isn't full to ensure that retention can delete or compact old data.

unclean.leader.election.enable

Type: boolean
Default: false

Server Default Property: unclean.leader.election.enable
Importance: medium
Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss.

message.downconversion.enable

Type: boolean
Default: true
Server Default Property: log.message.downconversion.enable
Importance: low
This configuration controls whether down-conversion of message formats is enabled to satisfy consume requests. When set to **false**, broker will not perform down-conversion for consumers expecting an older message format. The broker responds with **UNSUPPORTED_VERSION** error for consume requests from such older clients. This configurationdoes not apply to any message format conversion that might be required for replication to followers.

CHAPTER 3. CONSUMER CONFIGURATION PROPERTIES

key.deserializer

Type: class
Importance: high
Deserializer class for key that implements the **org.apache.kafka.common.serialization.Deserializer** interface.

value.deserializer

Type: class
Importance: high
Deserializer class for value that implements the **org.apache.kafka.common.serialization.Deserializer** interface.

bootstrap.servers

Type: list
Default: ""
Valid Values: non-null string
Importance: high
A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form **host1:port1,host2:port2,...**. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

fetch.min.bytes

Type: int
Default: 1
Valid Values: [0,...]
Importance: high
The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request. The default setting of 1 byte means that fetch requests are answered as soon as a single byte of data is available or the fetch request times out waiting for data to arrive. Setting this to something greater than 1 will cause the server to wait for larger amounts of data to accumulate which can improve server throughput a bit at the cost of some additional latency.

group.id

Type: string
Default: null
Importance: high
A unique string that identifies the consumer group this 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.

heartbeat.interval.ms

Type: int
Default: 3000 (3 seconds)
Importance: high
The expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the consumer's session stays active and to

facilitate rebalancing when new consumers join or leave the group. The value must be set lower than **session.timeout.ms**, but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.

max.partition.fetch.bytes

Type: int
Default: 1048576 (1 mebibyte)
Valid Values: [0,...]
Importance: high
The maximum amount of data per-partition the server will return. Records are fetched in batches by the consumer. If the first record batch in the first non-empty partition of the fetch is larger than this limit, the batch will still be returned to ensure that the consumer can make progress. The maximum record batch size accepted by the broker is defined via **message.max.bytes** (broker config) or **max.message.bytes** (topic config). See fetch.max.bytes for limiting the consumer request size.

session.timeout.ms

Type: int
Default: 45000 (45 seconds)
Importance: high
The timeout used to detect client failures when using Kafka's group management facility. The client sends periodic heartbeats to indicate its liveness to the broker. If no heartbeats are received by the broker before the expiration of this session timeout, then the broker will remove this client from the group and initiate a rebalance. Note that the value must be in the allowable range as configured in the broker configuration by **group.min.session.timeout.ms** and **group.max.session.timeout.ms**.

ssl.key.password

Type: password
Default: null
Importance: high
The password of the private key in the key store file or the PEM key specified in `ssl.keystore.key'.

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
Certificate chain in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with a list of X.509 certificates.

ssl.keystore.key

Type: password
Default: null
Importance: high
Private key in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with PKCS#8 keys. If the key is encrypted, key password must be specified using 'ssl.key.password'.

ssl.keystore.location

Type: string
Default: null
Importance: high
The location of the key store file. This is optional for client and can be used for two-way authentication for client.

ssl.keystore.password

Type: password
Default: null
Importance: high
The store password for the key store file. This is optional for client and only needed if 'ssl.keystore.location' is configured. Key store password is not supported for PEM format.

ssl.truststore.certificates

Type: password
Default: null
Importance: high
Trusted certificates in the format specified by 'ssl.truststore.type'. Default SSL engine factory supports only PEM format with X.509 certificates.

ssl.truststore.location

Type: string
Default: null
Importance: high
The location of the trust store file.

ssl.truststore.password

Type: password
Default: null
Importance: high
The password for the trust store file. If a password is not set, trust store file configured will still be used, but integrity checking is disabled. Trust store password is not supported for PEM format.

allow.auto.create.topics

Type: boolean
Default: true
Importance: medium
Allow automatic topic creation on the broker when subscribing to or assigning a topic. A topic being subscribed to will be automatically created only if the broker allows for it using **auto.create.topics.enable** broker configuration. This configuration must be set to **false** when using brokers older than 0.11.0.

auto.offset.reset

Type: string
Default: latest
Valid Values: [latest, earliest, none]
Importance: medium
What to do when there is no initial offset in Kafka or if the current offset does not exist any more on the server (e.g. because that data has been deleted):

- earliest: automatically reset the offset to the earliest offset
- latest: automatically reset the offset to the latest offset
- none: throw exception to the consumer if no previous offset is found for the consumer's group
- anything else: throw exception to the consumer.

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
Controls how the client uses DNS lookups. If set to **use_all_dns_ips**, connect to each returned IP address in sequence until a successful connection is established. After a disconnection, the next IP is used. Once all IPs have been used once, the client resolves the IP(s) from the hostname again (both the JVM and the OS cache DNS name lookups, however). If set to **resolve_canonical_bootstrap_servers_only**, resolve each bootstrap address into a list of canonical names. After the bootstrap phase, this behaves the same as **use_all_dns_ips**.

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
Close idle connections after the number of milliseconds specified by this config.

default.api.timeout.ms

Type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: medium
Specifies the timeout (in milliseconds) for client APIs. This configuration is used as the default timeout for all client operations that do not specify a **timeout** parameter.

enable.auto.commit

Type: boolean
Default: true
Importance: medium
If true the consumer's offset will be periodically committed in the background.

exclude.internal.topics

Type: boolean
Default: true
Importance: medium
Whether internal topics matching a subscribed pattern should be excluded from the subscription. It is always possible to explicitly subscribe to an internal topic.

fetch.max.bytes

Type: int
Default: 52428800 (50 mebibytes)
Valid Values: [0,...]
Importance: medium
The maximum amount of data the server should return for a fetch request. Records are fetched in batches by the consumer, and if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that the consumer can make progress. As such, this is not a absolute maximum. The maximum record batch size accepted by the broker is defined via **message.max.bytes** (broker config) or **max.message.bytes** (topic config). Note that the consumer performs multiple fetches in parallel.

group.instance.id

Type: string
Default: null
Valid Values: non-empty string
Importance: medium
A unique identifier of the consumer instance provided by the end user. Only non-empty strings are permitted. If set, the consumer is treated as a static member, which means that only one instance with this ID is allowed in the consumer group at any time. This can be used in combination with a larger session timeout to avoid group rebalances caused by transient unavailability (e.g. process restarts). If not set, the consumer will join the group as a dynamic member, which is the traditional behavior.

isolation.level

Type: string
Default: read_uncommitted
Valid Values: [read_committed, read_uncommitted]
Importance: medium
Controls how to read messages written transactionally. If set to **read_committed**, consumer.poll() will only return transactional messages which have been committed. If set to **read_uncommitted** (the default), consumer.poll() will return all messages, even transactional messages which have been aborted. Non-transactional messages will be returned unconditionally in either mode.

Messages will always be returned in offset order. Hence, in **read_committed** mode, consumer.poll() will only return messages up to the last stable offset (LSO), which is the one less than the offset of the first open transaction. In particular any messages appearing after messages belonging to ongoing transactions will be withheld until the relevant transaction has been completed. As a result, **read_committed** consumers will not be able to read up to the high watermark when there are in flight transactions.

> Further, when in `read_committed` the seekToEnd method will return the LSO
> .

max.poll.interval.ms

Type: int
Default: 300000 (5 minutes)
Valid Values: [1,...]
Importance: medium
The maximum delay between invocations of poll() when using consumer group management. This places an upper bound on the amount of time that the consumer can be idle before fetching more records. If poll() is not called before expiration of this timeout, then the consumer is considered failed and the group will rebalance in order to reassign the partitions to another member. For consumers using a non-null **group.instance.id** which reach this timeout, partitions will not be immediately reassigned. Instead, the consumer will stop sending heartbeats and partitions will be reassigned after expiration of **session.timeout.ms**. This mirrors the behavior of a static consumer which has shutdown.

max.poll.records

Type: int
Default: 500
Valid Values: [1,...]
Importance: medium
The maximum number of records returned in a single call to poll(). Note, that **max.poll.records** does not impact the underlying fetching behavior. The consumer will cache the records from each fetch request and returns them incrementally from each poll.

partition.assignment.strategy

Type: list
Default: class org.apache.kafka.clients.consumer.RangeAssignor,class org.apache.kafka.clients.consumer.CooperativeStickyAssignor
Valid Values: non-null string
Importance: medium
A list of class names or class types, ordered by preference, of supported partition assignment strategies that the client will use to distribute partition ownership amongst consumer instances when group management is used. Available options are:

- **org.apache.kafka.clients.consumer.RangeAssignor**: Assigns partitions on a per-topic basis.
- **org.apache.kafka.clients.consumer.RoundRobinAssignor**: Assigns partitions to consumers in a round-robin fashion.
- **org.apache.kafka.clients.consumer.StickyAssignor**: Guarantees an assignment that is maximally balanced while preserving as many existing partition assignments as possible.
- **org.apache.kafka.clients.consumer.CooperativeStickyAssignor**: Follows the same StickyAssignor logic, but allows for cooperative rebalancing.
  The default assignor is [RangeAssignor, CooperativeStickyAssignor], which will use the RangeAssignor by default, but allows upgrading to the CooperativeStickyAssignor with just a single rolling bounce that removes the RangeAssignor from the list.

  Implementing the **org.apache.kafka.clients.consumer.ConsumerPartitionAssignor** interface allows you to plug in a custom assignment strategy.

receive.buffer.bytes

Type: int
Default: 65536 (64 kibibytes)
Valid Values: [-1,...]
Importance: medium
The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.

request.timeout.ms

Type: int
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: medium
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.

sasl.jaas.config

Type: password
Default: null
Importance: medium
JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here. The format for the value is: **loginModuleClass controlFlag (optionName=optionValue)*;**. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;.

