Red Hat Streams for Apache Kafka 2.7

Kafka 配置属性

使用配置属性配置 Kafka 组件

Last Updated: 2024-05-29

Red Hat Streams for Apache Kafka 2.7 Kafka 配置属性

使用配置属性配置 Kafka 组件

法律通告

摘要

充分利用 Kafka 组件如何使用 Kafka 配置属性进行操作。

目录

前言

对红帽文档提供反馈

我们感谢您对我们文档的反馈。

要改进，创建一个 JIRA 问题并描述您推荐的更改。提供尽可能多的详细信息，以便我们快速解决您的请求。

前提条件

- 您有红帽客户门户网站帐户。此帐户可让您登录到 Red Hat Jira Software 实例。
  如果您没有帐户，系统会提示您创建一个帐户。

流程

1. 点以下内容： Create issue。
2. 在 **Summary** 文本框中输入问题的简短描述。
3. 在 **Description** 文本框中提供以下信息：
   - 找到此问题的页面的 URL。
   - 有关此问题的详细描述。
     您可以将信息保留在任何其他字段中的默认值。
4. 添加 reporter 名称。
5. 点 **Create** 将 JIRA 问题提交到文档团队。

感谢您花时间来提供反馈。

第 1 章 代理配置属性

advertised.listeners

Type: string
Default: null
Importance: high
Dynamic update: per-broker
与监听器配置属性不同，要发布到 ZooKeeper **的监听程序** 供客户端使用。在 IaaS 环境中，这可能需要与代理绑定到的接口不同。如果没有设置，则使用 **监听程序** 的值。与 **监听器** 不同，公告 0.0.0.0 meta-address 无效。与 **监听器** 不同，此属性中可能存在重复的端口，以便可以配置一个侦听器来公告另一个侦听器的地址。这在使用外部负载均衡器时很有用。

auto.create.topics.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
在服务器上启用自动创建主题。

auto.leader.rebalance.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
启用自动领导平衡。后台线程会定期检查分区领导分布，由 leader.imbalance.check.interval.seconds 进行配置。如果领导 imbalance 超过 leader.imbalance.per.broker.percentage，则触发领导到分区的首选领导。

background.threads

type: int
Default: 10
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
用于各种后台处理任务的线程数量。

broker.id

type: int
Default: -1
Importance: high
Dynamic update: read-only
此服务器的代理 ID。如果未设置，则会生成唯一的代理 id。要避免 ZooKeeper 生成的代理 ID 和用户配置的代理 ID 间的冲突，从 reserved.broker.max.id + 1 中生成的代理 ids start。

compression.type

Type: string
Default: producer
Valid Values: [uncompressed, zstd, lz4, snappy, gzip, producer]
Importance: high
Dynamic update: cluster-wide
指定给定主题的最终压缩类型。此配置接受标准压缩解码器('gzip', 'snappy', 'lz4', 'zstd')。它还接受等同于没有压缩的 'uncompressed'；而 'producer' 表示保留由 producer 设置的原始压缩 codec。

control.plane.listener.name

Type: string
Default: null
Importance: high
Dynamic update: read-only
用于控制器和代理之间的通信的监听程序名称。代理将使用 **control.plane.listener.name** 在监听器列表中找到端点，以侦听来自控制器的连接。例如，如果代理配置是： listener **= INTERNAL://192.1.1.8:9092, EXTERNAL://10.1.1.5:9093, CONTROLLER://192.1.1.8:9094listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSLcontrol.plane.listener.name = CONTROLLER** On startup, CONTROLLER On startup, broker will start listening on "1928:9094" with security protocol "SSL"在控制器一侧，当它通过 ZooKeeper 发现代理发布的端点时，它将使用 **control.plane.listener.name** 找到端点，它将用来与代理建立连接。例如，如果 ZooKeeper 上代理发布的端点有： **"endpoints" : ["INTERNAL://broker1.example.com:9092","EXTERNAL://broker1.example.com:9093","CONTROLLER://broker1.example.com:9094"]**，控制器的配置为： **listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSLcontrol.plane.listener.name = CONTROLLER**，控制器将使用 "broker1.example.com:9094" 和安全协议 "SSL" 连接到代理。如果没有显式配置，则默认值为 null，且没有控制器连接的专用端点。如果明确配置，则该值不能与 **inter.broker.listener.name** 的值相同。

controller.listener.names

Type: string
Default: null
Importance: high
Dynamic update: read-only
控制器使用的监听程序名称的逗号分隔列表。这在 KRaft 模式下运行时需要。与控制器仲裁通信时，代理总是使用这个列表中的第一个监听程序。注： 基于 ZooKeeper 的控制器不应设置此配置。

controller.quorum.election.backoff.max.ms

type: int
Default: 1000 (1 second)
Importance: high
Dynamic update: read-only
开始新选举前的最长时间（毫秒）。这在有助于防止网格锁定的二进制指数 backoff 机制中使用。

controller.quorum.election.timeout.ms

type: int
Default: 1000 (1 second)
Importance: high
Dynamic update: read-only
在触发新选举前，等待的最长时间（以毫秒为单位），而无需从领导中获取。

controller.quorum.fetch.timeout.ms

type: int
Default: 2000 (2 seconds)
Importance: high
Dynamic update: read-only
在成为候选人并触发票务选举前，没有从当前领导获得成功的最长时间；领导机在重新签名前可以从大多数仲裁接收有效的获取或获取Snapshot 请求。

controller.quorum.voters

type: list

Default: ""
Valid Values: non-empty list
Importance: high
Dynamic update: read-only
在以逗号分隔的 **{id}@{host}:{port}** 条目列表中，为一组 voters 的 id/endpoint 信息映射。例如：**1@localhost:9092,2@localhost:9093,3@localhost:9094**.

delete.topic.enable

Type: boolean
Default: true
Importance: high
Dynamic update: read-only
启用删除主题。如果关闭此配置，通过 admin 工具删除主题将无效。

early.start.listeners

Type: string
Default: null
Importance: high
Dynamic update: read-only
以逗号分隔的监听程序名称列表，这些名称可以在授权器完成初始化前启动。当授权器依赖于集群本身进行 bootstrap 时，这很有用，如 StandardAuthorizer （在元数据日志中存储 ACL 的情况） 默认情况下，controller.listener.names 中包含的所有监听程序也是早期启动监听程序。如果监听程序接受外部流量，则不应出现在此列表中。

eligible.leader.replicas.enable

Type: boolean
Default: false
Importance: high
Dynamic update: read-only
启用 Eligible leader 副本。

leader.imbalance.check.interval.seconds

Type: long
Default: 300
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
控制器触发分区重新平衡检查的频率。

leader.imbalance.per.broker.percentage

Type: int
Default: 10
Importance: high
Dynamic update: read-only
每个代理允许的 leader 的比率。如果每个代理超过这个值，控制器会触发领导平衡。该值以百分比为单位。

监听器

Type: string
Default: PLAINTEXT://:9092
Importance: high
Dynamic update: per-broker

侦听器列表 - 我们将侦听的 URI 列表和侦听器名称。如果侦听器名称不是安全协议，还必须设置 **listener.security.protocol.map**。侦听器名称和端口号必须是唯一的，除非一个监听器是 IPv4 地址，另一个监听器是 IPv6 地址（用于同一端口）。将 hostname 指定为 0.0.0.0 以绑定到所有接口。将 hostname 留空，以绑定到默认接口。法律侦听器列表示例：**PLAINTEXT://myhost:9092,SSL://:9091 CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093 PLAINTEXT://127.0.0.1:9092,SSL://[::1]:9092**.

log.dir

type: string
Default: /tmp/kafka-logs
Importance: high
Dynamic update: read-only
保存日志数据的目录（为 log.dirs 属性提供）。

log.dirs

Type: string
Default: null
Importance: high
Dynamic update: read-only
存储日志数据的目录的逗号分隔列表。如果没有设置，则使用 log.dir 的值。

log.flush.interval.messages

type: long
Default: 9223372036854775807
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
在消息刷新到磁盘前，日志分区上积累的消息数量。

log.flush.interval.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
在刷新到磁盘前，任何主题中的消息保存在内存中的最长时间(ms)。如果没有设置，则使用 log.flush.scheduler.interval.ms 中的值。

log.flush.offset.checkpoint.interval.ms

type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
更新最后一次刷新的持久记录的频率，该记录充当日志恢复点。

log.flush.scheduler.interval.ms

type: long
Default: 9223372036854775807
Importance: high
Dynamic update: read-only
日志清除程序的频率(ms)检查是否需要清空到磁盘中的日志。

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

type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
我们更新日志启动偏移的永久记录的频率。

log.retention.bytes

Type: long
Default: -1
Importance: high
Dynamic update: cluster-wide
删除日志前的最大大小。

log.retention.hours

type: int
Default: 168
Importance: high
Dynamic update: read-only
删除日志文件前的小时数（以小时为单位），以及 log.retention.ms 属性的几小时数。

log.retention.minutes

type: int
Default: null
Importance: high
Dynamic update: read-only
在删除日志文件前（以分钟为单位）保留日志文件的分钟数，次要到 log.retention.ms 属性。如果没有设置，则使用 log.retention.hours 中的值。

log.retention.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
删除日志文件前的毫秒数（以毫秒为单位），如果没有设置，则使用 log.retention.minutes 的值。如果设置为 -1，则不会应用时间限制。

log.roll.hours

type: int
Default: 168
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
推出新日志段前的最长时间（以小时为单位），二级到 log.roll.ms 属性。

log.roll.jitter.hours

type: int
Default: 0
Valid Values: [0,...]

Importance: high
Dynamic update: read-only
从 logRollTimeMillis （以小时为单位）中减去的最大 jitter，次要到 log.roll.jitter.ms 属性。

log.roll.jitter.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
从 logRollTimeMillis （以毫秒为单位）中减去的最大 jitter。如果没有设置，则使用 log.roll.jitter.hours 中的值。

log.roll.ms

Type: long
Default: null
Importance: high
Dynamic update: cluster-wide
推出新日志段前的最长时间（以毫秒为单位）。如果没有设置，则使用 log.roll.hours 中的值。

log.segment.bytes

type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [14,...]
Importance: high
Dynamic update: cluster-wide
单个日志文件的最大大小。

log.segment.delete.delay.ms

type: long
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: cluster-wide
从文件系统中删除文件前等待的时间。

message.max.bytes

type: int
Default: 1048588
Valid Values: [0,...]
Importance: high
Dynamic update: cluster-wide
Kafka 允许的最大记录批处理大小（如果启用了压缩，在压缩后进行压缩）。如果这被增加，且有 0.10.2 旧的消费者，还必须增加用户的获取大小，以便他们可以获取记录批处理。在最新的消息格式版本中，记录始终分组到批处理中，以获得效率。在以前的消息格式版本中，未压缩记录没有分组到批处理中，在这种情况下，这个限制仅适用于单个记录。可以使用主题级别 **max.message.bytes** 配置为每个主题设置。

metadata.log.dir

Type: string
Default: null
Importance: high
Dynamic update: read-only

此配置决定了将集群元数据日志放在 KRaft 模式的位置。如果没有设置，则元数据日志会放置在 log.dirs 的第一个日志目录中。

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

type: long
Default: 20971520
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
这是生成新快照前日志的最大字节数，以及生成新快照前所需的高水位线之间的最大字节数。默认值为 20971520。要根据经过的时间生成快照，请查看 **metadata.log.max.snapshot.interval.ms** 配置。当达到最大时间间隔或达到最大字节数时，Kafka 节点将生成一个快照。

metadata.log.max.snapshot.interval.ms

type: long
Default: 3600000 (1 hour)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
如果日志中没有包括在最新快照中的记录，这是等待生成快照的最大毫秒数。值为零禁用基于时间的快照生成。默认值为 3600000。要根据元数据字节数生成快照，请参阅 **metadata.log.max.record.bytes.between.snapshots** 配置。当达到最大时间间隔或达到最大字节数时，Kafka 节点将生成一个快照。

metadata.log.segment.bytes

type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [12,...]
Importance: high
Dynamic update: read-only
单个元数据日志文件的最大大小。

metadata.log.segment.ms

Type: long
Default: 604800000 (7 days)
Importance: high
Dynamic update: read-only
推出新元数据日志文件前的最长时间（以毫秒为单位）。

metadata.max.retention.bytes

Type: long
Default: 104857600 (100 mebibytes)
Importance: high
Dynamic update: read-only
在删除旧快照和日志文件前，元数据日志和快照的最大组合大小。由于在任何日志可以被删除之前，至少有一个快照必须存在，所以这是一个软限制。

metadata.max.retention.ms

Type: long
Default: 604800000 (7 days)
Importance: high
Dynamic update: read-only