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler.

sasl.login.class

Type: class
Default: null
Importance: medium
The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
The OAuth/OIDC provider URL from which the provider's JWKS (JSON Web Key Set) can be retrieved. The URL can be HTTP(S)-based or file-based. If the URL is HTTP(S)-based, the JWKS data will be retrieved from the OAuth/OIDC provider via the configured URL on broker startup. All then-current keys will be cached on the broker for incoming requests. If an authentication request is received for a JWT that includes a "kid" header claim value that isn't yet in the cache, the JWKS endpoint will be queried again on demand. However, the broker polls the URL every sasl.oauthbearer.jwks.endpoint.refresh.ms milliseconds to refresh the cache with any forthcoming

keys before any JWT requests that include them are received. If the URL is file-based, the broker will load the JWKS file from a configured location on startup. In the event that the JWT includes a "kid" header value that isn't in the JWKS file, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
The URL for the OAuth/OIDC identity provider. If the URL is HTTP(S)-based, it is the issuer's token endpoint URL to which requests will be made to login based on the configuration in sasl.jaas.config. If the URL is file-based, it specifies a file containing an access token (in JWT serialized form) issued by the OAuth/OIDC identity provider to use for authorization.

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

send.buffer.bytes

Type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.

socket.connection.setup.timeout.max.ms

Type: long
Default: 30000 (30 seconds)
Importance: medium
The maximum amount of time the client will wait for the socket connection to be established. The connection setup timeout will increase exponentially for each consecutive connection failure up to this maximum. To avoid connection storms, a randomization factor of 0.2 will be applied to the timeout resulting in a random range between 20% below and 20% above the computed value.

socket.connection.setup.timeout.ms

Type: long
Default: 10000 (10 seconds)
Importance: medium
The amount of time the client will wait for the socket connection to be established. If the connection is not built before the timeout elapses, clients will close the socket channel.

ssl.enabled.protocols

Type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
The list of protocols enabled for SSL connections. The default is 'TLSv1.2,TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. With the default value for Java 11, clients and servers will prefer TLSv1.3 if both support it and fallback to TLSv1.2 otherwise (assuming both support at least TLSv1.2).

This default should be fine for most cases. Also see the config documentation for **ssl.protocol**.

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
The file format of the key store file. This is optional for client. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

ssl.protocol

Type: string
Default: TLSv1.3
Importance: medium
The SSL protocol used to generate the SSLContext. The default is 'TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. This value should be fine for most use cases. Allowed values in recent JVMs are 'TLSv1.2' and 'TLSv1.3'. 'TLS', 'TLSv1.1', 'SSL', 'SSLv2' and 'SSLv3' may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities. With the default value for this config and 'ssl.enabled.protocols', clients will downgrade to 'TLSv1.2' if the server does not support 'TLSv1.3'. If this config is set to 'TLSv1.2', clients will not use 'TLSv1.3' even if it is one of the values in ssl.enabled.protocols and the server only supports 'TLSv1.3'.

ssl.provider

Type: string
Default: null
Importance: medium
The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
The file format of the trust store file. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

auto.commit.interval.ms

Type: int
Default: 5000 (5 seconds)
Valid Values: [0,...]
Importance: low
The frequency in milliseconds that the consumer offsets are auto-committed to Kafka if **enable.auto.commit** is set to **true**.

check.crcs

Type: boolean
Default: true
Importance: low
Automatically check the CRC32 of the records consumed. This ensures no on-the-wire or on-disk corruption to the messages occurred. This check adds some overhead, so it may be disabled in cases seeking extreme performance.

client.id

Type: string
Default: ""
Importance: low
An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.

client.rack

Type: string
Default: ""
Importance: low
A rack identifier for this client. This can be any string value which indicates where this client is physically located. It corresponds with the broker config 'broker.rack'.

fetch.max.wait.ms

Type: int
Default: 500
Valid Values: [0,...]
Importance: low
The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy the requirement given by fetch.min.bytes.

interceptor.classes

Type: list
Default: ""
Valid Values: non-null string
Importance: low
A list of classes to use as interceptors. Implementing the **org.apache.kafka.clients.consumer.ConsumerInterceptor** interface allows you to intercept (and possibly mutate) records received by the consumer. By default, there are no interceptors.

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.

metric.reporters

Type: list
Default: ""
Valid Values: non-null string
Importance: low
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]

Importance: low
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The window of time a metrics sample is computed over.

reconnect.backoff.max.ms

Type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.

reconnect.backoff.ms

Type: long
Default: 50
Valid Values: [0,...]
Importance: low
The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

sasl.kerberos.kinit.cmd

Type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit command path.

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
Login thread sleep time between refresh attempts.

sasl.kerberos.ticket.renew.jitter

Type: double
Default: 0.05
Importance: low
Percentage of random jitter added to the renewal time.

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider connection timeout. Currently applies only to OAUTHBEARER.

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider read timeout. Currently applies only to OAUTHBEARER.

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum

exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.factor

Type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.jitter

Type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
The (optional) value in seconds to allow for differences between the time of the OAuth/OIDC identity provider and the broker.

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low

The (optional) comma-delimited setting for the broker to use to verify that the JWT was issued for one of the expected audiences. The JWT will be inspected for the standard OAuth "aud" claim and if this value is set, the broker will match the value from JWT's "aud" claim to see if there is an exact match. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
The (optional) setting for the broker to use to verify that the JWT was created by the expected issuer. The JWT will be inspected for the standard OAuth "iss" claim and if this value is set, the broker will match it exactly against what is in the JWT's "iss" claim. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
The (optional) value in milliseconds for the broker to wait between refreshing its JWKS (JSON Web Key Set) cache that contains the keys to verify the signature of the JWT.

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between attempts to retrieve the JWKS (JSON Web Key Set) from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between JWKS (JSON Web Key Set) retrieval attempts from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
The OAuth claim for the scope is often named "scope", but this (optional) setting can provide a different name to use for the scope included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
The OAuth claim for the subject is often named "sub", but this (optional) setting can provide a different name to use for the subject included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

security.providers

Type: string
Default: null
Importance: low
A list of configurable creator classes each returning a provider implementing security algorithms. These classes should implement the **org.apache.kafka.common.security.auth.SecurityProviderCreator** interface.

ssl.cipher.suites

Type: list
Default: null
Importance: low
A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
The endpoint identification algorithm to validate server hostname using server certificate.

ssl.engine.factory.class

Type: class
Default: null
Importance: low
The class of type org.apache.kafka.common.security.auth.SslEngineFactory to provide SSLEngine objects. Default value is org.apache.kafka.common.security.ssl.DefaultSslEngineFactory.

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
The SecureRandom PRNG implementation to use for SSL cryptography operations.

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.

CHAPTER 4. PRODUCER CONFIGURATION PROPERTIES

key.serializer

Type: class
Importance: high
Serializer class for key that implements the **org.apache.kafka.common.serialization.Serializer** interface.

value.serializer

Type: class
Importance: high
Serializer class for value that implements the **org.apache.kafka.common.serialization.Serializer** interface.

bootstrap.servers

Type: list
Default: ""
Valid Values: non-null string
Importance: high
A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form **host1:port1,host2:port2,...**. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

buffer.memory

Type: long
Default: 33554432
Valid Values: [0,...]
Importance: high
The total bytes of memory the producer can use to buffer records waiting to be sent to the server. If records are sent faster than they can be delivered to the server the producer will block for **max.block.ms** after which it will throw an exception.

This setting should correspond roughly to the total memory the producer will use, but is not a hard bound since not all memory the producer uses is used for buffering. Some additional memory will be used for compression (if compression is enabled) as well as for maintaining in-flight requests.

compression.type

Type: string
Default: none
Valid Values: [none, gzip, snappy, lz4, zstd]
Importance: high
The compression type for all data generated by the producer. The default is none (i.e. no compression). Valid values are **none**, **gzip**, **snappy**, **lz4**, or **zstd**. Compression is of full batches of data, so the efficacy of batching will also impact the compression ratio (more batching means better compression).

retries

Type: int
Default: 2147483647

Valid Values: [0,...,2147483647]
Importance: high
Setting a value greater than zero will cause the client to resend any record whose send fails with a potentially transient error. Note that this retry is no different than if the client resent the record upon receiving the error. Produce requests will be failed before the number of retries has been exhausted if the timeout configured by **delivery.timeout.ms** expires first before successful acknowledgement. Users should generally prefer to leave this config unset and instead use **delivery.timeout.ms** to control retry behavior.

Enabling idempotence requires this config value to be greater than 0. If conflicting configurations are set and idempotence is not explicitly enabled, idempotence is disabled.

Allowing retries while setting **enable.idempotence** to **false** and **max.in.flight.requests.per.connection** to 1 will potentially change the ordering of records because if two batches are sent to a single partition, and the first fails and is retried but the second succeeds, then the records in the second batch may appear first.

ssl.key.password

Type: password
Default: null
Importance: high
The password of the private key in the key store file or the PEM key specified in `ssl.keystore.key'.

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
Certificate chain in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with a list of X.509 certificates.

ssl.keystore.key

Type: password
Default: null
Importance: high
Private key in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with PKCS#8 keys. If the key is encrypted, key password must be specified using 'ssl.key.password'.

ssl.keystore.location

Type: string
Default: null
Importance: high
The location of the key store file. This is optional for client and can be used for two-way authentication for client.

ssl.keystore.password

Type: password
Default: null
Importance: high
The store password for the key store file. This is optional for client and only needed if 'ssl.keystore.location' is configured. Key store password is not supported for PEM format.

ssl.truststore.certificates

Type: password
Default: null
Importance: high
Trusted certificates in the format specified by 'ssl.truststore.type'. Default SSL engine factory supports only PEM format with X.509 certificates.

ssl.truststore.location

Type: string
Default: null
Importance: high
The location of the trust store file.

ssl.truststore.password

Type: password
Default: null
Importance: high
The password for the trust store file. If a password is not set, trust store file configured will still be used, but integrity checking is disabled. Trust store password is not supported for PEM format.

batch.size

Type: int
Default: 16384
Valid Values: [0,...]
Importance: medium
The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition. This helps performance on both the client and the server. This configuration controls the default batch size in bytes.

No attempt will be made to batch records larger than this size.

Requests sent to brokers will contain multiple batches, one for each partition with data available to be sent.

A small batch size will make batching less common and may reduce throughput (a batch size of zero will disable batching entirely). A very large batch size may use memory a bit more wastefully as we will always allocate a buffer of the specified batch size in anticipation of additional records.

Note: This setting gives the upper bound of the batch size to be sent. If we have fewer than this many bytes accumulated for this partition, we will 'linger' for the **linger.ms** time waiting for more records to show up. This **linger.ms** setting defaults to 0, which means we'll immediately send out a record even the accumulated batch size is under this **batch.size** setting.

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
Controls how the client uses DNS lookups. If set to **use_all_dns_ips**, connect to each returned IP address in sequence until a successful connection is established. After a disconnection, the next IP is used. Once all IPs have been used once, the client resolves the IP(s) from the hostname again (both

the JVM and the OS cache DNS name lookups, however). If set to **resolve_canonical_bootstrap_servers_only**, resolve each bootstrap address into a list of canonical names. After the bootstrap phase, this behaves the same as **use_all_dns_ips**.

client.id

Type: string
Default: ""
Importance: medium
An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
Close idle connections after the number of milliseconds specified by this config.

delivery.timeout.ms

Type: int
Default: 120000 (2 minutes)
Valid Values: [0,...]
Importance: medium
An upper bound on the time to report success or failure after a call to **send()** returns. This limits the total time that a record will be delayed prior to sending, the time to await acknowledgement from the broker (if expected), and the time allowed for retriable send failures. The producer may report failure to send a record earlier than this config if either an unrecoverable error is encountered, the retries have been exhausted, or the record is added to a batch which reached an earlier delivery expiration deadline. The value of this config should be greater than or equal to the sum of **request.timeout.ms** and **linger.ms**.

linger.ms

Type: long
Default: 0
Valid Values: [0,...]
Importance: medium
The producer groups together any records that arrive in between request transmissions into a single batched request. Normally this occurs only under load when records arrive faster than they can be sent out. However in some circumstances the client may want to reduce the number of requests even under moderate load. This setting accomplishes this by adding a small amount of artificial delay—that is, rather than immediately sending out a record, the producer will wait for up to the given delay to allow other records to be sent so that the sends can be batched together. This can be thought of as analogous to Nagle's algorithm in TCP. This setting gives the upper bound on the delay for batching: once we get **batch.size** worth of records for a partition it will be sent immediately regardless of this setting, however if we have fewer than this many bytes accumulated for this partition we will 'linger' for the specified time waiting for more records to show up. This setting defaults to 0 (i.e. no delay). Setting **linger.ms=5**, for example, would have the effect of reducing the number of requests sent but would add up to 5ms of latency to records sent in the absence of load.

max.block.ms

Type: long
Default: 60000 (1 minute)
Valid Values: [0,...]

Importance: medium
The configuration controls how long the **KafkaProducer's `send()**, **partitionsFor()**, **initTransactions()**, **sendOffsetsToTransaction()**, **commitTransaction()** and **abortTransaction()** methods will block. For **send()** this timeout bounds the total time waiting for both metadata fetch and buffer allocation (blocking in the user-supplied serializers or partitioner is not counted against this timeout). For **partitionsFor()** this timeout bounds the time spent waiting for metadata if it is unavailable. The transaction-related methods always block, but may timeout if the transaction coordinator could not be discovered or did not respond within the timeout.

max.request.size

Type: int
Default: 1048576
Valid Values: [0,...]
Importance: medium
The maximum size of a request in bytes. This setting will limit the number of record batches the producer will send in a single request to avoid sending huge requests. This is also effectively a cap on the maximum uncompressed record batch size. Note that the server has its own cap on the record batch size (after compression if compression is enabled) which may be different from this.

partitioner.class

Type: class
Default: null
Importance: medium
A class to use to determine which partition to be send to when produce the records. Available options are:

- If not set, the default partitioning logic is used. This strategy will try sticking to a partition until batch.size bytes is produced to the partition. It works with the strategy:
- If no partition is specified but a key is present, choose a partition based on a hash of the key
- If no partition or key is present, choose the sticky partition that changes when batch.size bytes are produced to the partition.
- **org.apache.kafka.clients.producer.RoundRobinPartitioner**: This partitioning strategy is that each record in a series of consecutive records will be sent to a different partition(no matter if the 'key' is provided or not), until we run out of partitions and start over again. Note: There's a known issue that will cause uneven distribution when new batch is created. Please check KAFKA-9965 for more detail.
  Implementing the **org.apache.kafka.clients.producer.Partitioner** interface allows you to plug in a custom partitioner.

partitioner.ignore.keys

Type: boolean
Default: false
Importance: medium
When set to 'true' the producer won't use record keys to choose a partition. If 'false', producer would choose a partition based on a hash of the key when a key is present. Note: this setting has no effect if a custom partitioner is used.

receive.buffer.bytes

Type: int
Default: 32768 (32 kibibytes)

Valid Values: [-1,...]
Importance: medium
The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.

request.timeout.ms

Type: int
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: medium
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted. This should be larger than **replica.lag.time.max.ms** (a broker configuration) to reduce the possibility of message duplication due to unnecessary producer retries.

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.

sasl.jaas.config

Type: password
Default: null
Importance: medium
JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here. The format for the value is: **loginModuleClass controlFlag (optionName=optionValue)*;**. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;.

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler.

sasl.login.class

Type: class
Default: null
Importance: medium
The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
The OAuth/OIDC provider URL from which the provider's JWKS (JSON Web Key Set) can be retrieved. The URL can be HTTP(S)-based or file-based. If the URL is HTTP(S)-based, the JWKS data will be retrieved from the OAuth/OIDC provider via the configured URL on broker startup. All then-current keys will be cached on the broker for incoming requests. If an authentication request is received for a JWT that includes a "kid" header claim value that isn't yet in the cache, the JWKS endpoint will be queried again on demand. However, the broker polls the URL every sasl.oauthbearer.jwks.endpoint.refresh.ms milliseconds to refresh the cache with any forthcoming keys before any JWT requests that include them are received. If the URL is file-based, the broker will load the JWKS file from a configured location on startup. In the event that the JWT includes a "kid" header value that isn't in the JWKS file, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
The URL for the OAuth/OIDC identity provider. If the URL is HTTP(S)-based, it is the issuer's token endpoint URL to which requests will be made to login based on the configuration in sasl.jaas.config. If the URL is file-based, it specifies a file containing an access token (in JWT serialized form) issued by the OAuth/OIDC identity provider to use for authorization.

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

send.buffer.bytes

Type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.

socket.connection.setup.timeout.max.ms

Type: long
Default: 30000 (30 seconds)
Importance: medium
The maximum amount of time the client will wait for the socket connection to be established. The connection setup timeout will increase exponentially for each consecutive connection failure up to this maximum. To avoid connection storms, a randomization factor of 0.2 will be applied to the timeout resulting in a random range between 20% below and 20% above the computed value.

socket.connection.setup.timeout.ms

Type: long
Default: 10000 (10 seconds)
Importance: medium
The amount of time the client will wait for the socket connection to be established. If the connection is not built before the timeout elapses, clients will close the socket channel.

ssl.enabled.protocols

Type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
The list of protocols enabled for SSL connections. The default is 'TLSv1.2,TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. With the default value for Java 11, clients and servers will prefer TLSv1.3 if both support it and fallback to TLSv1.2 otherwise (assuming both support at least TLSv1.2). This default should be fine for most cases. Also see the config documentation for **ssl.protocol**.

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
The file format of the key store file. This is optional for client. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

ssl.protocol

Type: string
Default: TLSv1.3
Importance: medium
The SSL protocol used to generate the SSLContext. The default is 'TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. This value should be fine for most use cases. Allowed values in recent JVMs are 'TLSv1.2' and 'TLSv1.3'. 'TLS', 'TLSv1.1', 'SSL', 'SSLv2' and 'SSLv3' may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities. With the default value for this config and 'ssl.enabled.protocols', clients will downgrade to 'TLSv1.2' if the server does not support 'TLSv1.3'. If this config is set to 'TLSv1.2', clients will not use 'TLSv1.3' even if it is one of the values in ssl.enabled.protocols and the server only supports 'TLSv1.3'.

ssl.provider

Type: string
Default: null
Importance: medium
The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
The file format of the trust store file. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

acks

Type: string
Default: all
Valid Values: [all, -1, 0, 1]
Importance: low
The number of acknowledgments the producer requires the leader to have received before considering a request complete. This controls the durability of records that are sent. The following settings are allowed:

- **acks=0** If set to zero then the producer will not wait for any acknowledgment from the server at all. The record will be immediately added to the socket buffer and considered sent. No guarantee can be made that the server has received the record in this case, and the **retries** configuration will not take effect (as the client won't generally know of any failures). The offset given back for each record will always be set to **-1**.

- **acks=1** This will mean the leader will write the record to its local log but will respond without awaiting full acknowledgement from all followers. In this case should the leader fail immediately after acknowledging the record but before the followers have replicated it then the record will be lost.

- **acks=all** This means the leader will wait for the full set of in-sync replicas to acknowledge the record. This guarantees that the record will not be lost as long as at least one in-sync replica remains alive. This is the strongest available guarantee. This is equivalent to the acks=-1 setting.
  Note that enabling idempotence requires this config value to be 'all'. If conflicting configurations are set and idempotence is not explicitly enabled, idempotence is disabled.

enable.idempotence

Type: boolean
Default: true
Importance: low
When set to 'true', the producer will ensure that exactly one copy of each message is written in the stream. If 'false', producer retries due to broker failures, etc., may write duplicates of the retried message in the stream. Note that enabling idempotence requires **max.in.flight.requests.per.connection** to be less than or equal to 5 (with message ordering preserved for any allowable value), **retries** to be greater than 0, and **acks** must be 'all'.