删除元数据日志文件或快照前的毫秒数。由于在任何日志可以被删除之前，至少有一个快照必须存在，所以这是一个软限制。

min.insync.replicas

type: int
Default: 1
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
当制作者将 acks 设为 "all" （或 "-1"），**min.insync.replicas** 指定必须确认写入的最少副本数才被视为成功。如果无法满足此最小值，则生成者将引发异常(**NotEnoughReplicas** 或 **NotEnoughReplicasAfterAppend**)。当一起使用时，**min.insync.replicas** 和 acks 允许您强制实施更大的持久性保证。典型的场景是创建带有复制因子 3 的主题，将 **min.insync.replicas** 设置为 2，并使用 acks of "all" 生成。这将确保如果大多数副本都未收到写入，则生成者引发异常。

node.id

type: int
Default: -1
Importance: high
Dynamic update: read-only
与此进程时所扮演的角色关联的节点 ID 。**roles** 是非空的。这在以 KRaft 模式运行时是必需的配置。

num.io.threads

type: int
Default: 8
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
服务器用来处理请求的线程数量，其中可能包括磁盘 I/O。

num.network.threads

type: int
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
服务器用来从网络接收请求并将响应发送到网络的线程数量。注： 每个监听程序（控制器监听程序除外）创建自己的线程池。

num.recovery.threads.per.data.dir

type: int
Default: 1
Valid Values: [1,...]
Importance: high
Dynamic update: cluster-wide
在启动时用于日志恢复的每个数据目录的线程数量，并在关闭时清除。

num.replica.alter.log.dirs.threads

type: int
Default: null
Importance: high
Dynamic update: read-only

可在日志目录之间移动副本的线程数量，其中可能包括磁盘 I/O。

num.replica.fetchers

Type: int
Default: 1
Importance: high
Dynamic update: cluster-wide
用于从每个源代理复制记录的获取器线程数量。每个代理上的获取者总数由 **num.replica.fetchers** 乘以集群中的代理数量。这个值可以增加后续 CPU 和内存利用率中的 I/O 并行性程度。

offset.metadata.max.bytes

Type: int
Default: 4096 (4 kibibytes)
Importance: high
Dynamic update: read-only
与偏移提交关联的元数据条目的最大大小。

offsets.commit.required.acks

Type: short
Default: -1
Importance: high
Dynamic update: read-only
在可以接受提交前所需的 acks。通常，默认(-1)不应被覆盖。

offsets.commit.timeout.ms

type: int
Default: 5000 (5 seconds)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
偏移提交会延迟，直到 offsets 主题的所有副本都接收提交或达到此超时为止。这与制作者请求超时类似。

offsets.load.buffer.size

type: int
Default: 5242880
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
将偏移加载到缓存中时从偏移段读取的批处理大小（如果记录太大，则覆盖）。

offsets.retention.check.interval.ms

type: long
Default: 600000 (10 minutes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
检查过时偏移的频率。

offsets.retention.minutes

type: int

Default: 10080
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
对于订阅的用户，提交特定分区的误差将在 1 后过期并丢弃，当消费者组丢失了所有消费者（即变得为空）后，这个保留周期将会被过期并丢弃；2）此保留周期已经过，因为偏移提交了最后一次的时间，组不再订阅相应的主题。对于独立的消费者（使用手动分配），自上次提交的时间起，偏移将在保留周期后过期。请注意，当通过 delete-group 请求删除组时，其提交的偏移也会在没有额外的保留周期的情况下删除。另外，当一个主题通过 delete-topic 请求删除时，当传播元数据更新任何组的提交偏移时，也会删除该主题的提交偏移，而无需额外的保留周期。

offsets.topic.compression.codec

Type: int
Default: 0
Importance: high
Dynamic update: read-only
偏移主题的压缩 codec - 压缩可用于实现"原子"提交。

offsets.topic.num.partitions

type: int
Default: 50
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
偏移提交主题的分区数量（部署后不应更改）。

offsets.topic.replication.factor

Type: short
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
偏移主题的复制因素（设置更高以确保可用性）。内部主题创建将失败，直到集群大小满足此复制因素要求。

offsets.topic.segment.bytes

type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
偏移主题片段字节数应该保持相对小，以便加快日志压缩和缓存负载。

process.roles

type: list
Default: ""
Valid Values: [broker, controller]
Importance: high
Dynamic update: read-only
此进程 play 的角色： 'broker'、'controller' 或 'broker,controller'（如果两者都是）。此配置仅适用于 KRaft (Kafka Raft)模式（而不是 ZooKeeper）中的集群。为 ZooKeeper 集群保留此配置未定义或为空。

queued.max.requests

type: int
Default: 500
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
在阻止网络线程前，允许 data-plane 的已排队请求数。

replica.fetch.min.bytes

Type: int
Default: 1
Importance: high
Dynamic update: read-only
每个获取响应的预期最小字节数。如果没有足够字节，请等待 **replica.fetch.wait.max.ms** (broker config)。

replica.fetch.wait.max.ms

Type: int
Default: 500
Importance: high
Dynamic update: read-only
由后续副本发布的每个获取者请求的最大等待时间。这个值应该始终小于 replica.lag.time.max.ms，以防止频繁缩小 ISR 用于低吞吐量主题。

replica.high.watermark.checkpoint.interval.ms

Type: long
Default: 5000 (5 seconds)
Importance: high
Dynamic update: read-only
将高水位线保存到磁盘的频率。

replica.lag.time.max.ms

type: long
Default: 30000 (30 秒)
Importance: high
Dynamic update: read-only
如果后续者没有发送任何获取请求，或者还没有消耗领导日志结束偏移，则领导机将从 isr 中删除后续者。

replica.socket.receive.buffer.bytes

type: int
Default: 65536 (64 kibibytes)
Importance: high
Dynamic update: read-only
到复制数据的领导的套接字接收缓冲区。

replica.socket.timeout.ms

type: int
Default: 30000 (30 秒)
Importance: high
Dynamic update: read-only

网络请求的套接字超时。其值应至少为 replica.fetch.wait.max.ms。

request.timeout.ms

type: int
Default: 30000 (30 秒)
Importance: high
Dynamic update: read-only
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

sasl.mechanism.controller.protocol

Type: string
Default: GSSAPI
Importance: high
Dynamic update: read-only
用于与控制器通信的 SASL 机制。默认为 GSSAPI。

socket.receive.buffer.bytes

type: int
Default: 102400 (100 kibibytes)
Importance: high
Dynamic update: read-only
套接字服务器套接字的 SO_RCVBUF 缓冲。如果值为 -1，则使用操作系统默认值。

socket.request.max.bytes

type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
套接字请求中的最大字节数。

socket.send.buffer.bytes

type: int
Default: 102400 (100 kibibytes)
Importance: high
Dynamic update: read-only
套接字服务器套接字的 SO_SNDBUF 缓冲区。如果值为 -1，则使用操作系统默认值。

transaction.max.timeout.ms

type: int
Default: 900000 (15 minutes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
事务允许的最大超时时间。如果客户端请求的事务时间超过这个值，则代理将在 InitProducerIdRequest 中返回错误。这可防止客户端太大的超时时间，这可能会使消费者从事务中包含的主题中读取。

transaction.state.log.load.buffer.size

type: int

Default: 5242880
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
将制作者 ID 和事务加载到缓存中时从事务日志片段读取的批处理大小（如果记录太大，则覆盖）。

transaction.state.log.min.isr

type: int
Default: 2
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
覆盖 min.insync.replicas 配置，用于事务主题。

transaction.state.log.num.partitions

type: int
Default: 50
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
事务主题的分区数量（部署后不应更改）。

transaction.state.log.replication.factor

Type: short
Default: 3
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
事务主题的复制因素（设置更高），以确保可用性。内部主题创建将失败，直到集群大小满足此复制因素要求。

transaction.state.log.segment.bytes

type: int
Default: 104857600 (100 mebibytes)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
事务主题片段字节数应该保持相对小，以便加快日志压缩和缓存负载。

transactional.id.expiration.ms

type: int
Default: 604800000 (7 days)
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
事务协调器将在不接收当前事务状态更新前等待的时间（毫秒）。当事务仍在持续时，事务 ID 不会过期。

unclean.leader.election.enable

Type: boolean
Default: false
Importance: high

Dynamic update: cluster-wide
指明是否启用 ISR 集中的副本作为最后的手段，即使这样做可能会导致数据丢失。

zookeeper.connect

Type: string
Default: null
Importance: high
Dynamic update: read-only
以 **hostname:port** 指定 ZooKeeper 连接字符串，其中 host 和 port 是 ZooKeeper 服务器的主机和端口。当 ZooKeeper 机器停机时，要允许连接到其他 ZooKeeper 节点，您还可以以 **hostname1:port1,hostname2:port2,hostname3:port3** 格式指定多个主机。服务器也可以具有 ZooKeeper chroot 路径作为其 ZooKeeper 连接字符串的一部分，将其数据置于全局 ZooKeeper 命名空间中的一些路径下。例如，要提供 chroot 路径 **/chroot/path**，您可以将连接字符串指定为 **hostname1:port1,hostname2:port2,hostname3:port3/chroot/path**。

zookeeper.connection.timeout.ms

type: int
Default: null
Importance: high
Dynamic update: read-only
客户端等待建立与 ZooKeeper 的连接的最大时间。如果没有设置，则使用 zookeeper.session.timeout.ms 中的值。

zookeeper.max.in.flight.requests

type: int
Default: 10
Valid Values: [1,...]
Importance: high
Dynamic update: read-only
在阻止前，客户端将发送到 ZooKeeper 的最大未确认的请求数量。

zookeeper.metadata.migration.enable

Type: boolean
Default: false
Importance: high
Dynamic update: read-only
启用 ZK 到 KRaft 迁移。

zookeeper.session.timeout.ms

Type: int
Default: 18000 (18 seconds)
Importance: high
Dynamic update: read-only
ZooKeeper 会话超时。

zookeeper.set.acl

Type: boolean
Default: false
Importance: high
Dynamic update: read-only
将 client 设置为使用安全 ACL。

broker.heartbeat.interval.ms

type: int
Default: 2000 (2 seconds)
Importance: medium
Dynamic update: read-only
代理心跳之间时间长度（以毫秒为单位）。在 KRaft 模式下运行时使用。

broker.id.generation.enable

type: boolean
Default: true
Importance: medium
Dynamic update: read-only
在服务器上启用自动代理 ID 生成。启用为 reserved.broker.max.id 配置的值时，应检查。

broker.rack

Type: string
Default: null
Importance: medium
Dynamic update: read-only
代理的机架。这将用于机架感知为容错的复制分配。示例： **RACK1、 us-east-1d**.

broker.session.timeout.ms

type: int
Default: 9000 (9 seconds)
Importance: medium
Dynamic update: read-only
没有心跳时代理租期最后一次的时间长度（以毫秒为单位）。在 KRaft 模式下运行时使用。

connections.max.idle.ms

Type: long
Default: 600000 (10 minutes)
Importance: medium
Dynamic update: read-only
闲置连接超时：服务器套接字处理器线程关闭闲置超过这个连接。

connections.max.reauth.ms

Type: long
Default: 0
Importance: medium
Dynamic update: read-only
当明确设置为正数（默认值为 0，而不是正数）时，不会超过配置的值的会话生命周期会在验证时与 v2.2.0 或更高版本的客户端进行通信。代理将断开在会话生命周期中未重新验证的任何这样的连接，然后用于除重新身份验证以外的任何目的。配置名称可以选择在小写中使用监听程序前缀和 SASL 机制名称作为前缀。例如，listener.name.sasl_ssl.oauthbearer.connections.max.reauth.ms=3600000。

controlled.shutdown.enable

type: boolean
Default: true
Importance: medium
Dynamic update: read-only
启用服务器的受控关闭。

controlled.shutdown.max.retries

type: int
Default: 3
Importance: medium
Dynamic update: read-only
受控关闭可能会因为多个原因失败。这决定了发生此类失败时的重试次数。

controlled.shutdown.retry.backoff.ms

Type: long
Default: 5000 (5 seconds)
Importance: medium
Dynamic update: read-only
每次重试前，系统需要从导致之前失败的状态中恢复(Controller 故障转移、副本滞后等)。此配置决定了重试前等待的时间。

controller.quorum.append.linger.ms

type: int
Default: 25
Importance: medium
Dynamic update: read-only
领导领导将等待写入的时间（以毫秒为单位），然后再将它们刷新到磁盘。

controller.quorum.request.timeout.ms

type: int
Default: 2000 (2 seconds)
Importance: medium
Dynamic update: read-only
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

controller.socket.timeout.ms

type: int
Default: 30000 (30 秒)
Importance: medium
Dynamic update: read-only
controller-to-broker 频道的套接字超时。

default.replication.factor

Type: int
Default: 1
Importance: medium
Dynamic update: read-only
自动创建主题的默认复制因素。

delegation.token.expiry.time.ms

type: long
Default: 86400000 (1 day)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
需要续订令牌前的令牌有效期（以毫秒为单位）。默认值为 1 天。

delegation.token.master.key

Type: password
Default: null
Importance: medium
Dynamic update: read-only
DEPRECATED ： delegation.token.secret.key 的别名，它应该被使用而不是此配置。

delegation.token.max.lifetime.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
令牌具有最长生命周期，其无法再续订。默认值为 7 天。

delegation.token.secret.key

Type: password
Default: null
Importance: medium
Dynamic update: read-only
生成和验证委托令牌的 secret 密钥。必须在所有代理间配置相同的密钥。如果使用 KRaft 的 Kafka，还必须在所有控制器中设置密钥。如果密钥未设置或设置为空字符串，代理将禁用委派令牌支持。

delete.records.purgatory.purge.interval.requests

Type: int
Default: 1
Importance: medium
Dynamic update: read-only
删除记录的清除间隔（请求数）

fetch.max.bytes

type: int
Default: 57671680 (55 mebibytes)
Valid Values: [1024,...]
Importance: medium
Dynamic update: read-only
我们将返回获取请求的最大字节数。必须至少为 1024。

fetch.purgatory.purge.interval.requests

type: int
Default: 1000
Importance: medium
Dynamic update: read-only
获取请求的清除间隔（请求数）。

group.consumer.assignors

Type: list
Default:
org.apache.kafka.coordinator.group.assignor.UniformAssignor,org.apache.kafka.coordinator.group.assig
Importance: medium
Dynamic update: read-only