Idempotence is enabled by default if no conflicting configurations are set. If conflicting configurations are set and idempotence is not explicitly enabled, idempotence is disabled. If idempotence is explicitly enabled and conflicting configurations are set, a **ConfigException** is thrown.

interceptor.classes

Type: list
Default: ""
Valid Values: non-null string
Importance: low
A list of classes to use as interceptors. Implementing the

org.apache.kafka.clients.producer.ProducerInterceptor interface allows you to intercept (and possibly mutate) the records received by the producer before they are published to the Kafka cluster. By default, there are no interceptors.

max.in.flight.requests.per.connection

Type: int
Default: 5
Valid Values: [1,...]
Importance: low
The maximum number of unacknowledged requests the client will send on a single connection before blocking. Note that if this configuration is set to be greater than 1 and **enable.idempotence** is set to false, there is a risk of message reordering after a failed send due to retries (i.e., if retries are enabled); if retries are disabled or if **enable.idempotence** is set to true, ordering will be preserved. Additionally, enabling idempotence requires the value of this configuration to be less than or equal to 5. If conflicting configurations are set and idempotence is not explicitly enabled, idempotence is disabled.

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.

metadata.max.idle.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [5000,...]
Importance: low
Controls how long the producer will cache metadata for a topic that's idle. If the elapsed time since a topic was last produced to exceeds the metadata idle duration, then the topic's metadata is forgotten and the next access to it will force a metadata fetch request.

metric.reporters

Type: list
Default: ""
Valid Values: non-null string
Importance: low
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]
Importance: low
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string

Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The window of time a metrics sample is computed over.

partitioner.adaptive.partitioning.enable

Type: boolean
Default: true
Importance: low
When set to 'true', the producer will try to adapt to broker performance and produce more messages to partitions hosted on faster brokers. If 'false', producer will try to distribute messages uniformly. Note: this setting has no effect if a custom partitioner is used.

partitioner.availability.timeout.ms

Type: long
Default: 0
Valid Values: [0,...]
Importance: low
If a broker cannot process produce requests from a partition for **partitioner.availability.timeout.ms** time, the partitioner treats that partition as not available. If the value is 0, this logic is disabled. Note: this setting has no effect if a custom partitioner is used or
`partitioner.adaptive.partitioning.enable<code/> is set to 'false'.

reconnect.backoff.max.ms

Type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.

reconnect.backoff.ms

Type: long
Default: 50
Valid Values: [0,...]
Importance: low
The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.

retry.backoff.ms

Type: long
Default: 100

Valid Values: [0,...]
Importance: low
The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

sasl.kerberos.kinit.cmd

Type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit command path.

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
Login thread sleep time between refresh attempts.

sasl.kerberos.ticket.renew.jitter

Type: double
Default: 0.05
Importance: low
Percentage of random jitter added to the renewal time.

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider connection timeout. Currently applies only to OAUTHBEARER.

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider read timeout. Currently applies only to OAUTHBEARER.

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds

then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.factor

Type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.jitter

Type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum

wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
The (optional) value in seconds to allow for differences between the time of the OAuth/OIDC identity provider and the broker.

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
The (optional) comma-delimited setting for the broker to use to verify that the JWT was issued for one of the expected audiences. The JWT will be inspected for the standard OAuth "aud" claim and if this value is set, the broker will match the value from JWT's "aud" claim to see if there is an exact match. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
The (optional) setting for the broker to use to verify that the JWT was created by the expected issuer. The JWT will be inspected for the standard OAuth "iss" claim and if this value is set, the broker will match it exactly against what is in the JWT's "iss" claim. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
The (optional) value in milliseconds for the broker to wait between refreshing its JWKS (JSON Web Key Set) cache that contains the keys to verify the signature of the JWT.

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between attempts to retrieve the JWKS (JSON Web Key Set) from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between JWKS (JSON Web Key Set) retrieval attempts from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting

and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
The OAuth claim for the scope is often named "scope", but this (optional) setting can provide a different name to use for the scope included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
The OAuth claim for the subject is often named "sub", but this (optional) setting can provide a different name to use for the subject included in the JWT payload's claims if the OAuth/OIDC provider uses a different name for that claim.

security.providers

Type: string
Default: null
Importance: low
A list of configurable creator classes each returning a provider implementing security algorithms. These classes should implement the **org.apache.kafka.common.security.auth.SecurityProviderCreator** interface.

ssl.cipher.suites

Type: list
Default: null
Importance: low
A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
The endpoint identification algorithm to validate server hostname using server certificate.

ssl.engine.factory.class

Type: class
Default: null
Importance: low
The class of type org.apache.kafka.common.security.auth.SslEngineFactory to provide SSLEngine objects. Default value is org.apache.kafka.common.security.ssl.DefaultSslEngineFactory.

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
The SecureRandom PRNG implementation to use for SSL cryptography operations.

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.

transaction.timeout.ms

Type: int
Default: 60000 (1 minute)
Importance: low
The maximum amount of time in ms that the transaction coordinator will wait for a transaction status update from the producer before proactively aborting the ongoing transaction.If this value is larger than the transaction.max.timeout.ms setting in the broker, the request will fail with a **InvalidTxnTimeoutException** error.

transactional.id

Type: string
Default: null
Valid Values: non-empty string
Importance: low
The TransactionalId to use for transactional delivery. This enables reliability semantics which span multiple producer sessions since it allows the client to guarantee that transactions using the same TransactionalId have been completed prior to starting any new transactions. If no TransactionalId is provided, then the producer is limited to idempotent delivery. If a TransactionalId is configured, **enable.idempotence** is implied. By default the TransactionId is not configured, which means transactions cannot be used. Note that, by default, transactions require a cluster of at least three brokers which is the recommended setting for production; for development you can change this, by adjusting broker setting **transaction.state.log.replication.factor**.

CHAPTER 5. ADMIN CLIENT CONFIGURATION PROPERTIES

bootstrap.servers

Type: list
Importance: high
A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form **host1:port1,host2:port2,...**. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

ssl.key.password

Type: password
Default: null
Importance: high
The password of the private key in the key store file or the PEM key specified in `ssl.keystore.key'.

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
Certificate chain in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with a list of X.509 certificates.

ssl.keystore.key

Type: password
Default: null
Importance: high
Private key in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with PKCS#8 keys. If the key is encrypted, key password must be specified using 'ssl.key.password'.

ssl.keystore.location

Type: string
Default: null
Importance: high
The location of the key store file. This is optional for client and can be used for two-way authentication for client.

ssl.keystore.password

Type: password
Default: null
Importance: high
The store password for the key store file. This is optional for client and only needed if 'ssl.keystore.location' is configured. Key store password is not supported for PEM format.

ssl.truststore.certificates

Type: password
Default: null
Importance: high

Trusted certificates in the format specified by 'ssl.truststore.type'. Default SSL engine factory supports only PEM format with X.509 certificates.

ssl.truststore.location

Type: string
Default: null
Importance: high
The location of the trust store file.

ssl.truststore.password

Type: password
Default: null
Importance: high
The password for the trust store file. If a password is not set, trust store file configured will still be used, but integrity checking is disabled. Trust store password is not supported for PEM format.

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
Controls how the client uses DNS lookups. If set to **use_all_dns_ips**, connect to each returned IP address in sequence until a successful connection is established. After a disconnection, the next IP is used. Once all IPs have been used once, the client resolves the IP(s) from the hostname again (both the JVM and the OS cache DNS name lookups, however). If set to **resolve_canonical_bootstrap_servers_only**, resolve each bootstrap address into a list of canonical names. After the bootstrap phase, this behaves the same as **use_all_dns_ips**.

client.id

Type: string
Default: ""
Importance: medium
An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.

connections.max.idle.ms

Type: long
Default: 300000 (5 minutes)
Importance: medium
Close idle connections after the number of milliseconds specified by this config.

default.api.timeout.ms

Type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: medium
Specifies the timeout (in milliseconds) for client APIs. This configuration is used as the default timeout for all client operations that do not specify a **timeout** parameter.

receive.buffer.bytes

Type: int
Default: 65536 (64 kibibytes)
Valid Values: [-1,...]
Importance: medium
The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.

request.timeout.ms

Type: int
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: medium
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.

sasl.jaas.config

Type: password
Default: null
Importance: medium
JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here. The format for the value is: **loginModuleClass controlFlag (optionName=optionValue)*;**. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;.

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler.

sasl.login.class

Type: class
Default: null
Importance: medium
The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
The OAuth/OIDC provider URL from which the provider's JWKS (JSON Web Key Set) can be retrieved. The URL can be HTTP(S)-based or file-based. If the URL is HTTP(S)-based, the JWKS data will be retrieved from the OAuth/OIDC provider via the configured URL on broker startup. All then-current keys will be cached on the broker for incoming requests. If an authentication request is received for a JWT that includes a "kid" header claim value that isn't yet in the cache, the JWKS endpoint will be queried again on demand. However, the broker polls the URL every sasl.oauthbearer.jwks.endpoint.refresh.ms milliseconds to refresh the cache with any forthcoming keys before any JWT requests that include them are received. If the URL is file-based, the broker will load the JWKS file from a configured location on startup. In the event that the JWT includes a "kid" header value that isn't in the JWKS file, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
The URL for the OAuth/OIDC identity provider. If the URL is HTTP(S)-based, it is the issuer's token endpoint URL to which requests will be made to login based on the configuration in sasl.jaas.config. If the URL is file-based, it specifies a file containing an access token (in JWT serialized form) issued by the OAuth/OIDC identity provider to use for authorization.

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

send.buffer.bytes

Type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.

socket.connection.setup.timeout.max.ms

Type: long
Default: 30000 (30 seconds)
Importance: medium
The maximum amount of time the client will wait for the socket connection to be established. The connection setup timeout will increase exponentially for each consecutive connection failure up to this maximum. To avoid connection storms, a randomization factor of 0.2 will be applied to the timeout resulting in a random range between 20% below and 20% above the computed value.

socket.connection.setup.timeout.ms

Type: long
Default: 10000 (10 seconds)
Importance: medium
The amount of time the client will wait for the socket connection to be established. If the connection is not built before the timeout elapses, clients will close the socket channel.

ssl.enabled.protocols

Type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
The list of protocols enabled for SSL connections. The default is 'TLSv1.2,TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. With the default value for Java 11, clients and servers will prefer TLSv1.3 if both support it and fallback to TLSv1.2 otherwise (assuming both support at least TLSv1.2). This default should be fine for most cases. Also see the config documentation for **ssl.protocol**.

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
The file format of the key store file. This is optional for client. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

ssl.protocol

Type: string
Default: TLSv1.3
Importance: medium
The SSL protocol used to generate the SSLContext. The default is 'TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. This value should be fine for most use cases. Allowed values in recent JVMs are 'TLSv1.2' and 'TLSv1.3'. 'TLS', 'TLSv1.1', 'SSL', 'SSLv2' and 'SSLv3' may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities. With the default value for this config and 'ssl.enabled.protocols', clients will downgrade to 'TLSv1.2' if the server does not support 'TLSv1.3'. If this config is set to 'TLSv1.2', clients will not use 'TLSv1.3' even if it is one of the values in ssl.enabled.protocols and the server only supports 'TLSv1.3'.

ssl.provider

Type: string
Default: null
Importance: medium
The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
The file format of the trust store file. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.

metric.reporters

Type: list
Default: ""
Importance: low
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]
Importance: low
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The window of time a metrics sample is computed over.

reconnect.backoff.max.ms

Type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.

reconnect.backoff.ms

Type: long
Default: 50
Valid Values: [0,...]
Importance: low
The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.

retries

Type: int
Default: 2147483647
Valid Values: [0,...,2147483647]
Importance: low
Setting a value greater than zero will cause the client to resend any request that fails with a potentially transient error. It is recommended to set the value to either zero or **MAX_VALUE** and use corresponding timeout parameters to control how long a client should retry a request.

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
The amount of time to wait before attempting to retry a failed request. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

sasl.kerberos.kinit.cmd

Type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit command path.

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
Login thread sleep time between refresh attempts.

sasl.kerberos.ticket.renew.jitter

Type: double
Default: 0.05
Importance: low
Percentage of random jitter added to the renewal time.

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider connection timeout. Currently applies only to OAUTHBEARER.

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider read timeout. Currently applies only to OAUTHBEARER.

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.factor

Type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.jitter

Type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low

The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
The (optional) value in seconds to allow for differences between the time of the OAuth/OIDC identity provider and the broker.

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
The (optional) comma-delimited setting for the broker to use to verify that the JWT was issued for one of the expected audiences. The JWT will be inspected for the standard OAuth "aud" claim and if this value is set, the broker will match the value from JWT's "aud" claim to see if there is an exact match. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
The (optional) setting for the broker to use to verify that the JWT was created by the expected issuer. The JWT will be inspected for the standard OAuth "iss" claim and if this value is set, the broker will match it exactly against what is in the JWT's "iss" claim. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
The (optional) value in milliseconds for the broker to wait between refreshing its JWKS (JSON Web Key Set) cache that contains the keys to verify the signature of the JWT.

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between attempts to retrieve the JWKS (JSON Web Key Set) from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between JWKS (JSON Web Key Set) retrieval attempts from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
The OAuth claim for the scope is often named "scope", but this (optional) setting can provide a different name to use for the scope included in the JWT payload’s claims if the OAuth/OIDC provider uses a different name for that claim.

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
The OAuth claim for the subject is often named "sub", but this (optional) setting can provide a different name to use for the subject included in the JWT payload’s claims if the OAuth/OIDC provider uses a different name for that claim.

security.providers

Type: string
Default: null
Importance: low
A list of configurable creator classes each returning a provider implementing security algorithms. These classes should implement the **org.apache.kafka.common.security.auth.SecurityProviderCreator** interface.

ssl.cipher.suites

Type: list
Default: null
Importance: low
A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
The endpoint identification algorithm to validate server hostname using server certificate.

ssl.engine.factory.class

Type: class
Default: null
Importance: low
The class of type org.apache.kafka.common.security.auth.SslEngineFactory to provide SSLEngine objects. Default value is org.apache.kafka.common.security.ssl.DefaultSslEngineFactory.

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
The SecureRandom PRNG implementation to use for SSL cryptography operations.

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.

CHAPTER 6. KAFKA CONNECT CONFIGURATION PROPERTIES

config.storage.topic

Type: string
Importance: high
The name of the Kafka topic where connector configurations are stored.

group.id

Type: string
Importance: high
A unique string that identifies the Connect cluster group this worker belongs to.

key.converter

Type: class
Importance: high
Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the keys in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro.

offset.storage.topic

Type: string
Importance: high
The name of the Kafka topic where connector offsets are stored.

status.storage.topic

Type: string
Importance: high
The name of the Kafka topic where connector and task status are stored.

value.converter

Type: class
Importance: high
Converter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the values in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro.

bootstrap.servers

Type: list
Default: localhost:9092
Importance: high
A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form **host1:port1,host2:port2,...**. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

exactly.once.source.support

Type: string
Default: disabled
Valid Values: (case insensitive) [DISABLED, ENABLED, PREPARING]
Importance: high
Whether to enable exactly-once support for source connectors in the cluster by using transactions to write source records and their source offsets, and by proactively fencing out old task generations before bringing up new ones.

heartbeat.interval.ms

Type: int
Default: 3000 (3 seconds)
Importance: high
The expected time between heartbeats to the group coordinator when using Kafka's group management facilities. Heartbeats are used to ensure that the worker's session stays active and to facilitate rebalancing when new members join or leave the group. The value must be set lower than **session.timeout.ms**, but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.

rebalance.timeout.ms

Type: int
Default: 60000 (1 minute)
Importance: high
The maximum allowed time for each worker to join the group once a rebalance has begun. This is basically a limit on the amount of time needed for all tasks to flush any pending data and commit offsets. If the timeout is exceeded, then the worker will be removed from the group, which will cause offset commit failures.

session.timeout.ms

Type: int
Default: 10000 (10 seconds)
Importance: high
The timeout used to detect worker failures. The worker sends periodic heartbeats to indicate its liveness to the broker. If no heartbeats are received by the broker before the expiration of this session timeout, then the broker will remove the worker from the group and initiate a rebalance. Note that the value must be in the allowable range as configured in the broker configuration by **group.min.session.timeout.ms** and **group.max.session.timeout.ms**.

ssl.key.password

Type: password
Default: null
Importance: high
The password of the private key in the key store file or the PEM key specified in `ssl.keystore.key'.

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
Certificate chain in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with a list of X.509 certificates.

ssl.keystore.key

Type: password
Default: null
Importance: high
Private key in the format specified by 'ssl.keystore.type'. Default SSL engine factory supports only PEM format with PKCS#8 keys. If the key is encrypted, key password must be specified using 'ssl.key.password'.

ssl.keystore.location

Type: string
Default: null
Importance: high
The location of the key store file. This is optional for client and can be used for two-way authentication for client.

ssl.keystore.password

Type: password
Default: null
Importance: high
The store password for the key store file. This is optional for client and only needed if 'ssl.keystore.location' is configured. Key store password is not supported for PEM format.

ssl.truststore.certificates

Type: password
Default: null
Importance: high
Trusted certificates in the format specified by 'ssl.truststore.type'. Default SSL engine factory supports only PEM format with X.509 certificates.

ssl.truststore.location

Type: string
Default: null
Importance: high
The location of the trust store file.

ssl.truststore.password

Type: password
Default: null
Importance: high
The password for the trust store file. If a password is not set, trust store file configured will still be used, but integrity checking is disabled. Trust store password is not supported for PEM format.

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
Controls how the client uses DNS lookups. If set to **use_all_dns_ips**, connect to each returned IP address in sequence until a successful connection is established. After a disconnection, the next IP is used. Once all IPs have been used once, the client resolves the IP(s) from the hostname again (both the JVM and the OS cache DNS name lookups, however). If set to **resolve_canonical_bootstrap_servers_only**, resolve each bootstrap address into a list of canonical names. After the bootstrap phase, this behaves the same as **use_all_dns_ips**.

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
Close idle connections after the number of milliseconds specified by this config.

connector.client.config.override.policy

Type: string
Default: All
Importance: medium
Class name or alias of implementation of **ConnectorClientConfigOverridePolicy**. Defines what client configurations can be overriden by the connector. The default implementation is **All**, meaning connector configurations can override all client properties. The other possible policies in the framework include **None** to disallow connectors from overriding client properties, and **Principal** to allow connectors to override only client principals.

receive.buffer.bytes

Type: int
Default: 32768 (32 kibibytes)
Valid Values: [0,...]
Importance: medium
The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.

request.timeout.ms

Type: int
Default: 40000 (40 seconds)
Valid Values: [0,...]
Importance: medium
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL client callback handler class that implements the AuthenticateCallbackHandler interface.

sasl.jaas.config

Type: password
Default: null
Importance: medium
JAAS login context parameters for SASL connections in the format used by JAAS configuration files. JAAS configuration file format is described here. The format for the value is: **loginModuleClass controlFlag (optionName=optionValue)*;**. For brokers, the config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;.