服务器端分配器作为完整类名称列表。如果消费者没有指定分配器，则列表中的第一个被视为要使用的默认分配器。

group.consumer.heartbeat.interval.ms

type: int
Default: 5000 (5 seconds)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
提供给消费者组成员的心跳间隔。

group.consumer.max.heartbeat.interval.ms

type: int
Default: 15000 (15 seconds)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
注册的消费者的最大心跳间隔。

group.consumer.max.session.timeout.ms

type: int
Default: 60000 (1 minute)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
注册消费者允许的最大会话超时。

group.consumer.max.size

type: int
Default: 2147483647
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
单个消费者组可以容纳的最大消费者数量。

group.consumer.min.heartbeat.interval.ms

type: int
Default: 5000 (5 seconds)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
注册的用户的最小心跳间隔。

group.consumer.min.session.timeout.ms

type: int
Default: 45000 (45 seconds)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
注册使用者允许的最小会话超时。

group.consumer.session.timeout.ms

type: int
Default: 45000 (45 seconds)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
使用消费者组协议时检测客户端故障的超时。

group.coordinator.rebalance.protocols

type: list
Default: classic
Valid Values: [consumer, classic]
Importance: medium
Dynamic update: read-only
启用重新平衡协议列表。支持的协议：消费者、经典.消费者重新平衡协议处于早期访问，因此不得在生产环境中使用。

group.coordinator.threads

type: int
Default: 1
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
组协调器使用的线程数量。

group.initial.rebalance.delay.ms

type: int
Default: 3000 (3 秒)
Importance: medium
Dynamic update: read-only
在执行第一次重新平衡前，组协调器将等待更多消费者加入新组的时间。较长的延迟意味着重新平衡可能会减少，但会增加处理开始的时间。

group.max.session.timeout.ms

Type: int
Default: 1800000 (30 minutes)
Importance: medium
Dynamic update: read-only
注册消费者允许的最大会话超时。超时时间较长，可让消费者在心跳之间处理消息的时间较长，以检测故障的时间。

group.max.size

type: int
Default: 2147483647
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
单个消费者组可以容纳的最大消费者数量。

group.min.session.timeout.ms

type: int
Default: 6000 (6 seconds)
Importance: medium

Dynamic update: read-only
注册使用者允许的最小会话超时。超时时间较短，会导致在更频繁的消费者心跳的情况下更快地检测失败，这可能会造成代理资源的问题。

initial.broker.registration.timeout.ms

type: int
Default: 60000 (1 minute)
Importance: medium
Dynamic update: read-only
最初注册控制器仲裁时，在声明失败并退出代理进程前要等待的毫秒数。

inter.broker.listener.name

Type: string
Default: null
Importance: medium
Dynamic update: read-only
用于代理间通信的监听程序名称。如果未设置，监听程序名称由 security.inter.broker.protocol 定义。同时设置这个和 security.inter.broker.protocol 属性是一个错误。

inter.broker.protocol.version

Type: string
Default: 3.7-IV4
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, 1.1-IV0, 1.1-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, 3.0-iv0, 3.0-iv1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.3-IV0, 3.5-IV0, 3.5-IV1, 3.5-IV2, 3.6-IV0, 3.6-IV1, 3.6-IV2, 3.7-IV0, 3.7-IV1, 3.7-IV2, 3.7-IV3, 3.7-IV4, 3.8-IV4, 3.8-IV0]
Importance: medium
Dynamic update: read-only
指定要使用的 inter-broker 协议的版本。这通常会在所有代理都升级到新版本后被禁止。一些有效值的示例为 0.8.0, 0.8.1, 0.8.1.1, 0.8.2, 0.8.2, 0.8.2.1, 0.9.0.0, 0.9.0.0, 0.9.0.1 Check MetadataVersion for the full list。

log.cleaner.backoff.ms

type: long
Default: 15000 (15 秒)
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
没有日志清理时休眠的时间长度。

log.cleaner.dedupe.buffer.size

Type: long
Default: 134217728
Importance: medium
Dynamic update: cluster-wide
在所有清理线程中用于日志 deduplication 的总内存。

log.cleaner.delete.retention.ms

type: long
Default: 86400000 (1 day)

Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
为日志压缩主题保留 tombstone 消息标记的时间长度。此设置也会在消费者从偏移 0 开始读取的时间绑定，以确保它们获得最终阶段的有效快照（在消费者完成扫描前可能会收集它们）。

log.cleaner.enable

type: boolean
Default: true
Importance: medium
Dynamic update: read-only
启用日志清理程序，以在服务器上运行。如果使用包含 cleanup.policy=compact 的任何主题，则应启用，包括内部偏移主题。如果禁用了这些主题，则不会压缩并持续增大。

log.cleaner.io.buffer.load.factor

type: double
Default: 0.9
Importance: medium
Dynamic update: cluster-wide
Log cleaner deduer dedupe 缓冲区负载因素。去除重复缓冲区的百分比可能会变为。较高的值将允许一次清理更多日志，但会导致更多的哈希冲突。

log.cleaner.io.buffer.size

type: int
Default: 524288
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
在所有清理线程中用于日志清理 I/O 缓冲区的总内存。

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

type: double
Default: 1.7976931348623157E308
Importance: medium
Dynamic update: cluster-wide
日志清理器将被节流，其 read 和 write i/o 的总和将小于这个值。

log.cleaner.max.compaction.lag.ms

type: long
Default: 9223372036854775807
Valid Values: [1,...]
Importance: medium
Dynamic update: cluster-wide
消息在日志中保留无法压缩的最长时间。仅适用于被压缩的日志。

log.cleaner.min.cleanable.ratio

type: double
Default: 0.5
Valid Values: [0,...,1]
Importance: medium
Dynamic update: cluster-wide
脏日志的最小比率，达到日志的总日志，以满足清理的要求。如果还指定了

log.cleaner.max.compaction.lag.ms 或 log.cleaner.min.compaction.lag.ms 配置，则日志紧凑器会在 log.cleaner.minaction.minaction.gms 期间内马上考虑压缩条件。 或(ii)如果日志在 log.cleaner.max.compaction.lag.ms 期间内有脏(uncompacted)记录。

log.cleaner.min.compaction.lag.ms

Type: long
Default: 0
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
消息在日志中保持 uncompacted 的最短时间。仅适用于被压缩的日志。

log.cleaner.threads

type: int
Default: 1
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
用于日志清理的后台线程数量。

log.cleanup.policy

type: list
Default: delete
Valid Values: [compact, delete]
Importance: medium
Dynamic update: cluster-wide
除了保留窗口外的片段的默认清理策略。以逗号分隔的有效策略列表。有效策略包括："delete" 和 "compact"。

log.index.interval.bytes

Type: int
Default: 4096 (4 kibibytes)
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
我们向偏移索引中添加条目的时间间隔。

log.index.size.max.bytes

type: int
Default: 10485760 (10 mebibytes)
Valid Values: [4,...]
Importance: medium
Dynamic update: cluster-wide
偏移索引的最大大小（以字节为单位）。

log.local.retention.bytes

type: long
Default: -2
Valid Values: [-2,...]
Importance: medium
Dynamic update: cluster-wide

在分区有资格删除前可增大的本地日志片段的最大大小。默认值为 -2，它代表要使用的 **log.retention.bytes** 值。有效的值应始终小于或等于 **log.retention.bytes** 值。

log.local.retention.ms

type: long
Default: -2
Valid Values: [-2,...]
Importance: medium
Dynamic update: cluster-wide
在进行删除前保留本地日志片段的毫秒数。默认值为 -2，它代表 **log.retention.ms** 值被使用。有效的值应始终小于或等于 **log.retention.ms** 值。

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, 1.1-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, 3.0-iv0, 3.0-iv1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.3-IV0, 3.5-IV0, 3.5-IV1, 3.5-IV2, 3.6-IV0, 3.6-IV1, 3.6-IV2, 3.7-IV0, 3.7-IV1, 3.7-IV2, 3.7-IV3, 3.7-IV4, 3.8-IV4, 3.8-IV0]
Importance: medium
Dynamic update: read-only
指定代理用来将消息附加到日志中的消息格式版本。该值应该是有效的 MetadataVersion。一些示例为 0.8.2, 0.9.0.0, 0.10.0，检查 MetadataVersion 获取更多详细信息。通过设置特定的消息格式版本，用户认证磁盘上所有现有的消息是否都小于或等于指定版本。错误地设置这个值将导致具有旧版本的用户中断，因为它们会收到一个不理解的格式的消息。

log.message.timestamp.after.max.ms

type: long
Default: 9223372036854775807
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
此配置设定了消息时间戳和代理时间戳之间的允许时间戳差异。消息时间戳可能高于代理的时间戳，最大允许差别由此配置中设置的值决定。如果 log.message.timestamp.type=CreateTime，如果时间戳的差异超过这个指定的阈值，则消息将被拒绝。如果 log.message.timestamp.type=LogAppendTime，则忽略此配置。

log.message.timestamp.before.max.ms

type: long
Default: 9223372036854775807
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
此配置设定了代理的时间戳和消息时间戳之间的允许时间戳差异。消息时间戳可以早于或等于代理的时间戳，其最大允许区别由此配置中设置的值决定。如果 log.message.timestamp.type=CreateTime，如果时间戳的差异超过这个指定的阈值，则消息将被拒绝。如果 log.message.timestamp.type=LogAppendTime，则忽略此配置。

log.message.timestamp.difference.max.ms

type: long
Default: 9223372036854775807

Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
[DEPRECATED] 代理收到消息和消息中指定的时间戳之间允许的最大区别。如果 log.message.timestamp.type=CreateTime，如果时间戳的差异超过这个阈值，则会被拒绝。如果 log.message.timestamp.type=LogAppendTime。允许的最大时间戳差异不应大于 log.retention.ms，以避免不必要频繁的日志滚动。

log.message.timestamp.type

Type: string
Default: CreateTime
Valid Values: [CreateTime, LogAppendTime]
Importance: medium
Dynamic update: cluster-wide
定义消息中的时间戳是消息创建时间还是日志附加时间。该值应该是 **CreateTime** 或 **LogAppendTime**。

log.preallocate

Type: boolean
Default: false
Importance: medium
Dynamic update: cluster-wide
创建新段时，应预先分配文件？如果您在 Windows 上使用 Kafka，您可能需要将其设置为 true。

log.retention.check.interval.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
日志清理程序的频率（以毫秒为单位）检查任何日志是否有资格删除。

max.connection.creation.rate

type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
我们随时在代理中允许的最大连接创建率。也可以通过使用监听器前缀前缀配置监听程序级别的限制，如 **listener.name.internal.max.connection.creation.rate**.Broker-wide 连接速率限制，但应该根据应用程序要求配置监听程序限制。如果达到监听器或代理限制，则新连接将节流，但 inter-broker 监听程序除外。只有在达到监听器级别速率限制时，才会节流 inter-broker 侦听器上的连接。

max.connections

type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
我们随时允许的最大连接数。除了使用 max.connections.per.ip 配置的任何每个ip 限制外，还会应用这个限制。也可以通过使用监听程序前缀前缀（如 listener.name. **internal.max.connections** ）为**配置监听程序** 级别的限制来配置。代理范围限制应该基于代理容量，而监听器限制则应该根据应用程序要

求进行配置。如果达到监听程序或代理限制，则阻止新的连接。即使达到代理范围限制，也允许 inter-broker 侦听器上的连接。在这种情况下，其他监听器中最早使用的连接将关闭。

max.connections.per.ip

type: int
Default: 2147483647
Valid Values: [0,...]
Importance: medium
Dynamic update: cluster-wide
我们允许从每个 ip 地址允许的最大连接数。如果使用 max.connections.per.ip.overrides 属性配置覆盖，则可以将其设置为 0。如果达到限制，则将丢弃来自 ip 地址的新连接。

max.connections.per.ip.overrides

Type: string
Default: ""
Importance: medium
Dynamic update: cluster-wide
每个ip 或 hostname 的逗号分隔列表覆盖默认最大连接数。示例值为 "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
我们将维护的最大增量获取会话数量。

num.partitions

type: int
Default: 1
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
每个主题的默认日志分区数量。

password.encoder.old.secret

Type: password
Default: null
Importance: medium
Dynamic update: read-only
用于动态配置的编码密码的旧 secret。这只在更新 secret 时才需要。如果指定，所有动态编码的密码都使用这个旧 secret 进行解码，并在代理启动时使用 password.encoder.secret 重新编码。

password.encoder.secret

Type: password
Default: null
Importance: medium
Dynamic update: read-only
用于为这个代理动态配置的编码密码的 secret。

principal.builder.class

type: class
Default: org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalBuilder
Importance: medium
Dynamic update: per-broker
实现 KafkaPrincipalBuilder 接口的类的完全限定名称，用于构建授权期间使用的 KafkaPrincipal 对象。如果没有定义主体构建器，则默认行为取决于所使用的安全协议。对于 SSL 身份验证，主体将使用由与客户端证书区分名称（如果提供）的 **ssl.principal.mapping.rules** 定义的规则派生。否则，如果不需要客户端身份验证，则主体名称为 ANONYMOUS。对于 SASL 身份验证，如果 GSSAPI 正在使用中，则主体将使用 **sasl.kerberos.principal.to.local.rules** 定义的规则派生，以及其它机制的 SASL 身份验证 ID。对于 PLAINTEXT，主体将是 ANONYMOUS。

producer.purgatory.purge.interval.requests

type: int
Default: 1000
Importance: medium
Dynamic update: read-only
生成者请求的清除间隔（请求数）。

queued.max.request.bytes

Type: long
Default: -1
Importance: medium
Dynamic update: read-only
在没有读取更多请求前允许的排队字节数。

remote.log.manager.thread.pool.size

type: int
Default: 10
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
调度任务用于复制片段的线程池的大小，获取远程日志索引并清理远程日志片段。

remote.log.metadata.manager.class.name

Type: string
Default:
org.apache.kafka.server.log.remote.metadata.storage.TopicBasedRemoteLogMetadataManager
Valid Values: non-empty string
Importance: medium
Dynamic update: read-only
RemoteLogMetadataManager 实现的完全限定域名。

remote.log.metadata.manager.class.path

Type: string
Default: null
Importance: medium
Dynamic update: read-only
RemoteLogMetadataManager 实现的类路径。如果指定，则 RemoteLogMetadataManager 实现及其依赖库将由专用的类加载加载，该加载器将在 Kafka 代理类路径前搜索此类路径。此参数的语法与标准 Java 类路径字符串相同。

remote.log.metadata.manager.impl.prefix

type: string
Default: rlmm.config.
Valid Values: non-empty string
Importance: medium
Dynamic update: read-only
用于传递给 RemoteLogMetadataManager 实现的属性的前缀。例如，这个值可以是 **rlmm.config**。

remote.log.metadata.manager.listener.name

type: string
Default: null
Valid Values: non-empty string
Importance: medium
Dynamic update: read-only
RemoteLogMetadataManager 实现需要的本地代理监听程序名称。

remote.log.reader.max.pending.tasks

type: int
Default: 100
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
最大远程日志读取器线程池任务队列大小。如果任务队列已满，则提供获取请求并显示错误。

remote.log.reader.threads

type: int
Default: 10
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
为处理远程日志读取而分配的线程池大小。

remote.log.storage.manager.class.name

type: string
Default: null
Valid Values: non-empty string
Importance: medium
Dynamic update: read-only
RemoteStorageManager 实现的完全限定类名称。

remote.log.storage.manager.class.path

Type: string
Default: null
Importance: medium
Dynamic update: read-only
RemoteStorageManager 实现的类路径。如果指定，RemoteStorageManager 实现及其依赖库将由专用的类加载加载，该加载器会在 Kafka 代理类路径前搜索此类路径。此参数的语法与标准 Java 类路径字符串相同。

remote.log.storage.manager.impl.prefix

type: string
Default: rsm.config.
Valid Values: non-empty string

Importance: medium
Dynamic update: read-only
用于传递给 RemoteStorageManager 实现的属性的前缀。例如，这个值可以是 **rsm.config**。

remote.log.storage.system.enable

Type: boolean
Default: false
Importance: medium
Dynamic update: read-only
是否在代理中启用分层存储功能。有效值为 **true** 或 **false**，默认值为 false。当为 true 代理时，代理启动分层存储功能所需的所有服务。

replica.fetch.backoff.ms

type: int
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
获取分区错误时休眠的时间长度。

replica.fetch.max.bytes

type: int
Default: 1048576 (1 mebibyte)
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
尝试为每个分区获取的信息字节数。这不是绝对的最大值，如果获取的第一个非空分区中的第一个记录批处理大于这个值，则记录批处理仍会返回，以确保可以进行进度。代理接受的最大记录批处理大小通过 **message.max.bytes** (broker config)或 **max.message.bytes** (topic config)定义。

replica.fetch.response.max.bytes

type: int
Default: 10485760 (10 mebibytes)
Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
整个获取响应预期的最大字节数。记录批量获取，如果获取的第一个非空分区中的第一个记录批处理大于这个值，则记录批处理仍会返回，以确保可以进行进度。因此，这不是绝对最大值。代理接受的最大记录批处理大小通过 **message.max.bytes** (broker config)或 **max.message.bytes** (topic config)定义。

replica.selector.class

Type: string
Default: null
Importance: medium
Dynamic update: read-only
实现 ReplicaSelector 的完全限定类名称。代理使用它来查找首选读取副本。默认情况下，我们使用返回领导的实施。

reserved.broker.max.id

type: int
Default: 1000

Valid Values: [0,...]
Importance: medium
Dynamic update: read-only
可用于 broker.id 的最大数量。

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
实现 AuthenticateCallbackHandler 接口的 SASL 客户端回调处理程序类的完全限定名称。

sasl.enabled.mechanisms

type: list
Default: GSSAPI
Importance: medium
Dynamic update: per-broker
Kafka 服务器中启用的 SASL 机制列表。列表中可以包含安全提供程序可用的机制。只有 GSSAPI 会被默认启用。

sasl.jaas.config

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
用于 SASL 连接的 JAAS 登录上下文参数，其格式供 JAAS 配置文件使用。JAAS 配置文件格式描述在此处。该值的格式是： **loginModuleClass controlFlag (optionName=optionValue)*;**。对于代理，配置必须在小写中带有监听前缀和 SASL 机制名称前缀。例如： 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 命令路径。

sasl.kerberos.min.time.before.relogin

type: long
Default: 60000
Importance: medium
Dynamic update: per-broker
刷新尝试之间的登录线程睡眠时间。

sasl.kerberos.principal.to.local.rules

Type: list
Default: DEFAULT
Importance: medium
Dynamic update: per-broker
从主体名称映射到短名称（通常是操作系统用户名）的规则列表。规则按顺序评估，第一个与主体名称匹配的规则用于将其映射到短名称。稍后列表中的任何规则都会被忽略。默认情况下，形式 **{username}/{hostname}@{REALM}** 的主体名称映射到 **{username}**。有关格式的详情，请查看 安全

授权和 acls。请注意，如果 **principal.builder.class** 配置提供了 **KafkaPrincipalBuilder** 的扩展，则此配置会被忽略。

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。

sasl.kerberos.ticket.renew.jitter

type: double
Default: 0.05
Importance: medium
Dynamic update: per-broker
添加到续订时间的随机 jitter 的百分比。

sasl.kerberos.ticket.renew.window.factor

type: double
Default: 0.8
Importance: medium
Dynamic update: per-broker
登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
实现 AuthenticateCallbackHandler 接口的 SASL 登录回调处理程序类的完全限定名称。对于代理，登录回调处理器配置必须以监听器前缀和 SASL 机制名称作为前缀作为前缀。例如：
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
实现登录接口的类的完全限定名称。对于代理，登录配置必须在小写中带有监听前缀和 SASL 机制名称前缀。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
在刷新凭证时凭证过期前的缓冲时间（以秒为单位）。如果刷新情况比缓冲区秒数接近过期，则刷新将移动，以尽可能多地维护缓冲区时间。法律值介于 0 到 3600 (1 小时)之间；如果没有指定值，则使用默认值 300 (5 分钟)。如果这个值和 sasl.login.refresh.min.period.seconds 超过了凭证剩余的生命周

期，则忽略这个值。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Importance: medium
Dynamic update: per-broker
登录刷新线程在刷新凭证前等待的时间（以秒为单位）。法律值介于 0 到 900 (15 分钟)之间；如果没有指定值，则使用默认值 60 (1 分钟)。如果这个值和 sasl.login.refresh.buffer.seconds 都会被忽略，如果它们的总和超过凭证的剩余生命周期。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.factor

type: double
Default: 0.8
Importance: medium
Dynamic update: per-broker
登录刷新线程将休眠到与凭证生命周期相关的指定窗口因素，此时将尝试刷新凭证。法律值介于 0.5 (50%)和 1.0 (100%)之间，如果没有指定值，则使用默认值 0.8 (80%)。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.jitter

type: double
Default: 0.05
Importance: medium
Dynamic update: per-broker
与凭证的生命周期相关的最大随机 jitter 量添加到登录刷新线程的睡眠时间中。法律值介于 0 到 0.25(25%)之间，如果没有指定值，则使用默认值 0.05(5%)。目前仅适用于 OAUTHBEARER。

sasl.mechanism.inter.broker.protocol

Type: string
Default: GSSAPI
Importance: medium
Dynamic update: per-broker
用于代理间通信的 SASL 机制。默认为 GSSAPI。

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
Dynamic update: read-only
可以从中检索供应商的 JWKS (JSON Web Key Set) 的 OAuth/OIDC 供应商 URL。URL 可以是基于 HTTP (S)或基于文件的 URL。如果 URL 基于 HTTP (S)，则 JWKS 数据将通过代理启动时配置的 URL 从 OAuth/OIDC 供应商中检索。所有 then-current 密钥都将缓存在代理上以进行传入请求。如果为 JWT 收到了身份验证请求，其中包含缓存中尚未在缓存中的"kid"标头声明值，则将根据需要再次查询 JWKS 端点。但是，代理会轮询每个 sasl.oauthbearer.jwks.endpoint.refresh.ms 毫秒的 URL，以便在收到包含这些密钥的任何 JWT 请求前刷新缓存。如果 URL 基于文件，代理将在启动时从配置的位置加载 JWKS 文件。如果 JWT 包含没有在 JWKS 文件中的 "kid" 标头值，代理将拒绝 JWT 并且身份验证将失败。

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null

Importance: medium
Dynamic update: read-only
OAuth/OIDC 身份提供程序的 URL。如果 URL 基于 HTTP (S)，这是签发者的令牌端点 URL，其请求将根据 sasl.jaas.config 中的配置登录。如果 URL 基于文件，它指定一个包含由 OAuth/OIDC 身份提供程序发布的访问令牌(JWT 序列化形式)的文件，以用于授权。

sasl.server.callback.handler.class

Type: class
Default: null
Importance: medium
Dynamic update: read-only
实现 AuthenticateCallbackHandler 接口的 SASL 服务器回调处理程序类的完全限定名称。服务器回调处理程序必须在小写中带有监听器前缀和 SASL 机制名称作为前缀。例如：
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
在初始 SASL 身份验证前和期间允许的最大接收大小。默认接收大小为 512KB。GSSAPI 将请求限制为 64K，但我们默认为自定义 SASL 机制允许 512KB。实际上，PLAIN、SCRAM 和 OAUTH 机制可以使用更小的限制。

security.inter.broker.protocol

Type: string
Default: PLAINTEXT
Valid Values: [PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL]
Importance: medium
Dynamic update: read-only
用于在代理间进行通信的安全协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。这是一个同时设置这个和 inter.broker.listener.name 属性的错误。

socket.connection.setup.timeout.max.ms

type: long
Default: 30000 (30 秒)
Importance: medium
Dynamic update: read-only
客户端等待套接字连接建立的最大时间。当每个连续连接失败时，连接设置超时会达到这个最大值。为避免连接差异，将把一个随机化因值应用于超时，导致计算值高于 20% 的随机范围。

socket.connection.setup.timeout.ms

type: long
Default: 10000 (10 秒)
Importance: medium
Dynamic update: read-only
客户端等待套接字连接建立的时间长度。如果在超时时间前没有构建连接，客户端将关闭套接字频道。这个值是初始 backoff 值，每个连续连接失败都会按指数增加，最高为 **socket.connection.setup.timeout.max.ms** 值。

socket.listen.backlog.size

type: int

Default: 50
Valid Values: [1,...]
Importance: medium
Dynamic update: read-only
套接字上待处理连接的最大数量。在 Linux 中，您可能还需要相应地配置 **somaxconn** 和 **tcp_max_syn_backlog** 内核参数，以使配置生效。

ssl.cipher.suites

type: list
Default: ""
Importance: medium
Dynamic update: per-broker
密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。默认情况下，支持所有可用的密码套件。

ssl.client.auth

Type: string
Default: none
Valid Values: [required, requested, none]
Importance: medium
Dynamic update: per-broker
配置 kafka 代理以请求客户端身份验证。以下设置是通用的：

- 如果设为所需的客户端身份验证，则需要 **ssl.client.auth=required**。
- **ssl.client.auth=requested** 意味着客户端身份验证是可选的。与需要不同，如果设置此选项，则可以选择不提供有关其自身的身份验证信息
- **SSL.client.auth=none** 意味着不需要客户端身份验证。

ssl.enabled.protocols

type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
Dynamic update: per-broker
为 SSL 连接启用的协议列表。在使用 Java 11 或更新版本时，默认为 'TLSv1.2,TLSv1.3'，否则 'TLSv1.2'。使用 Java 11 的默认值，如果客户端和服务器同时支持 TLSv1.3，则客户端和服务器将首选 TLSv1.3，否则将回退到 TLSv1.2（假设两者都至少支持 TLSv1.2）。对于大多数情况，这个默认值应该可以正常工作。另请参阅 **ssl.protocol** 的配置文档。

ssl.key.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
密钥存储文件中私钥的密码或 'ssl.keystore.key' 中指定的 PEM 密钥。

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: medium
Dynamic update: per-broker
密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
使用由 'ssl.keystore.type" 指定的格式的证书链。默认 SSL 引擎工厂仅支持使用 X.509 证书列表的 PEM 格式。

ssl.keystore.key

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
使用 'ssl.keystore.type" 指定的格式的私钥。默认 SSL 引擎工厂只支持使用 PKCS#8 密钥的 PEM 格式。如果密钥加密，必须使用 'ssl.key.password' 指定密钥密码。

ssl.keystore.location

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。

ssl.keystore.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 'ssl.keystore.location' 时才需要。PEM 格式不支持密钥存储密码。

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
Dynamic update: per-broker
密钥存储文件的文件格式。这对客户端是可选的。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

ssl.protocol

type: string
Default: TLSv1.3
Importance: medium
Dynamic update: per-broker
用于生成 SSLContext 的 SSL 协议。使用 Java 11 或更新版本运行时，默认为 'TLSv1.3'，否则 'TLSv1.2'。对于大多数用例，这个值应该可以正常工作。最近的 JVM 中允许的值为 'TLSv1.2' 和 'TLSv1.3'。旧的 JVM 可以支持 'TLS', 'TLSv1.1', 'SSLv2', 'SSLv2' 和 'SSLv3'，但由于已知的安全漏洞，不建议使用它们。使用这个配置和 'ssl.enabled.protocols' 的默认值，如果服务器不支持 'TLSv1.3'，客户端将降级为 'TLSv1.2'。如果此配置被设置为 'TLSv1.2'，客户端将不会使用 'TLSv1.3'，即使它是 ssl.enabled.protocols 中的值之一，服务器只支持 'TLSv1.3'。

ssl.provider

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: medium
Dynamic update: per-broker
信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。

ssl.truststore.certificates

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
可信证书，格式为 'ssl.truststore.type'。默认 SSL 引擎工厂仅支持使用 X.509 证书使用 PEM 格式。

ssl.truststore.location

Type: string
Default: null
Importance: medium
Dynamic update: per-broker
信任存储文件的位置。

ssl.truststore.password

Type: password
Default: null
Importance: medium
Dynamic update: per-broker
信任存储文件的密码。如果没有设置密码，则仍然使用配置的信任存储文件，但禁用完整性检查。PEM 格式不支持信任存储密码。

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
Dynamic update: per-broker
信任存储文件的文件格式。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

zookeeper.clientCnxnSocket

Type: string
Default: null
Importance: medium
Dynamic update: read-only
通常，在使用 TLS 连接到 ZooKeeper 时，将 **org.apache.zookeeper.ClientCnxnSocketNetty** 设置为 org.apache.zookeeper.ClientCnxnSocketNetty。覆盖通过同名 **zookeeper.clientCnxnSocket** 系统属性设置的任何显式值。

zookeeper.ssl.client.enable

Type: boolean
Default: false
Importance: medium
Dynamic update: read-only
将 client 设置为在连接到 ZooKeeper 时使用 TLS。显式值会覆盖通过 **zookeeper.client.secure** 系统属性设置的任何值（请注意不同的名称）。如果没有设置，则默认为 false；如果未设置 **zookeeper.clientCnxnSocket** nSocket （通常为 **org.apache.zookeeper.ClientCnxnSocketNetty**）；其他值可以包括 **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
将客户端证书与 ZooKeeper 的 TLS 连接一起使用时的密钥存储位置。覆盖通过 **zookeeper.ssl.keyStore.location** 系统属性设置的任何显式值（请注意 camelCase）。

zookeeper.ssl.keystore.password

Type: password
Default: null
Importance: medium
Dynamic update: read-only
将客户端证书与 ZooKeeper 的 TLS 连接一起使用时，密钥存储密码。覆盖通过 **zookeeper.ssl.keyStore.password** 系统属性设置的任何显式值（请注意 camelCase）。请注意，Zookeeper 不支持与密钥存储密码不同的密钥密码，因此请确保将密钥存储中的密钥密码设置为与密钥存储密码相同；否则，尝试 Zookeeper 的连接将失败。

zookeeper.ssl.keystore.type

Type: string
Default: null
Importance: medium
Dynamic update: read-only
将客户端证书与 ZooKeeper 的 TLS 连接一起使用时，密钥存储类型。覆盖通过 **zookeeper.ssl.keyStore.type** 系统属性设置的任何显式值（请注意 camelCase）。默认值 **null** 表示类型将根据密钥存储的文件名扩展名自动探测到。

zookeeper.ssl.truststore.location

Type: string
Default: null
Importance: medium
Dynamic update: read-only
使用 TLS 连接到 ZooKeeper 时的 truststore 位置。覆盖通过 **zookeeper.ssl.trustStore.location** 系统属性设置的任何显式值（请注意 camelCase）。

zookeeper.ssl.truststore.password

Type: password

Default: null
Importance: medium
Dynamic update: read-only
使用 TLS 连接到 ZooKeeper 时的 truststore 密码。覆盖通过 **zookeeper.ssl.trustStore.password** 系统属性设置的任何显式值（请注意 camelCase）。

zookeeper.ssl.truststore.type

Type: string
Default: null
Importance: medium
Dynamic update: read-only
使用 TLS 连接 ZooKeeper 时的 truststore 类型。覆盖通过 **zookeeper.ssl.trustStore.type** 系统属性设置的任何显式值（请注意 camelCase）。默认值 **null** 表示类型将根据信任存储的文件名扩展名自动探测到。

alter.config.policy.class.name

Type: class
Default: null
Importance: low
Dynamic update: read-only
应该用于验证的更改配置策略类。该类应实施 **org.apache.kafka.server.policy.AlterConfigPolicy** 接口。