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
The Kerberos principal name that Kafka runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler.

sasl.login.class

Type: class
Default: null
Importance: medium
The fully qualified name of a class that implements the Login interface. For brokers, login config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
The OAuth/OIDC provider URL from which the provider's JWKS (JSON Web Key Set) can be retrieved. The URL can be HTTP(S)-based or file-based. If the URL is HTTP(S)-based, the JWKS data will be retrieved from the OAuth/OIDC provider via the configured URL on broker startup. All then-current keys will be cached on the broker for incoming requests. If an authentication request is received for a JWT that includes a "kid" header claim value that isn't yet in the cache, the JWKS endpoint will be queried again on demand. However, the broker polls the URL every sasl.oauthbearer.jwks.endpoint.refresh.ms milliseconds to refresh the cache with any forthcoming keys before any JWT requests that include them are received. If the URL is file-based, the broker will load the JWKS file from a configured location on startup. In the event that the JWT includes a "kid" header value that isn't in the JWKS file, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
The URL for the OAuth/OIDC identity provider. If the URL is HTTP(S)-based, it is the issuer's token endpoint URL to which requests will be made to login based on the configuration in sasl.jaas.config. If the URL is file-based, it specifies a file containing an access token (in JWT serialized form) issued by

the OAuth/OIDC identity provider to use for authorization.

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

send.buffer.bytes

Type: int
Default: 131072 (128 kibibytes)
Valid Values: [0,...]
Importance: medium
The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.

ssl.enabled.protocols

Type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
The list of protocols enabled for SSL connections. The default is 'TLSv1.2,TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. With the default value for Java 11, clients and servers will prefer TLSv1.3 if both support it and fallback to TLSv1.2 otherwise (assuming both support at least TLSv1.2). This default should be fine for most cases. Also see the config documentation for **ssl.protocol**.

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
The file format of the key store file. This is optional for client. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

ssl.protocol

Type: string
Default: TLSv1.3
Importance: medium
The SSL protocol used to generate the SSLContext. The default is 'TLSv1.3' when running with Java 11 or newer, 'TLSv1.2' otherwise. This value should be fine for most use cases. Allowed values in recent JVMs are 'TLSv1.2' and 'TLSv1.3'. 'TLS', 'TLSv1.1', 'SSL', 'SSLv2' and 'SSLv3' may be supported in older JVMs, but their usage is discouraged due to known security vulnerabilities. With the default value for this config and 'ssl.enabled.protocols', clients will downgrade to 'TLSv1.2' if the server does not support 'TLSv1.3'. If this config is set to 'TLSv1.2', clients will not use 'TLSv1.3' even if it is one of the values in ssl.enabled.protocols and the server only supports 'TLSv1.3'.

ssl.provider

Type: string
Default: null
Importance: medium
The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
The file format of the trust store file. The values currently supported by the default **ssl.engine.factory.class** are [JKS, PKCS12, PEM].

worker.sync.timeout.ms

Type: int
Default: 3000 (3 seconds)
Importance: medium
When the worker is out of sync with other workers and needs to resynchronize configurations, wait up to this amount of time before giving up, leaving the group, and waiting a backoff period before rejoining.

worker.unsync.backoff.ms

Type: int
Default: 300000 (5 minutes)
Importance: medium
When the worker is out of sync with other workers and fails to catch up within worker.sync.timeout.ms, leave the Connect cluster for this long before rejoining.

access.control.allow.methods

Type: string
Default: ""
Importance: low
Sets the methods supported for cross origin requests by setting the Access-Control-Allow-Methods header. The default value of the Access-Control-Allow-Methods header allows cross origin requests for GET, POST and HEAD.

access.control.allow.origin

Type: string
Default: ""
Importance: low
Value to set the Access-Control-Allow-Origin header to for REST API requests.To enable cross origin access, set this to the domain of the application that should be permitted to access the API, or '*' to allow access from any domain. The default value only allows access from the domain of the REST API.

admin.listeners

Type: list
Default: null
Valid Values: List of comma-separated URLs, ex: http://localhost:8080,https://localhost:8443.
Importance: low
List of comma-separated URIs the Admin REST API will listen on. The supported protocols are HTTP and HTTPS. An empty or blank string will disable this feature. The default behavior is to use the regular listener (specified by the 'listeners' property).

client.id

Type: string
Default: ""
Importance: low
An id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included in server-side request logging.

config.providers

Type: list
Default: ""
Importance: low
Comma-separated names of **ConfigProvider** classes, loaded and used in the order specified. Implementing the interface **ConfigProvider** allows you to replace variable references in connector configurations, such as for externalized secrets.

config.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the number of brokers in the Kafka cluster, or -1 to use the broker's default
Importance: low
Replication factor used when creating the configuration storage topic.

connect.protocol

Type: string
Default: sessioned
Valid Values: [eager, compatible, sessioned]
Importance: low
Compatibility mode for Kafka Connect Protocol.

header.converter

Type: class
Default: org.apache.kafka.connect.storage.SimpleHeaderConverter
Importance: low
HeaderConverter class used to convert between Kafka Connect format and the serialized form that is written to Kafka. This controls the format of the header values in messages written to or read from Kafka, and since this is independent of connectors it allows any connector to work with any serialization format. Examples of common formats include JSON and Avro. By default, the SimpleHeaderConverter is used to serialize header values to strings and deserialize them by inferring the schemas.

inter.worker.key.generation.algorithm

Type: string
Default: HmacSHA256
Valid Values: Any KeyGenerator algorithm supported by the worker JVM
Importance: low
The algorithm to use for generating internal request keys. The algorithm 'HmacSHA256' will be used as a default on JVMs that support it; on other JVMs, no default is used and a value for this property must be manually specified in the worker config.

inter.worker.key.size

Type: int
Default: null
Importance: low
The size of the key to use for signing internal requests, in bits. If null, the default key size for the key generation algorithm will be used.

inter.worker.key.ttl.ms

Type: int
Default: 3600000 (1 hour)
Valid Values: [0,...,2147483647]
Importance: low
The TTL of generated session keys used for internal request validation (in milliseconds).

inter.worker.signature.algorithm

Type: string
Default: HmacSHA256
Valid Values: Any MAC algorithm supported by the worker JVM
Importance: low
The algorithm used to sign internal requestsThe algorithm 'inter.worker.signature.algorithm' will be used as a default on JVMs that support it; on other JVMs, no default is used and a value for this property must be manually specified in the worker config.

inter.worker.verification.algorithms

Type: list
Default: HmacSHA256
Valid Values: A list of one or more MAC algorithms, each supported by the worker JVM
Importance: low
A list of permitted algorithms for verifying internal requests, which must include the algorithm used for the inter.worker.signature.algorithm property. The algorithm(s) '[HmacSHA256]' will be used as a default on JVMs that provide them; on other JVMs, no default is used and a value for this property must be manually specified in the worker config.

listeners

Type: list
Default: http://:8083
Valid Values: List of comma-separated URLs, ex: http://localhost:8080,https://localhost:8443.
Importance: low
List of comma-separated URIs the REST API will listen on. The supported protocols are HTTP and HTTPS. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: HTTP://myhost:8083,HTTPS://myhost:8084.

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.

metric.reporters

Type: list
Default: ""
Importance: low
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]
Importance: low
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG]
Importance: low
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The window of time a metrics sample is computed over.

offset.flush.interval.ms

Type: long
Default: 60000 (1 minute)
Importance: low
Interval at which to try committing offsets for tasks.

offset.flush.timeout.ms

Type: long
Default: 5000 (5 seconds)
Importance: low
Maximum number of milliseconds to wait for records to flush and partition offset data to be committed to offset storage before cancelling the process and restoring the offset data to be committed in a future attempt. This property has no effect for source connectors running with exactly-once support.

offset.storage.partitions

Type: int
Default: 25
Valid Values: Positive number, or -1 to use the broker's default
Importance: low
The number of partitions used when creating the offset storage topic.

offset.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the number of brokers in the Kafka cluster, or -1 to use the broker's default
Importance: low
Replication factor used when creating the offset storage topic.

plugin.path

Type: list
Default: null
Importance: low
List of paths separated by commas (,) that contain plugins (connectors, converters, transformations). The list should consist of top level directories that include any combination of: a) directories immediately containing jars with plugins and their dependencies b) uber-jars with plugins and their dependencies c) directories immediately containing the package directory structure of classes of plugins and their dependencies Note: symlinks will be followed to discover dependencies or plugins. Examples:
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors Do not use config provider variables in this property, since the raw path is used by the worker's scanner before config providers are initialized and used to replace variables.

reconnect.backoff.max.ms

Type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.

reconnect.backoff.ms

Type: long
Default: 50
Valid Values: [0,...]
Importance: low
The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.

response.http.headers.config

Type: string
Default: ""
Valid Values: Comma-separated header rules, where each header rule is of the form '[action] [header name]:[header value]' and optionally surrounded by double quotes if any part of a header rule contains a comma
Importance: low
Rules for REST API HTTP response headers.

rest.advertised.host.name

Type: string
Default: null
Importance: low

If this is set, this is the hostname that will be given out to other workers to connect to.

rest.advertised.listener

Type: string
Default: null
Importance: low
Sets the advertised listener (HTTP or HTTPS) which will be given to other workers to use.

rest.advertised.port

Type: int
Default: null
Importance: low
If this is set, this is the port that will be given out to other workers to connect to.

rest.extension.classes

Type: list
Default: ""
Importance: low
Comma-separated names of **ConnectRestExtension** classes, loaded and called in the order specified. Implementing the interface **ConnectRestExtension** allows you to inject into Connect's REST API user defined resources like filters. Typically used to add custom capability like logging, security, etc.

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

sasl.kerberos.kinit.cmd

Type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit command path.

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
Login thread sleep time between refresh attempts.

sasl.kerberos.ticket.renew.jitter

Type: double
Default: 0.05
Importance: low
Percentage of random jitter added to the renewal time.

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
Login thread will sleep until the specified window factor of time from last refresh to ticket's expiry has been reached, at which time it will try to renew the ticket.

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider connection timeout. Currently applies only to OAUTHBEARER.