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

type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
要在内存中保留的示例数量，用于更改日志目录复制配额。

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

type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
每个修改日志目录复制配额的时间范围。

authorizer.class.name

type: string
Default: ""
Valid Values: non-null string
Importance: low
Dynamic update: read-only
实现 **org.apache.kafka.server.authorizer.Authorizer** 接口的类的完全限定名称，代理将用于授权。

auto.include.jmx.reporter

type: boolean
Default: true
Importance: low
Dynamic update: read-only

已弃用。是否自动包含 JmxReporter，即使它在 **metric.reporters** 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 **metric.reporters** 中包含 **org.apache.kafka.common.metrics.JmxReporter**，以启用 JmxReporter。

client.quota.callback.class

Type: class
Default: null
Importance: low
Dynamic update: read-only
实现 ClientQuotaCallback 接口的类的完全限定名称，用于决定应用到客户端请求的配额限制。默认情况下，应用存储在 ZooKeeper 中的 <user> 和 <client-id> 配额。对于任何给定请求，会应用与会话的用户主体和请求的 client-id 匹配的最具体配额。

connection.failed.authentication.delay.ms

type: int
Default: 100
Valid Values: [0,...]
Importance: low
Dynamic update: read-only
在失败的身份验证时连接关闭延迟：这是在身份验证失败时连接关闭的时间（以毫秒为单位）。这必须配置为小于 connection.max.idle.ms，以防止连接超时。

controller.quorum.retry.backoff.ms

type: int
Default: 20
Importance: low
Dynamic update: read-only
尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 **retry.backoff.max.ms** 值。

controller.quota.window.num

type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
要在内存中为控制器修改配额保留的示例数量。

controller.quota.window.size.seconds

type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
每个控制器修改配额的示例的时间跨度。

create.topic.policy.class.name

Type: class
Default: null
Importance: low
Dynamic update: read-only

应该用于验证的 create 主题策略类。该类应实施 **org.apache.kafka.server.policy.CreateTopicPolicy** 接口。

delegation.token.expiry.check.interval.ms

type: long
Default: 3600000 (1 hour)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
扫描间隔，以删除已过期的委托令牌。

kafka.metrics.polling.interval.secs

type: int
Default: 10
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
指标轮询间隔（以秒为单位），可在 kafka.metrics.reporters 实现中使用。

kafka.metrics.reporters

Type: list
Default: ""
Importance: low
Dynamic update: read-only
用作 Yammer 指标定制报告器的类列表。报告者应该实现 **kafka.metrics.KafkaMetricsReporter** trait。如果客户端想要在自定义报告器上公开 JMX 操作，自定义报告器还需要实施一个 MBean 特征来扩展 **kafka.metrics.KafkaMetricsReporterMBean** 特征，以便注册的 MBean 符合标准的 MBean 约定。

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
在侦听器名称和安全协议之间进行映射。这必须针对同一安全协议定义，才能在多个端口或 IP 中使用。例如，即使两者都需要 SSL，也可以分隔内部和外部流量。另外，用户可以将名为 INTERNAL 和 EXTERNAL 的监听程序定义为：**INTERNAL:SSL,EXTERNAL:SSL**。如上所示，键和值使用冒号分隔，映射条目则用逗号分开。每个侦听器名称应当仅显示映射中的一次。可以通过向配置名称中添加规范化前缀（监听程序名称小写）来为每个监听程序配置不同的安全(SSL 和 SASL)设置。例如，要为 INTERNAL 侦听器设置不同的密钥存储，将设置名为 **listener.name.internal.ssl.keystore.location** 的配置。如果没有设置监听程序名称的配置，则配置将回退到通用配置（如 **ssl.keystore.location**）。请注意，在 KRaft 中，如果不提供显式映射且没有使用其他安全协议，则假定从 **controller.listener.names** 到 PLAINTEXT 定义的监听程序名称的默认映射。

log.message.downconversion.enable

Type: boolean
Default: true
Importance: low
Dynamic update: cluster-wide
此配置控制消息格式的下行是否启用以满足使用请求的使用。当设置为 **false** 时，代理不会为希望旧的消息格式的消费者执行 down-conversion。对于来自较旧的客户端的消费请求，代理会以 **UNSUPPORTED_VERSION** 错误响应。此配置不适用于复制可能需要的任何消息格式转换。

metadata.max.idle.interval.ms

type: int
Default: 500
Valid Values: [0,...]
Importance: low
Dynamic update: read-only
此配置控制活动控制器应将 no-op 记录写入元数据分区的频率。如果值为 0，则不会将 no-op 记录附加到元数据分区中。默认值为 500。

metric.reporters

Type: list
Default: ""
Importance: low
Dynamic update: cluster-wide
用作指标报告器的类列表。实施 **org.apache.kafka.common.metrics.MetricsReporter** 接口，允许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Importance: low
Dynamic update: read-only
指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
计算指标示例的时间窗口。

password.encoder.cipher.algorithm

type: string
Default: AES/CBC/PKCS5Padding
Importance: low
Dynamic update: read-only
用于动态配置的编码密码的 Cipher 算法。

password.encoder.iterations

type: int
Default: 4096
Valid Values: [1024,...]

Importance: low
Dynamic update: read-only
用于动态配置的编码密码的迭代计数。

password.encoder.key.length

type: int
Default: 128
Valid Values: [8,...]
Importance: low
Dynamic update: read-only
用于动态配置的编码密码的密钥长度。

password.encoder.keyfactory.algorithm

Type: string
Default: null
Importance: low
Dynamic update: read-only
用于动态配置的编码密码的 SecretKeyFactory 算法。如果可用，则默认为 PBKDF2WithHmacSHA512，否则为 PBKDF2WithHmacSHA1。

producer.id.expiration.ms

type: int
Default: 86400000 (1 day)
Valid Values: [1,...]
Importance: low
Dynamic update: cluster-wide
主题分区领导机在过期制作者 ID 前等待的时间（毫秒）。当与它们关联的事务仍然持续时，生成者 ID 不会过期。请注意，如果因为主题的保留设置而删除了来自制作者 ID 的最后写入，则生成者 ID 可能会更早过期。设置这个值的高于 **delivery.timeout.ms** 可以帮助防止重试期间过期并防止消息重复，但默认值应该适合于大多数用例。

quota.window.num

type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
要在内存中为客户端配额保留的示例数量。

quota.window.size.seconds

type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
客户端配额的每个示例的时间跨度。

remote.log.index.file.cache.total.size.bytes

Type: long
Default: 1073741824 (1 gibibyte)
Valid Values: [1,...]

Importance: low
Dynamic update: cluster-wide
分配给存储从本地存储中获取的索引文件的空间总量。

remote.log.manager.task.interval.ms

type: long
Default: 30000 (30 秒)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
远程日志管理器运行调度任务（如复制片段）的间隔，并清理远程日志片段。

remote.log.metadata.custom.metadata.max.bytes

type: int
Default: 128
Valid Values: [0,...]
Importance: low
Dynamic update: read-only
代理应从远程存储插件接受的最大自定义元数据大小，以字节为单位。如果自定义元数据超过这个限制，则不会存储更新的段元数据，复制的数据将尝试删除，并且此 topic-partition 的远程复制任务将停止并显示错误。

replication.quota.window.num

type: int
Default: 11
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
要在内存中为复制配额保留的示例数量。

replication.quota.window.size.seconds

type: int
Default: 1
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
复制配额的每个示例的时间跨度。

sasl.login.connect.timeout.ms

type: int
Default: null
Importance: low
Dynamic update: read-only
（可选）外部身份验证供应商连接超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.read.timeout.ms

type: int
Default: null
Importance: low
Dynamic update: read-only
（可选）外部身份验证供应商读取超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.max.ms

type: long
Default: 10000 (10 秒)
Importance: low
Dynamic update: read-only
（可选）登录尝试外部身份验证提供程序之间等待的最大等待值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
Dynamic update: read-only
（可选）登录尝试外部身份验证提供程序期间初始等待的值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.oauthbearer.clock.skew.seconds

type: int
Default: 30
Importance: low
Dynamic update: read-only
（可选）值（以秒为单位），允许 OAuth/OIDC 身份提供程序和代理间的差别。

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
Dynamic update: read-only
（可选）代理用逗号分隔的设置来验证是否已为其中一个预期的受众发出 JWT。JWT 将检查是否有标准的 OAuth "aud" 声明，如果设置了这个值，代理将与 JWT 的"aud"声明中的值匹配，以查看是否有完全匹配。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
Dynamic update: read-only
代理的 （可选）用于验证 JWT 是否已由预期签发者创建的（可选）设置。JWT 将检查是否有标准 OAuth "iss" 声明，如果设置了这个值，代理会完全匹配 JWT 的"iss"声明中的内容。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
Dynamic update: read-only
（可选）代理在刷新其 JWKS (JSON Web Key Set)缓存之间等待的值，其中包含键以验证 JWT 的签名。

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

type: long
Default: 10000 (10 秒)
Importance: low
Dynamic update: read-only
（可选）尝试从外部身份验证提供程序检索 JWKS (JSON Web Key Set)之间的最大等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

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

Type: long
Default: 100
Importance: low
Dynamic update: read-only
（可选）在 JWKS (JSON Web Key Set)检索外部身份验证提供程序尝试时的初始等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
Dynamic update: read-only
范围的 OAuth 声明通常命名为 "scope"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以提供用于 JWT 有效负载声明中包含的范围的名称。

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
Dynamic update: read-only
该主题的 OAuth 声明通常命名为 "sub"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以为 JWT 有效负载的声明中包含的主题提供不同的名称。

security.providers

Type: string
Default: null
Importance: low
Dynamic update: read-only
每个返回供应商实施安全算法的可配置创建者类列表。这些类应实施 **org.apache.kafka.common.security.auth.SecurityProviderCreator** 接口。

ssl.allow.dn.changes

Type: boolean
Default: false
Importance: low
Dynamic update: read-only
指明在动态重新配置证书期间是否应该允许更改证书的可分辨名称。

ssl.allow.san.changes

Type: boolean

Default: false
Importance: low
Dynamic update: read-only
指明在动态重新配置证书期间是否应该允许更改证书主题备用名称。

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
Dynamic update: per-broker
使用服务器证书验证服务器主机名的端点标识算法。

ssl.engine.factory.class

type: class
Default: null
Importance: low
Dynamic update: per-broker
org.apache.kafka.common.security.auth.SslEngineFactory 类型的类提供 SSLEngine 对象。默认值为 org.apache.kafka.common.security.ssl.DefaultSslEngineFactory。或者，将其设置为 org.apache.kafka.common.security.ssl.CommonNameLoggingSslEngineFactory 将记录客户端用于与日志级别 INFO 的任何代理进行身份验证的通用 SSL 证书名称。请注意，由于检查客户端提供的证书链，这会在建立从 mTLS 客户端到代理的新连接时造成小延迟。请注意，该实施还使用基于标准 Java 信任存储的自定义信任存储，因此由于标准版本不成熟，因此可能会被视为安全风险。

ssl.principal.mapping.rules

Type: string
Default: DEFAULT
Importance: low
Dynamic update: read-only
用于映射的规则列表，用于将名称与客户端证书区分到短名称。规则按顺序评估，第一个与主体名称匹配的规则用于将其映射到短名称。稍后列表中的任何规则都会被忽略。默认情况下，可区分 X.500 证书的名称将是主体。有关格式的详情，请查看 安全授权和 acls。请注意，如果 **principal.builder.class** 配置提供了 KafkaPrincipalBuilder 的扩展，则此配置会被忽略。

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
Dynamic update: per-broker
用于 SSL 加密操作的 SecureRandom PRNG 实施。

telemetry.max.bytes

type: int
Default: 1048576 (1 mebibyte)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
从客户端推送到代理的遥测指标的最大大小（如果使用压缩后）。默认值为 1048576 (1 MB)。

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

type: int
Default: 10000 (10 秒)

Valid Values: [1,...]
Importance: low
Dynamic update: read-only
回滚超时事务的时间间隔。

transaction.partition.verification.enable

Type: boolean
Default: true
Importance: low
Dynamic update: cluster-wide
启用验证，检查分区是否已添加到事务中，然后再将事务记录写入分区。

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

type: int
Default: 3600000 (1 hour)
Valid Values: [1,...]
Importance: low
Dynamic update: read-only
删除因为 **transactional.id.expiration.ms** 传递而过期的事务的时间间隔。

zookeeper.ssl.cipher.suites

Type: list
Default: null
Importance: low
Dynamic update: read-only
指定在 ZooKeeper TLS 协商(csv)中使用的启用的密码套件。覆盖通过 **zookeeper.ssl.ciphersuites** 系统属性明确设置的值（注意单个单词 "ciphersuites"）。默认值 **null** 表示启用的密码套件列表由所使用的 Java 运行时决定。

zookeeper.ssl.crl.enable

Type: boolean
Default: false
Importance: low
Dynamic update: read-only
指定是否在 ZooKeeper TLS 协议中启用证书撤销列表。覆盖通过 **zookeeper.ssl.crl** 系统属性设置的任何显式值（请注意较短的名称）。

zookeeper.ssl.enabled.protocols

Type: list
Default: null
Importance: low
Dynamic update: read-only
指定 ZooKeeper TLS negotiate (csv)中启用的协议。覆盖通过 **zookeeper.ssl.enabledProtocols** 系统属性设置的任何显式值（请注意 camelCase）。默认值 **null** 表示启用的协议将是 **zookeeper.ssl.protocol** 配置属性值。

zookeeper.ssl.endpoint.identification.algorithm

Type: string
Default: HTTPS
Importance: low
Dynamic update: read-only
指定是否在 ZooKeeper TLS 协商过程中启用主机名验证，使用 （不区分大小写） "https" 表示启用

ZooKeeper 主机名验证，并有一个显式空白值表示它被禁用（仅用于测试目的）。显式值会覆盖通过 **zookeeper.ssl.hostnameVerification** 系统属性设置的任何 "true" 或 "false" 值（请注意不同的名称和值； true 表示 https 和 false 表示空白）。

zookeeper.ssl.ocsp.enable

Type: boolean
Default: false
Importance: low
Dynamic update: read-only
指定是否在 ZooKeeper TLS 协议中启用在线证书状态协议。覆盖通过 **zookeeper.ssl.ocsp** 系统属性设置的任何显式值（请注意较短的名称）。

zookeeper.ssl.protocol

type: string
Default: TLSv1.2
Importance: low
Dynamic update: read-only
指定 ZooKeeper TLS 协商中使用的协议。显式值会覆盖通过同名的 **zookeeper.ssl.protocol** 系统属性设置的任何值。

第 2 章 主题配置属性

cleanup.policy

type: list
Default: delete
Valid Values: [compact, delete]
Server Default Property: log.cleanup.policy
Importance: medium
此配置指定要在日志片段中使用的保留策略。当达到保留时间或大小限制时，"删除"策略（默认）将丢弃旧的片段。"compact"策略将 启用日志压缩，它会保留每个键的最新值。也可以在逗号分隔列表中指定这两个策略（如 "delete,compact"）。在这种情况下，旧片段会根据保留时间和大小配置丢弃，而保留片段将被压缩。

compression.type

Type: string
Default: producer
Valid Values: [uncompressed, zstd, lz4, snappy, gzip, producer]
Server Default Property: compression.type
Importance: medium
指定给定主题的最终压缩类型。此配置接受标准压缩解码器('gzip', 'snappy', 'lz4', 'zstd')。它还接受等同于没有压缩的 'uncompressed'；而 'producer' 表示保留由 producer 设置的原始压缩 codec。

delete.retention.ms

Type: long
Default: 86400000 (1 day)
Valid Values: [0,...]
Server Default Property: log.cleaner.delete.retention.ms
Importance: medium
为日志压缩主题保留 delete tombstone 标记的时间长度。此设置也会在消费者从偏移 0 开始读取的时间绑定，以确保它们获得最终阶段的有效快照（否则，可以在完成扫描前收集该 tombstones）。

file.delete.delay.ms

type: long
Default: 60000 (1 minute)
Valid Values: [0,...]
Server Default Property: log.segment.delete.delay.ms
Importance: medium
从文件系统中删除文件前等待的时间。

flush.messages

type: long
Default: 9223372036854775807
Valid Values: [1,...]
Server Default Property: log.flush.interval.messages
Importance: medium
此设置允许指定将强制写入日志的 fsync 数据的间隔。例如，如果将其设置为 1，则每个消息都会执行 fsync；如果这是 5 次，则每五个消息都会执行 fsync。通常，我们建议您不要这样做，并使用复制来实现持久性，并允许操作系统的后台清除功能，因为它效率更高。这个设置可根据每个主题覆盖（请参阅 每个主题配置部分）。

flush.ms

type: long

Default: 9223372036854775807
Valid Values: [0,...]
Server Default Property: log.flush.interval.ms
Importance: medium
此设置允许指定将强制写入日志的 fsync 数据的时间间隔。例如，如果将其设置为 1000，则在 1000 毫秒被传递后将 fsync。通常，我们建议您不要这样做，并使用复制来实现持久性，并允许操作系统的后台清除功能，因为它效率更高。

follower.replication.throttled.replicas

type: list
Default: ""
Valid Values: [partitionId]:[brokerId],[partitionId]:[brokerId],...
Server Default Property: null
Importance: medium
应该在后续端节流日志复制的副本列表。该列表应该以 [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... 或者通配符 '*' 来节流这个主题的所有副本。

index.interval.bytes

type: int
Default: 4096 (4 kibibytes)
Valid Values: [0,...]
Server Default Property: log.index.interval.bytes
Importance: medium
此设置控制 Kafka 在偏移索引中添加索引条目的频率。默认设置可确保我们索引一个大约每 4096 字节的消息。通过更多索引，读取可以更接近日志中的确切位置，但使索引变大。您可能不需要更改此设置。

leader.replication.throttled.replicas

type: list
Default: ""
Valid Values: [partitionId]:[brokerId],[partitionId]:[brokerId],...
Server Default Property: null
Importance: medium
日志复制应节流到领导端的副本列表。该列表应该以 [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... 或者通配符 '*' 来节流这个主题的所有副本。

local.retention.bytes

type: long
Default: -2
Valid Values: [-2,...]
Server Default Property: log.local.retention.bytes
Importance: medium
在删除旧片段前，可以为分区增大的本地日志片段的最大大小。默认值为 -2，它代表要使用的 **retention.bytes** 值。有效的值应始终小于或等于 **retention.bytes** 值。

local.retention.ms

type: long
Default: -2
Valid Values: [-2,...]
Server Default Property: log.local.retention.ms
Importance: medium

在删除本地日志段前保持本地日志段的毫秒数。默认值为 -2，它代表使用 **retention.ms** 值。有效的值应始终小于或等于 **retention.ms** 值。

max.compaction.lag.ms

type: long
Default: 9223372036854775807
Valid Values: [1,...]
Server Default Property: log.cleaner.max.compaction.lag.ms
Importance: medium
消息在日志中保留无法压缩的最长时间。仅适用于被压缩的日志。

max.message.bytes

type: int
Default: 1048588
Valid Values: [0,...]
Server Default Property: message.max.bytes
Importance: medium
Kafka 允许的最大记录批处理大小（如果启用了压缩，在压缩后进行压缩）。如果这被增加，且有 0.10.2 旧的消费者，还必须增加用户的获取大小，以便他们可以获取记录批处理。在最新的消息格式版本中，记录始终分组到批处理中，以获得效率。在以前的消息格式版本中，未压缩记录没有分组到批处理中，在这种情况下，这个限制只适用于单个记录。

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, 1.1-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, 3.0-iv0, 3.0-iv1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.3-IV0, 3.5-IV0, 3.5-IV1, 3.5-IV2, 3.6-IV0, 3.6-IV1, 3.6-IV2, 3.7-IV0, 3.7-IV1, 3.7-IV2, 3.7-IV3, 3.7-IV4, 3.8-IV0, 3.8-IV0]
Server Default Property: log.message.format.version
Importance:
[DEPRECATED] 指定代理用来将消息附加到日志中的消息格式版本。如果 **inter.broker.protocol.version** 为 **3.0** 或更高版本（实际配置值将被忽略），则始终假定此配置的值为 3.0。否则，该值应该是有效的 ApiVersion。一些示例包括： 0.10.0、1.1、2.8、3.0。通过设置特定的消息格式版本，用户认证磁盘上所有现有的消息是否都小于或等于指定版本。错误地设置这个值将导致具有旧版本的用户中断，因为它们会收到一个不理解的格式的消息。

message.timestamp.after.max.ms

type: long
Default: 9223372036854775807
Valid Values: [0,...]
Server Default Property: log.message.timestamp.after.max.ms
Importance: medium
此配置设定了消息时间戳和代理时间戳之间的允许时间戳差异。消息时间戳可能高于代理的时间戳，最大允许差别由此配置中设置的值决定。如果 message.timestamp.type=CreateTime，如果时间戳的差异超过这个指定的阈值，则消息将被拒绝。如果 message.timestamp.type=LogAppendTime，则忽略此配置。

message.timestamp.before.max.ms

type: long
Default: 9223372036854775807

Valid Values: [0,...]
Server Default Property: log.message.timestamp.before.max.ms
Importance: medium
此配置设定了代理的时间戳和消息时间戳之间的允许时间戳差异。消息时间戳可以早于或等于代理的时间戳，其最大允许区别由此配置中设置的值决定。如果 message.timestamp.type=CreateTime，如果时间戳的差异超过这个指定的阈值，则消息将被拒绝。如果 message.timestamp.type=LogAppendTime，则忽略此配置。

message.timestamp.difference.max.ms

type: long
Default: 9223372036854775807
Valid Values: [0,...]
Server Default Property: log.message.timestamp.difference.max.ms
Importance: medium
[DEPRECATED] 代理收到消息和消息中指定的时间戳之间允许的最大区别。如果 message.timestamp.type=CreateTime，如果时间戳的差异超过这个阈值，则会拒绝一条消息。如果 message.timestamp.type=LogAppendTime，则忽略此配置。

message.timestamp.type

type : string
Default: CreateTime
Valid Values: [CreateTime, LogAppendTime]
Server Default Property: log.message.timestamp.type
Importance: medium
定义消息中的时间戳是消息创建时间还是日志附加时间。该值应该是 **CreateTime** 或 **LogAppendTime**。

min.cleanable.dirty.ratio

type: double
Default: 0.5
Valid Values: [0,...,1]
Server Default Property: log.cleaner.min.cleanable.ratio
Importance: medium
此配置控制日志压缩器将尝试清理日志的频率（假设启用了 日志压缩 ）。默认情况下，我们将避免清理日志超过 50% 的日志。这种比例使日志空间达到最大的空间（50% 最多 50% 的日志重复）。较高的比率将意味着较小的、效率更高，但意味着日志中有更多空间。如果还指定了 max.compaction.lag.ms 或 min.compaction.lag.ms 配置，则日志紧凑器会认为日志在 min.compaction.lag.ms 期间立即有资格进行压缩：(i)脏比率阈值，并且日志至少有 min.compaction.lag.ms 持续时间的脏（未紧凑）记录， 或(ii)如果日志在 max.compaction.lag.ms 期间内有脏(uncompacted)记录。

min.compaction.lag.ms

type: long
Default: 0
Valid Values: [0,...]
Server Default Property: log.cleaner.min.compaction.lag.ms
Importance: medium
消息在日志中保持 uncompacted 的最短时间。仅适用于被压缩的日志。

min.insync.replicas

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

Server Default Property: min.insync.replicas
Importance: medium
当制作者将 acks 设为 "all" （或 "-1"），此配置指定了必须确认写入的最小副本数才被视为成功。如果无法满足此最小值，则生成者将引发异常(NotEnoughReplicas 或 NotEnoughReplicasAfterAppend)。当一起使用时，**min.insync.replicas** 和 **acks** 允许您强制实施更大的持久性保证。典型的场景是创建包含复制原因 3 的主题，将 **min.insync.replicas** 设置为 2，并使用 **acks** of "all" 生成。这将确保如果大多数副本都未收到写入，则生成者引发异常。

preallocate

type: boolean
Default: false
Server Default Property: log.preallocate
Importance: medium
如果我们在创建新日志段时，应该预先在磁盘上分配该文件则为 true。

remote.storage.enable

Type: boolean
Default: false
Server Default Property: null
Importance: medium
要为主题启用分层存储，请将此配置设置为 true。启用后，您无法禁用此配置。它将在以后的版本中提供。

retention.bytes

type: long
Default: -1
Server Default Property: log.retention.bytes
Importance: medium
此配置控制分区（由日志片段组成的）在我们使用"删除"保留策略时丢弃旧日志片段以释放空间的最大大小。默认情况下，只有时间限制没有大小限制。由于这个限制在分区级别强制实施，因此乘以分区数量来计算主题保留（以字节为单位）。

retention.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [-1,...]
Server Default Property: log.retention.ms
Importance: medium
此配置控制在我们使用"删除"保留策略之前，我们将保留日志段的最长时间，以释放空间。这代表了 SLA，即消费者必须多久才能读取其数据。如果设置为 -1，则不会应用时间限制。

segment.bytes

type: int
Default: 1073741824 (1 gibibyte)
Valid Values: [14,...]
Server Default Property: log.segment.bytes
Importance: medium
此配置控制日志的片段文件大小。保留和清理始终会一次完成一个文件，因此大的片段大小意味着较少的文件，但对保留进行精细的控制。

segment.index.bytes

type: int

Default: 10485760 (10 mebibytes)
Valid Values: [4,...]
Server Default Property: log.index.size.max.bytes
Importance: medium
此配置控制将偏移映射到文件位置的索引大小。我们预先分配此索引文件，并仅在日志滚动后缩小它。您通常不需要更改此设置。

segment.jitter.ms

type: long
Default: 0
Valid Values: [0,...]
Server Default Property: log.roll.jitter.ms
Importance: medium
从调度的网段滚动时间中减去的最大随机 jitter，以避免理解网段滚动的混乱。

segment.ms

Type: long
Default: 604800000 (7 days)
Valid Values: [1,...]
Server Default Property: log.roll.ms
Importance: medium
此配置控制 Kafka 将强制日志回滚的时间周期，即使片段文件没有满，以确保保留可以删除或压缩旧数据。

unclean.leader.election.enable

type: boolean
Default: false
Server Default Property: unclean.leader.election.enable
Importance: medium
指明是否启用 ISR 集中的副本作为最后的手段，即使这样做可能会导致数据丢失。

message.downconversion.enable

type: boolean
Default: true
Server Default Property: log.message.downconversion.enable
Importance: low
此配置控制消息格式的下行是否启用以满足使用请求的使用。当设置为 **false** 时，代理不会为希望旧的消息格式的消费者执行 down-conversion。对于来自较旧的客户端的消费请求，代理会以 **UNSUPPORTED_VERSION** 错误响应。此配置不适用于复制可能需要的任何消息格式转换。