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: low
The (optional) value in milliseconds for the external authentication provider read timeout. Currently applies only to OAUTHBEARER.

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
The amount of buffer time before credential expiration to maintain when refreshing a credential, in seconds. If a refresh would otherwise occur closer to expiration than the number of buffer seconds then the refresh will be moved up to maintain as much of the buffer time as possible. Legal values are between 0 and 3600 (1 hour); a default value of 300 (5 minutes) is used if no value is specified. This value and sasl.login.refresh.min.period.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
The desired minimum time for the login refresh thread to wait before refreshing a credential, in seconds. Legal values are between 0 and 900 (15 minutes); a default value of 60 (1 minute) is used if no value is specified. This value and sasl.login.refresh.buffer.seconds are both ignored if their sum exceeds the remaining lifetime of a credential. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.factor

Type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
Login refresh thread will sleep until the specified window factor relative to the credential's lifetime has been reached, at which time it will try to refresh the credential. Legal values are between 0.5 (50%) and 1.0 (100%) inclusive; a default value of 0.8 (80%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.refresh.window.jitter

Type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
The maximum amount of random jitter relative to the credential's lifetime that is added to the login refresh thread's sleep time. Legal values are between 0 and 0.25 (25%) inclusive; a default value of 0.05 (5%) is used if no value is specified. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between login attempts to the external authentication provider. Login uses an exponential backoff algorithm with an initial wait based on the sasl.login.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.login.retry.backoff.max.ms setting. Currently applies only to OAUTHBEARER.

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
The (optional) value in seconds to allow for differences between the time of the OAuth/OIDC identity provider and the broker.

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
The (optional) comma-delimited setting for the broker to use to verify that the JWT was issued for one of the expected audiences. The JWT will be inspected for the standard OAuth "aud" claim and if this value is set, the broker will match the value from JWT's "aud" claim to see if there is an exact match. If there is no match, the broker will reject the JWT and authentication will fail.

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
The (optional) setting for the broker to use to verify that the JWT was created by the expected issuer. The JWT will be inspected for the standard OAuth "iss" claim and if this value is set, the broker will match it exactly against what is in the JWT's "iss" claim. If there is no match, the broker will reject

the JWT and authentication will fail.

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
The (optional) value in milliseconds for the broker to wait between refreshing its JWKS (JSON Web Key Set) cache that contains the keys to verify the signature of the JWT.

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms

Type: long
Default: 10000 (10 seconds)
Importance: low
The (optional) value in milliseconds for the maximum wait between attempts to retrieve the JWKS (JSON Web Key Set) from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms

Type: long
Default: 100
Importance: low
The (optional) value in milliseconds for the initial wait between JWKS (JSON Web Key Set) retrieval attempts from the external authentication provider. JWKS retrieval uses an exponential backoff algorithm with an initial wait based on the sasl.oauthbearer.jwks.endpoint.retry.backoff.ms setting and will double in wait length between attempts up to a maximum wait length specified by the sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting.

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
The OAuth claim for the scope is often named "scope", but this (optional) setting can provide a different name to use for the scope included in the JWT payload’s claims if the OAuth/OIDC provider uses a different name for that claim.

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
The OAuth claim for the subject is often named "sub", but this (optional) setting can provide a different name to use for the subject included in the JWT payload’s claims if the OAuth/OIDC provider uses a different name for that claim.

scheduled.rebalance.max.delay.ms

Type: int
Default: 300000 (5 minutes)
Valid Values: [0,...,2147483647]
Importance: low

The maximum delay that is scheduled in order to wait for the return of one or more departed workers before rebalancing and reassigning their connectors and tasks to the group. During this period the connectors and tasks of the departed workers remain unassigned.

socket.connection.setup.timeout.max.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The maximum amount of time the client will wait for the socket connection to be established. The connection setup timeout will increase exponentially for each consecutive connection failure up to this maximum. To avoid connection storms, a randomization factor of 0.2 will be applied to the timeout resulting in a random range between 20% below and 20% above the computed value.

socket.connection.setup.timeout.ms

Type: long
Default: 10000 (10 seconds)
Valid Values: [0,...]
Importance: low
The amount of time the client will wait for the socket connection to be established. If the connection is not built before the timeout elapses, clients will close the socket channel.

ssl.cipher.suites

Type: list
Default: null
Importance: low
A list of cipher suites. This is a named combination of authentication, encryption, MAC and key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol. By default all the available cipher suites are supported.

ssl.client.auth

Type: string
Default: none
Valid Values: [required, requested, none]
Importance: low
Configures kafka broker to request client authentication. The following settings are common:

- **ssl.client.auth=required** If set to required client authentication is required.
- **ssl.client.auth=requested** This means client authentication is optional. unlike required, if this option is set client can choose not to provide authentication information about itself
- **ssl.client.auth=none** This means client authentication is not needed.

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
The endpoint identification algorithm to validate server hostname using server certificate.

ssl.engine.factory.class

Type: class
Default: null
Importance: low
The class of type org.apache.kafka.common.security.auth.SslEngineFactory to provide SSLEngine objects. Default value is org.apache.kafka.common.security.ssl.DefaultSslEngineFactory.

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
The algorithm used by key manager factory for SSL connections. Default value is the key manager factory algorithm configured for the Java Virtual Machine.

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
The SecureRandom PRNG implementation to use for SSL cryptography operations.

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
The algorithm used by trust manager factory for SSL connections. Default value is the trust manager factory algorithm configured for the Java Virtual Machine.

status.storage.partitions

Type: int
Default: 5
Valid Values: Positive number, or -1 to use the broker's default
Importance: low
The number of partitions used when creating the status storage topic.

status.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the number of brokers in the Kafka cluster, or -1 to use the broker's default
Importance: low
Replication factor used when creating the status storage topic.

task.shutdown.graceful.timeout.ms

Type: long
Default: 5000 (5 seconds)
Importance: low
Amount of time to wait for tasks to shutdown gracefully. This is the total amount of time, not per task. All task have shutdown triggered, then they are waited on sequentially.

topic.creation.enable

Type: boolean
Default: true
Importance: low
Whether to allow automatic creation of topics used by source connectors, when source connectors are configured with **topic.creation.** properties. Each task will use an admin client to create its topics and will not depend on the Kafka brokers to create topics automatically.

topic.tracking.allow.reset

Type: boolean
Default: true
Importance: low
If set to true, it allows user requests to reset the set of active topics per connector.

topic.tracking.enable

Type: boolean
Default: true
Importance: low
Enable tracking the set of active topics per connector during runtime.

CHAPTER 7. KAFKA STREAMS CONFIGURATION PROPERTIES

application.id

Type: string
Importance: high
An identifier for the stream processing application. Must be unique within the Kafka cluster. It is used as 1) the default client-id prefix, 2) the group-id for membership management, 3) the changelog topic prefix.

bootstrap.servers

Type: list
Importance: high
A list of host/port pairs to use for establishing the initial connection to the Kafka cluster. The client will make use of all servers irrespective of which servers are specified here for bootstrapping—this list only impacts the initial hosts used to discover the full set of servers. This list should be in the form **host1:port1,host2:port2,...**. Since these servers are just used for the initial connection to discover the full cluster membership (which may change dynamically), this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

num.standby.replicas

Type: int
Default: 0
Importance: high
The number of standby replicas for each task.

state.dir

Type: string
Default: /tmp/kafka-streams
Importance: high
Directory location for state store. This path must be unique for each streams instance sharing the same underlying filesystem.

acceptable.recovery.lag

Type: long
Default: 10000
Valid Values: [0,...]
Importance: medium
The maximum acceptable lag (number of offsets to catch up) for a client to be considered caught-up enough to receive an active task assignment. Upon assignment, it will still restore the rest of the changelog before processing. To avoid a pause in processing during rebalances, this config should correspond to a recovery time of well under a minute for a given workload. Must be at least 0.

cache.max.bytes.buffering

Type: long
Default: 10485760
Valid Values: [0,...]
Importance: medium
Maximum number of memory bytes to be used for buffering across all threads.

client.id

Type: string
Default: ""
Importance: medium
An ID prefix string used for the client IDs of internal consumer, producer and restore-consumer, with pattern '<client.id>-StreamThread-<threadSequenceNumber>-<consumer|producer|restore-consumer>'.

default.deserialization.exception.handler

Type: class
Default: org.apache.kafka.streams.errors.LogAndFailExceptionHandler
Importance: medium
Exception handling class that implements the **org.apache.kafka.streams.errors.DeserializationExceptionHandler** interface.

default.key.serde

Type: class
Default: null
Importance: medium
Default serializer / deserializer class for key that implements the **org.apache.kafka.common.serialization.Serde** interface. Note when windowed serde class is used, one needs to set the inner serde class that implements the **org.apache.kafka.common.serialization.Serde** interface via 'default.windowed.key.serde.inner' or 'default.windowed.value.serde.inner' as well.

default.list.key.serde.inner

Type: class
Default: null
Importance: medium
Default inner class of list serde for key that implements the **org.apache.kafka.common.serialization.Serde** interface. This configuration will be read if and only if **default.key.serde** configuration is set to **org.apache.kafka.common.serialization.Serdes.ListSerde**.

default.list.key.serde.type

Type: class
Default: null
Importance: medium
Default class for key that implements the **java.util.List** interface. This configuration will be read if and only if **default.key.serde** configuration is set to **org.apache.kafka.common.serialization.Serdes.ListSerde** Note when list serde class is used, one needs to set the inner serde class that implements the **org.apache.kafka.common.serialization.Serde** interface via 'default.list.key.serde.inner'.

default.list.value.serde.inner

Type: class
Default: null
Importance: medium
Default inner class of list serde for value that implements the **org.apache.kafka.common.serialization.Serde** interface. This configuration will be read if and only if **default.value.serde** configuration is set to **org.apache.kafka.common.serialization.Serdes.ListSerde**.