第 3 章 消费者配置属性

key.deserializer

type: class
Importance: high
deserializer 类用于实现 **org.apache.kafka.common.serialization.Deserializer** 接口的密钥。

value.deserializer

type: class
Importance: high
deserializer 类用于实现 **org.apache.kafka.common.serialization.Deserializer** 接口的值。

bootstrap.servers

type: list
Default: ""
Valid Values: non-null string
Importance: high
用于建立到 Kafka 集群的初始连接的主机/端口对列表。客户端将使用所有服务器与此处为引导指定的服务器无关 - 此列表仅影响用于发现完整服务器集的初始主机。此列表的格式应为 **host1:port1,host2:port2,...**。由于这些服务器仅用于初始连接来发现完整的群集成员身份（可能会动态更改），因此此列表不需要包含整组服务器（但是如果服务器停机，您可能需要多个服务器）。

fetch.min.bytes

type: int
Default: 1
Valid Values: [0,...]
Importance: high
服务器应返回的最小数据量，用于获取请求。如果数据不足，请求将等待这么多的数据在回答请求前累积。1 字节的默认设置表示，当数据有很多字节或获取请求超时等待数据到达时，就会对获取请求进行回答。把它设置为较大的值将导致服务器等待大量数据积累，这可以提高服务器吞吐量（以一些额外的延迟为代价）。

group.id

Type: string
Default: null
Importance: high
标识此消费者所属的消费者组的唯一字符串。如果消费者使用 subscription **(topic)** 或基于 Kafka 的偏移管理策略，则需要此属性。

group.protocol

Type: string
Default: classic
Valid Values: (case insensitive)[CONSUMER, CLASSIC]
Importance: high
组协议消费者应使用。我们目前支持"经典"或"消费者"。如果指定了 "consumer"，则使用消费者组协议。否则，将使用经典组协议。

heartbeat.interval.ms

Type: int
Default: 3000 (3 秒)
Importance: high

使用 Kafka 的组管理功能时，与消费者协调器之间预期的时间。心跳用于确保消费者保持活跃状态，并在新用户加入或离开该组时促进重新平衡。该值必须小于 **session.timeout.ms**，但设置的值通常不应超过这个值的 1/3。它可以调整甚至较低，以控制正常重新平衡的预期时间。

max.partition.fetch.bytes

type: int
Default: 1048576 (1 mebibyte)
Valid Values: [0,...]
Importance: high
服务器将返回的每个分区的最大数据量。记录由消费者批量获取。如果获取的第一个非空分区中的第一个记录批处理大于这个限制，则批处理仍会返回，以确保消费者能够进行进度。代理接受的最大记录批处理大小通过 **message.max.bytes** (broker config)或 **max.message.bytes** (topic config)定义。如需限制消费者请求大小，请参阅 fetch.max.bytes。

session.timeout.ms

Type: int
Default: 45000 (45 seconds)
Importance: high
使用 Kafka 的组管理功能时检测客户端故障的超时。客户端发送定期心跳以指示其存活度到代理。如果在这个会话超时前代理没有接收心跳，则代理将从组中删除此客户端并启动重新平衡。请注意，该值必须在允许范围内，如 **group.min.session.timeout.ms** 和 **group.max.session.timeout.ms** 中配置。

ssl.key.password

Type: password
Default: null
Importance: high
密钥存储文件中私钥的密码或 'ssl.keystore.key' 中指定的 PEM 密钥。

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
使用由 'ssl.keystore.type" 指定的格式的证书链。默认 SSL 引擎工厂仅支持使用 X.509 证书列表的 PEM 格式。

ssl.keystore.key

Type: password
Default: null
Importance: high
使用 'ssl.keystore.type" 指定的格式的私钥。默认 SSL 引擎工厂只支持使用 PKCS#8 密钥的 PEM 格式。如果密钥加密，必须使用 'ssl.key.password' 指定密钥密码。

ssl.keystore.location

Type: string
Default: null
Importance: high
密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。

ssl.keystore.password

Type: password
Default: null
Importance: high
密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 'ssl.keystore.location' 时才需要。
PEM 格式不支持密钥存储密码。

ssl.truststore.certificates

Type: password
Default: null
Importance: high
可信证书，格式为 'ssl.truststore.type'。默认 SSL 引擎工厂仅支持使用 X.509 证书使用 PEM 格式。

ssl.truststore.location

Type: string
Default: null
Importance: high
信任存储文件的位置。

ssl.truststore.password

Type: password
Default: null
Importance: high
信任存储文件的密码。如果没有设置密码，则仍然使用配置的信任存储文件，但禁用完整性检查。
PEM 格式不支持信任存储密码。

allow.auto.create.topics

Type: boolean
Default: true
Importance: medium
在订阅或分配主题时，允许对代理自动创建主题。只有在代理允许使用 **auto.create.topics.enable** 代理配置时，才会自动创建订阅的主题。当使用早于 0.11.0 的代理时，此配置必须设置为 **false**。

auto.offset.reset

Type: string
Default: latest
Valid Values: [latest, earliest, none]
Importance: medium
当 Kafka 中没有初始偏移，或者服务器上不存在当前偏移时该怎么办（例如，因为数据已被删除）：

- 最早：自动将偏移重置为最早的偏移
- latest ：自动将偏移重置为最新的偏移
- none：如果没有为消费者的组找到以前的偏移，则向消费者丢弃异常
- 任何其他内容：向消费者抛出异常。
  请注意，在将此配置设置为 latest 时更改分区编号可能会导致消息发送丢失，因为生成者可能开始向新添加的分区发送消息（例如，在消费者重置偏移前，也不存在初始偏移）。

client.dns.lookup

Type: string
Default: use_all_dns_ips

Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
控制客户端如何使用 DNS 查找。如果设置为 **use_all_dns_ips**，请按顺序连接到每个返回的 IP 地址，直到成功建立连接。断开连接后，会使用下一个 IP。当所有 IP 都被一次使用后，客户端会再次解析主机名(JVM 和 OS 缓存 DNS 名称查找)中的 IP。如果设置为 **resolve_canonical_bootstrap_servers_only**，请将每个 bootstrap 地址解析为规范名称列表。bootstrap 阶段后，它的行为与 **use_all_dns_ips** 相同。

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
在这个配置指定的毫秒数后关闭闲置连接。

default.api.timeout.ms

type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: medium
指定客户端 API 的超时时间（以毫秒为单位）。此配置用作没有指定 **timeout** 参数的所有客户端操作的默认超时时间。

enable.auto.commit

Type: boolean
Default: true
Importance: medium
如果为 true，则消费者的偏移将在后台定期提交。

exclude.internal.topics

Type: boolean
Default: true
Importance: medium
订阅中是否应该排除与订阅模式匹配的内部主题。始终可以明确订阅内部主题。

fetch.max.bytes

type: int
Default: 52428800 (50 mebibytes)
Valid Values: [0,...]
Importance: medium
服务器应返回的最大数据量，用于获取请求。记录批处理由消费者批量获取，如果获取的第一个非空分区中的第一个记录批处理大于这个值，则记录批处理仍会返回，以确保消费者进行进度。因此，这不是绝对最大值。代理接受的最大记录批处理大小通过 **message.max.bytes** (broker config)或 **max.message.bytes** (topic config)定义。请注意，消费者并行执行多个获取。

group.instance.id

Type: string
Default: null
Valid Values: non-empty string
Importance: medium
最终用户提供的消费者实例的唯一标识符。只允许非空字符串。如果设置，消费者被视为静态成员，这意味着任何时候都只允许具有此 ID 的一个实例。这可与更大的会话超时结合使用，以避免由临时不可用造成的组重新平衡（如进程重启）。如果没有设置，消费者将加入组作为动态成员，这是传统行

为。

group.remote.assignor

Type: string
Default: null
Importance: medium
要使用的服务器端分配器。如果没有指定分配器，则组协调器将选择一个。只有在 **group.protocol** 设置为 "consumer" 时，才会应用此配置。

isolation.level

Type: string
Default: read_uncommitted
Valid Values: [read_committed, read_uncommitted]
Importance: medium
控制如何读取事务写的消息。如果设置为 **read_committed**，则 consumer.poll （）将仅返回已提交的事务消息。如果设置为 **read_uncommitted** （默认值），则 consumer.poll （）将返回所有消息，即使已中止的事务消息也是如此。非事务性消息将在任一模式中无条件返回。

消息将始终以偏移顺序返回。因此，在 **read_committed** 模式中，consumer.poll （）将仅返回最多最后一个稳定偏移(LSO)的消息，这是第一个打开事务的偏移量。特别是，在属于持续事务的消息后出现的任何消息都会被告知，直到相关事务完成为止。因此，在有机机交易时，**read_committed** 的消费者将无法读取高水位线。

> 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
使用消费者组管理时，poll （）调用之间的最大延迟。这会在获取更多记录前将上限放在消费者可以闲置的时间长度。如果在这个超时时间前没有调用 poll （），则消费者被视为失败，并且组将重新平衡，以便将分区重新分配给另一个成员。对于使用达到这个超时的非null **group.instance.id** 的用户，不会立即重新分配分区。相反，消费者将停止发送心跳，分区将在 **session.timeout.ms** 过期后被重新分配。这会镜像已关闭的静态消费者的行为。

max.poll.records

Type: int
Default: 500
Valid Values: [1,...]
Importance: medium
单个调用返回到 poll （）中返回的最大记录数。请注意，**max.poll.records** 不会影响底层获取行为。消费者将从每个获取请求中缓存记录，并从每个轮询递增返回记录。

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

类名称或类类型列表，按首选顺序排列，在使用组管理时客户端将用于在消费者实例之间分发分区所有权。可用的选项有：

- **org.apache.kafka.clients.consumer.RangeAssignor**: 根据每个主题分配分区。
- **org.apache.kafka.clients.consumer.RoundRobinAssignor**: 以轮循方式将分区分配给消费者。
- **org.apache.kafka.clients.consumer.StickyAssignor**: Guarantees 的一个分配量是最大平衡的，同时保留尽可能多的现有分区分配。
- **org.apache.kafka.clients.consumer.CooperativeStickyAssignor**: Follows 相同的 StickyAssignor 逻辑，但允许合作重新平衡。
  默认分配器为 [RangeAssignor, CooperativeStickyAssignor]，它默认使用 RangeAssignor，但允许升级到 CooperativeStickyAssignor，它只从列表中删除 RangeAssignor。

  通过实施 **org.apache.kafka.clients.consumer.ConsumerPartitionAssignor** 接口，您可以插入自定义分配策略。

receive.buffer.bytes

type: int
Default: 65536 (64 kibibytes)
Valid Values: [-1,...]
Importance: medium
读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。如果值为 -1，则使用操作系统默认值。

request.timeout.ms

type: int
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: medium
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 客户端回调处理程序类的完全限定名称。

sasl.jaas.config

Type: password
Default: null
Importance: medium
用于 SASL 连接的 JAAS 登录上下文参数，其格式供 JAAS 配置文件使用。JAAS 配置文件格式描述在此处。该值的格式是： **loginModuleClass controlFlag (optionName=optionValue)*;**。对于代理，配置必须在小写中带有监听前缀和 SASL 机制名称前缀。例如： listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;。

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium

Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 登录回调处理程序类的完全限定名称。对于代理，登录回调处理器配置必须以监听器前缀和 SASL 机制名称作为前缀作为前缀。例如：
listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler。

sasl.login.class

Type: class
Default: null
Importance: medium
实现登录接口的类的完全限定名称。对于代理，登录配置必须在小写中带有监听前缀和 SASL 机制名称前缀。For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
用于客户端连接的 SASL 机制。这可能是提供安全提供程序的任何机制。GSSAPI 是默认机制。

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
可以从中检索供应商的 JWKS (JSON Web Key Set) 的 OAuth/OIDC 供应商 URL。URL 可以是基于 HTTP (S)或基于文件的 URL。如果 URL 基于 HTTP (S)，则 JWKS 数据将通过代理启动时配置的 URL 从 OAuth/OIDC 供应商中检索。所有 then-current 密钥都将缓存在代理上以进行传入请求。如果为 JWT 收到了身份验证请求，其中包含缓存中尚未在缓存中的"kid"标头声明值，则将根据需要再次查询 JWKS 端点。但是，代理会轮询每个 sasl.oauthbearer.jwks.endpoint.refresh.ms 毫秒的 URL，以便在收到包含这些密钥的任何 JWT 请求前刷新缓存。如果 URL 基于文件，代理将在启动时从配置的位置加载 JWKS 文件。如果 JWT 包含没有在 JWKS 文件中的 "kid" 标头值，代理将拒绝 JWT 并且身份验证将失败。

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
OAuth/OIDC 身份提供程序的 URL。如果 URL 基于 HTTP (S)，这是签发者的令牌端点 URL，其请求将根据 sasl.jaas.config 中的配置登录。如果 URL 基于文件，它指定一个包含由 OAuth/OIDC 身份提供程序发布的访问令牌(JWT 序列化形式)的文件，以用于授权。

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: (case insensitive)[SASL_SSL, PLAINTEXT, SSL, SASL_PLAINTEXT]
Importance: medium
用于与代理通信的协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。

send.buffer.bytes

type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
发送数据时要使用的 TCP 发送缓冲区(SO_SNDBUF)的大小。如果值为 -1，则使用操作系统默认值。

socket.connection.setup.timeout.max.ms

type: long
Default: 30000 (30 秒)
Importance: medium
客户端等待套接字连接建立的最大时间。当每个连续连接失败时，连接设置超时会达到这个最大值。为避免连接差异，将把一个随机化因值应用于超时，导致计算值高于 20% 的随机范围。

socket.connection.setup.timeout.ms

type: long
Default: 10000 (10 seconds)
Importance: medium
客户端等待套接字连接建立的时间长度。如果在超时时间前没有构建连接，客户端将关闭套接字频道。这个值是初始 backoff 值，每个连续连接失败都会按指数增加，最高为 **socket.connection.setup.timeout.max.ms** 值。

ssl.enabled.protocols

type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
为 SSL 连接启用的协议列表。在使用 Java 11 或更新版本时，默认为 'TLSv1.2,TLSv1.3'，否则 'TLSv1.2'。使用 Java 11 的默认值，如果客户端和服务器同时支持 TLSv1.3，则客户端和服务器将首选 TLSv1.3，否则将回退到 TLSv1.2（假设两者都至少支持 TLSv1.2）。对于大多数情况，这个默认值应该可以正常工作。另请参阅 **ssl.protocol** 的配置文档。