default.list.value.serde.type

Type: class
Default: null
Importance: medium
Default class for value that implements the **java.util.List** interface. This configuration will be read if and only if **default.value.serde** configuration is set to **org.apache.kafka.common.serialization.Serdes.ListSerde** Note when list serde class is used, one needs to set the inner serde class that implements the **org.apache.kafka.common.serialization.Serde** interface via 'default.list.value.serde.inner'.

default.production.exception.handler

Type: class
Default: org.apache.kafka.streams.errors.DefaultProductionExceptionHandler
Importance: medium
Exception handling class that implements the **org.apache.kafka.streams.errors.ProductionExceptionHandler** interface.

default.timestamp.extractor

Type: class
Default: org.apache.kafka.streams.processor.FailOnInvalidTimestamp
Importance: medium
Default timestamp extractor class that implements the **org.apache.kafka.streams.processor.TimestampExtractor** interface.

default.value.serde

Type: class
Default: null
Importance: medium
Default serializer / deserializer class for value that implements the **org.apache.kafka.common.serialization.Serde** interface. Note when windowed serde class is used, one needs to set the inner serde class that implements the **org.apache.kafka.common.serialization.Serde** interface via 'default.windowed.key.serde.inner' or 'default.windowed.value.serde.inner' as well.

max.task.idle.ms

Type: long
Default: 0
Importance: medium
This config controls whether joins and merges may produce out-of-order results. The config value is the maximum amount of time in milliseconds a stream task will stay idle when it is fully caught up on some (but not all) input partitions to wait for producers to send additional records and avoid potential out-of-order record processing across multiple input streams. The default (zero) does not wait for producers to send more records, but it does wait to fetch data that is already present on the brokers. This default means that for records that are already present on the brokers, Streams will process them in timestamp order. Set to -1 to disable idling entirely and process any locally available data, even though doing so may produce out-of-order processing.

max.warmup.replicas

Type: int
Default: 2
Valid Values: [1,...]
Importance: medium

The maximum number of warmup replicas (extra standbys beyond the configured num.standbys) that can be assigned at once for the purpose of keeping the task available on one instance while it is warming up on another instance it has been reassigned to. Used to throttle how much extra broker traffic and cluster state can be used for high availability. Must be at least 1.

num.stream.threads

Type: int
Default: 1
Importance: medium
The number of threads to execute stream processing.

processing.guarantee

Type: string
Default: at_least_once
Valid Values: [at_least_once, exactly_once, exactly_once_beta, exactly_once_v2]
Importance: medium
The processing guarantee that should be used. Possible values are **at_least_once** (default) and **exactly_once_v2** (requires brokers version 2.5 or higher). Deprecated options are **exactly_once** (requires brokers version 0.11.0 or higher) and **exactly_once_beta** (requires brokers version 2.5 or higher). Note that exactly-once processing requires a cluster of at least three brokers by default what is the recommended setting for production; for development you can change this, by adjusting broker setting **transaction.state.log.replication.factor** and **transaction.state.log.min.isr**.

rack.aware.assignment.tags

Type: list
Default: ""
Valid Values: List containing maximum of 5 elements
Importance: medium
List of client tag keys used to distribute standby replicas across Kafka Streams instances. When configured, Kafka Streams will make a best-effort to distribute the standby tasks over each client tag dimension.

replication.factor

Type: int
Default: -1
Importance: medium
The replication factor for change log topics and repartition topics created by the stream processing application. The default of **-1** (meaning: use broker default replication factor) requires broker version 2.4 or newer.

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

task.timeout.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]

Importance: medium
The maximum amount of time in milliseconds a task might stall due to internal errors and retries until an error is raised. For a timeout of 0ms, a task would raise an error for the first internal error. For any timeout larger than 0ms, a task will retry at least once before an error is raised.

topology.optimization

Type: string
Default: none
Valid Values: [none, all]
Importance: medium
A configuration telling Kafka Streams if it should optimize the topology, disabled by default.

application.server

Type: string
Default: ""
Importance: low
A host:port pair pointing to a user-defined endpoint that can be used for state store discovery and interactive queries on this KafkaStreams instance.

buffered.records.per.partition

Type: int
Default: 1000
Importance: low
Maximum number of records to buffer per partition.

built.in.metrics.version

Type: string
Default: latest
Valid Values: [latest]
Importance: low
Version of the built-in metrics to use.

commit.interval.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The frequency in milliseconds with which to commit processing progress. For at-least-once processing, committing means to save the position (ie, offsets) of the processor. For exactly-once processing, it means to commit the transaction which includes to save the position and to make the committed data in the output topic visible to consumers with isolation level read_committed. (Note, if **processing.guarantee** is set to **exactly_once_v2**, **exactly_once**,the default value is **100**, otherwise the default value is **30000**.

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: low
Close idle connections after the number of milliseconds specified by this config.

default.dsl.store

Type: string
Default: rocksDB
Valid Values: [rocksDB, in_memory]
Importance: low
The default state store type used by DSL operators.

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions.

metric.reporters

Type: list
Default: ""
Importance: low
A list of classes to use as metrics reporters. Implementing the **org.apache.kafka.common.metrics.MetricsReporter** interface allows plugging in classes that will be notified of new metric creation. The JmxReporter is always included to register JMX statistics.

metrics.num.samples

Type: int
Default: 2
Valid Values: [1,...]
Importance: low
The number of samples maintained to compute metrics.

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
The highest recording level for metrics.

metrics.sample.window.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The window of time a metrics sample is computed over.

poll.ms

Type: long
Default: 100
Importance: low
The amount of time in milliseconds to block waiting for input.

probing.rebalance.interval.ms

Type: long

Default: 600000 (10 minutes)
Valid Values: [60000,...]
Importance: low
The maximum time in milliseconds to wait before triggering a rebalance to probe for warmup replicas that have finished warming up and are ready to become active. Probing rebalances will continue to be triggered until the assignment is balanced. Must be at least 1 minute.

receive.buffer.bytes

Type: int
Default: 32768 (32 kibibytes)
Valid Values: [-1,...]
Importance: low
The size of the TCP receive buffer (SO_RCVBUF) to use when reading data. If the value is -1, the OS default will be used.

reconnect.backoff.max.ms

Type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
The maximum amount of time in milliseconds to wait when reconnecting to a broker that has repeatedly failed to connect. If provided, the backoff per host will increase exponentially for each consecutive connection failure, up to this maximum. After calculating the backoff increase, 20% random jitter is added to avoid connection storms.

reconnect.backoff.ms

Type: long
Default: 50
Valid Values: [0,...]
Importance: low
The base amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all connection attempts by the client to a broker.

repartition.purge.interval.ms

Type: long
Default: 30000 (30 seconds)
Valid Values: [0,...]
Importance: low
The frequency in milliseconds with which to delete fully consumed records from repartition topics. Purging will occur after at least this value since the last purge, but may be delayed until later. (Note, unlike **commit.interval.ms**, the default for this value remains unchanged when **processing.guarantee** is set to **exactly_once_v2**).

request.timeout.ms

Type: int
Default: 40000 (40 seconds)
Valid Values: [0,...]
Importance: low
The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

retries

Type: int
Default: 0
Valid Values: [0,...,2147483647]
Importance: low
Setting a value greater than zero will cause the client to resend any request that fails with a potentially transient error. It is recommended to set the value to either zero or **MAX_VALUE** and use corresponding timeout parameters to control how long a client should retry a request.

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.

rocksdb.config.setter

Type: class
Default: null
Importance: low
A Rocks DB config setter class or class name that implements the **org.apache.kafka.streams.state.RocksDBConfigSetter** interface.

send.buffer.bytes

Type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: low
The size of the TCP send buffer (SO_SNDBUF) to use when sending data. If the value is -1, the OS default will be used.

state.cleanup.delay.ms

Type: long
Default: 600000 (10 minutes)
Importance: low
The amount of time in milliseconds to wait before deleting state when a partition has migrated. Only state directories that have not been modified for at least **state.cleanup.delay.ms** will be removed.

upgrade.from

Type: string
Default: null
Valid Values: [null, 0.10.0, 0.10.1, 0.10.2, 0.11.0, 1.0, 1.1, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 3.0, 3.1, 3.2]
Importance: low
Allows upgrading in a backward compatible way. This is needed when upgrading from [0.10.0, 1.1] to 2.0+, or when upgrading from [2.0, 2.3] to 2.4+. When upgrading from 3.3 to a newer version it is not required to specify this config. Default is **null**. Accepted values are "0.10.0", "0.10.1", "0.10.2", "0.11.0", "1.0", "1.1", "2.0", "2.1", "2.2", "2.3", "2.4", "2.5", "2.6", "2.7", "2.8", "3.0", "3.1", "3.2" (for upgrading from the corresponding old version).

window.size.ms

Type: long
Default: null
Importance: low
Sets window size for the deserializer in order to calculate window end times.

windowed.inner.class.serde

Type: string
Default: null
Importance: low
Default serializer / deserializer for the inner class of a windowed record. Must implement the **org.apache.kafka.common.serialization.Serde** interface. Note that setting this config in KafkaStreams application would result in an error as it is meant to be used only from Plain consumer client.

windowstore.changelog.additional.retention.ms

Type: long
Default: 86400000 (1 day)
Importance: low
Added to a windows maintainMs to ensure data is not deleted from the log prematurely. Allows for clock drift. Default is 1 day.

APPENDIX A. USING YOUR SUBSCRIPTION

AMQ Streams is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing Your Account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a Subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading Zip and Tar Files

To access zip or tar files, use the customer portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Locate the **AMQ Streams for Apache Kafka** entries in the **INTEGRATION AND AUTOMATION** category.
3. Select the desired AMQ Streams product. The **Software Downloads** page opens.
4. Click the **Download** link for your component.

Installing packages with DNF

To install a package and all the package dependencies, use:

```
dnf install <package_name>
```

To install a previously-downloaded package from a local directory, use:

```
dnf install <path_to_download_package>
```

Revised on 2023-01-16 17:54:26 UTC