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
密钥存储文件的文件格式。这对客户端是可选的。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

ssl.protocol

type: string
Default: TLSv1.3
Importance: medium
用于生成 SSLContext 的 SSL 协议。使用 Java 11 或更新版本运行时，默认为 'TLSv1.3'，否则 'TLSv1.2'。对于大多数用例，这个值应该可以正常工作。最近的 JVM 中允许的值为 'TLSv1.2' 和 'TLSv1.3'。旧的 JVM 可以支持 'TLS', 'TLSv1.1', 'SSLv2', 'SSLv2' 和 'SSLv3'，但由于已知的安全漏洞，不建议使用它们。使用这个配置和 'ssl.enabled.protocols' 的默认值，如果服务器不支持 'TLSv1.3'，客户端将降级为 'TLSv1.2'。如果此配置被设置为 'TLSv1.2'，客户端将不会使用 'TLSv1.3'，即使它是 ssl.enabled.protocols 中的值之一，服务器只支持 'TLSv1.3'。

ssl.provider

Type: string
Default: null
Importance: medium
用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
信任存储文件的文件格式。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

auto.commit.interval.ms

Type: int
Default: 5000 (5 seconds)
Valid Values: [0,...]
Importance: low
如果 **enable.auto.commit** 设置为 **true**，则消费者偏移的频率（以毫秒为单位）。

auto.include.jmx.reporter

Type: boolean
Default: true
Importance: low
已弃用。是否自动包含 JmxReporter，即使它在 **metric.reporters** 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 **metric.reporters** 中包含 **org.apache.kafka.common.metrics.JmxReporter**，以启用 JmxReporter。

check.crcs

Type: boolean
Default: true
Importance: low
自动检查消耗的记录的 CRC32。这样可确保不会发生对消息进行 on-wire 或 on-disk 崩溃。此检查增加了一些开销，因此在寻求极佳性能的情况下可能会禁用它。

client.id

Type: string
Default: ""
Importance: low
在发出请求时传递给服务器的 id 字符串。这样做的目的是通过允许逻辑应用程序名称包含在服务器端请求日志记录中，从而跟踪除 ip/port 以外的请求源。

client.rack

Type: string
Default: ""
Importance: low
此客户端的机架标识符。这可以是任何字符串值，这表示此客户端物理位置。它与代理配置 'broker.rack' 对应。

enable.metrics.push

Type: boolean
Default: true
Importance: low

是否启用将客户端指标推送到集群，如果集群具有与此客户端匹配的客户端指标订阅。

fetch.max.wait.ms

Type: int
Default: 500
Valid Values: [0,...]
Importance: low
如果没有足够的数据立即满足 fetch.min.bytes 提供的要求，服务器将在回答获取请求前阻断的最大时间。

interceptor.classes

type: list
Default: ""
Valid Values: non-null string
Importance: low
用作拦截器的类列表。通过实施 **org.apache.kafka.clients.consumer.ConsumerInterceptor** 接口，您可以截获（并可能修改）消费者收到的记录。默认情况下，没有拦截器。

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。

metric.reporters

type: list
Default: ""
Valid Values: non-null string
Importance: low
用作指标报告器的类列表。实施 **org.apache.kafka.common.metrics.MetricsReporter** 接口，允许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,...]
Importance: low
为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)

Valid Values: [0,...]
Importance: low
计算指标示例的时间窗口。

reconnect.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。

reconnect.backoff.ms

type: long
Default: 50
Valid Values: [0,...]
Importance: low
尝试重新连接到给定主机前等待的基本时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到客户端到代理的所有连接尝试。这个值是初始 backoff 值，并为每个连续连接失败指数增加，最高为 **reconnect.backoff.max.ms** 值。

retry.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
当重试请求重复失败的代理时，等待的最大时间（毫秒）。如果提供，每个客户端将为每个失败请求按指数增加 backoff，直到达到这个最大值。要防止所有客户端在重试时同步，一个随机的 jitter 会被应用到 backoff，从而导致 backoff 在以下 20 到 20% 之间的范围内，超过计算值。如果将 **retry.backoff.ms** 设置为大于 **retry.backoff.max.ms**，则 **retry.backoff.max.ms** 将用作开始的常量 backoff，而不会增加任何指数增加。

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 **retry.backoff.max.ms** 值。

sasl.kerberos.kinit.cmd

type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit 命令路径。

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low

刷新尝试之间的登录线程睡眠时间。

sasl.kerberos.ticket.renew.jitter

type: double
Default: 0.05
Importance: low
添加到续订时间的随机 jitter 的百分比。

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。

sasl.login.connect.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商连接超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.read.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商读取超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
在刷新凭证时凭证过期前的缓冲时间（以秒为单位）。如果刷新情况比缓冲区秒数接近过期，则刷新将移动，以尽可能多地维护缓冲区时间。法律值介于 0 到 3600 (1 小时)之间；如果没有指定值，则使用默认值 300 (5 分钟)。如果这个值和 sasl.login.refresh.min.period.seconds 超过了凭证剩余的生命周期，则忽略这个值。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
登录刷新线程在刷新凭证前等待的时间（以秒为单位）。法律值介于 0 到 900 (15 分钟)之间；如果没有指定值，则使用默认值 60 (1 分钟)。如果这个值和 sasl.login.refresh.buffer.seconds 都会被忽略，如果它们的总和超过凭证的剩余生命周期。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.factor

type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low

登录刷新线程将休眠到与凭证生命周期相关的指定窗口因素，此时将尝试刷新凭证。法律值介于 0.5 (50%)和 1.0 (100%)之间，如果没有指定值，则使用默认值 0.8 (80%)。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.jitter

type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
与凭证的生命周期相关的最大随机 jitter 量添加到登录刷新线程的睡眠时间中。法律值介于 0 到 0.25(25%)之间，如果没有指定值，则使用默认值 0.05(5%)。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.max.ms

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）登录尝试外部身份验证提供程序之间等待的最大等待值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
（可选）登录尝试外部身份验证提供程序期间初始等待的值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
（可选）值（以秒为单位），允许 OAuth/OIDC 身份提供程序和代理间的差别。

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
（可选）代理用逗号分隔的设置来验证是否已为其中一个预期的受众发出 JWT。JWT 将检查是否有标准的 OAuth "aud" 声明，如果设置了这个值，代理将与 JWT 的"aud"声明中的值匹配，以查看是否有完全匹配。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
代理的 （可选）用于验证 JWT 是否已由预期签发者创建的（可选）设置。JWT 将检查是否有标准 OAuth "iss" 声明，如果设置了这个值，代理会完全匹配 JWT 的"iss"声明中的内容。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
（可选）代理在刷新其 JWKS (JSON Web Key Set)缓存之间等待的值，其中包含键以验证 JWT 的签名。

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

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）尝试从外部身份验证提供程序检索 JWKS (JSON Web Key Set)之间的最大等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

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

Type: long
Default: 100
Importance: low
（可选）在 JWKS (JSON Web Key Set)检索外部身份验证提供程序尝试时的初始等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
范围的 OAuth 声明通常命名为 "scope"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以提供用于 JWT 有效负载声明中包含的范围的名称。

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
该主题的 OAuth 声明通常命名为 "sub"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以为 JWT 有效负载的声明中包含的主题提供不同的名称。

security.providers

Type: string
Default: null
Importance: low
每个返回供应商实施安全算法的可配置创建者类列表。这些类应实施 **org.apache.kafka.common.security.auth.SecurityProviderCreator** 接口。

ssl.cipher.suites

Type: list
Default: null
Importance: low
密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。默认情况下，支持所有可用的密码套件。

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
使用服务器证书验证服务器主机名的端点标识算法。

ssl.engine.factory.class

Type: class
Default: null
Importance: low
org.apache.kafka.common.security.auth.SslEngineFactory 类型的类提供 SSLEngine 对象。默认值为 org.apache.kafka.common.security.ssl.DefaultSslEngineFactory。或者，将其设置为 org.apache.kafka.common.security.ssl.CommonNameLoggingSslEngineFactory 将记录客户端用于与日志级别 INFO 的任何代理进行身份验证的通用 SSL 证书名称。请注意，由于检查客户端提供的证书链，这会在建立从 mTLS 客户端到代理的新连接时造成小延迟。请注意，该实施还使用基于标准 Java 信任存储的自定义信任存储，因此由于标准版本不成熟，因此可能会被视为安全风险。

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
用于 SSL 加密操作的 SecureRandom PRNG 实施。

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。

第 4 章 制作者配置属性

key.serializer

type: class
Importance: high
serializer 类用于实现 **org.apache.kafka.common.serialization.Serializer** 接口的密钥。

value.serializer

type: class
Importance: high
Serializer 类用于实现 **org.apache.kafka.common.serialization.Serializer** 接口的值。

bootstrap.servers

type: list
Default: ""
Valid Values: non-null string
Importance: high
用于建立到 Kafka 集群的初始连接的主机/端口对列表。客户端将使用所有服务器与此处为引导指定的服务器无关 - 此列表仅影响用于发现完整服务器集的初始主机。此列表的格式应为 **host1:port1,host2:port2,...**。由于这些服务器仅用于初始连接来发现完整的群集成员身份（可能会动态更改），因此此列表不需要包含整组服务器（但是如果服务器停机，您可能需要多个服务器）。

buffer.memory

type: long
Default: 33554432
Valid Values: [0,...]
Importance: high
制作者可用于缓冲记录等待发送到服务器的内存总量。如果发送记录速度快于可以发送到服务器，则生成者将阻止 **max.block.ms** 引发异常。

此设置应当与生成者将使用的内存总量对应，但不是硬绑定，因为生产者使用的所有内存都用于缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。

compression.type

Type: string
Default: none
Valid Values: [none, gzip, snappy, lz4, zstd]
Importance: high
producer 生成的所有数据的压缩类型。默认为 none （例如，没有压缩）。有效值为 **none**,**gzip**,**snappy**,**lz4**, 或 **zstd**。压缩是整个数据的批处理，因此批处理的效率也会影响压缩率（更多批处理意味着更好的压缩）。

retries

type: int
Default: 2147483647
Valid Values: [0,...,2147483647]
Importance: high
设置大于零的值将导致客户端重新发送发送失败的记录，并显示潜在的临时错误。请注意，这个重试与在收到错误时重新记录不同的是不同的。如果在通过 **delivery.timeout.ms** 配置超时前，在重试次数之前，生成请求将失败，然后再成功确认前过期。用户通常应不设置此配置，而是使用 **delivery.timeout.ms** 来控制重试行为。

启用 idempotence 需要此配置值大于 0。如果设置了冲突配置，且没有显式启用 idempotence，则禁用 idempotence。

在将 **enable.idempotence** 设置为 **false** 和 **max.in.flight.requests.per.connection** 时进行重试可能会更改大于 1 的记录顺序，因为如果两个批处理发送到单个分区，第一次失败，然后重试，但第二个批处理中的记录可能会显示第一个。

ssl.key.password

Type: password
Default: null
Importance: high
密钥存储文件中私钥的密码或 'ssl.keystore.key' 中指定的 PEM 密钥。

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
使用由 'ssl.keystore.type" 指定的格式的证书链。默认 SSL 引擎工厂仅支持使用 X.509 证书列表的 PEM 格式。

ssl.keystore.key

Type: password
Default: null
Importance: high
使用 'ssl.keystore.type" 指定的格式的私钥。默认 SSL 引擎工厂只支持使用 PKCS#8 密钥的 PEM 格式。如果密钥加密，必须使用 'ssl.key.password' 指定密钥密码。

ssl.keystore.location

Type: string
Default: null
Importance: high
密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。

ssl.keystore.password

Type: password
Default: null
Importance: high
密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 'ssl.keystore.location' 时才需要。PEM 格式不支持密钥存储密码。

ssl.truststore.certificates

Type: password
Default: null
Importance: high
可信证书，格式为 'ssl.truststore.type'。默认 SSL 引擎工厂仅支持使用 X.509 证书使用 PEM 格式。

ssl.truststore.location

Type: string
Default: null
Importance: high
信任存储文件的位置。

ssl.truststore.password

Type: password
Default: null
Importance: high
信任存储文件的密码。如果没有设置密码，则仍然使用配置的信任存储文件，但禁用完整性检查。PEM 格式不支持信任存储密码。

batch.size

type: int
Default: 16384
Valid Values: [0,...]
Importance: medium
每当将多个记录发送到同一分区时，生成者将尝试将记录一起批处理到较少的请求。这有助于客户端和服务器的性能。此配置控制默认批处理大小（以字节为单位）。

不会尝试批处理记录大于这个大小。

发送到代理的请求将包含多个批处理，每个分区对应一个数据可以发送。

小批处理大小会减少批处理频率，并可能会降低吞吐量（零的批处理将完全禁用批处理）。非常大的批处理大小可能会更严重地使用内存，因为我们将始终在附加记录下分配指定批处理大小的缓冲。

注意：此设置提供要发送的批处理大小的上限。如果我们为这个分区积累的数量少于此指定的字节，那么我们会"闲置" **linger.ms** 时间来等待更多记录。这个 **linger.ms** 设置默认为 0，这意味着我们将立即发送记录，即使累积的批处理大小位于这个 **batch.size** 设置下。

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
控制客户端如何使用 DNS 查找。如果设置为 **use_all_dns_ips**，请按顺序连接到每个返回的 IP 地址，直到成功建立连接。断开连接后，会使用下一个 IP。当所有 IP 都被一次使用后，客户端会再次解析主机名(JVM 和 OS 缓存 DNS 名称查找)中的 IP。如果设置为 **resolve_canonical_bootstrap_servers_only**，请将每个 bootstrap 地址解析为规范名称列表。bootstrap 阶段后，它的行为与 **use_all_dns_ips** 相同。

client.id

Type: string
Default: ""
Importance: medium
在发出请求时传递给服务器的 id 字符串。这样做的目的是通过允许逻辑应用程序名称包含在服务器端请求日志记录中，从而跟踪除 ip/port 以外的请求源。

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
在这个配置指定的毫秒数后关闭闲置连接。

delivery.timeout.ms

type: int

Default: 120000 (2 minutes)
Valid Values: [0,...]
Importance: medium
在调用 **send （）** 返回后报告成功或失败的上限。这限制了发送前记录总时间、从代理等待确认时间（如果预期），以及可重新发送失败的时间。如果遇到了无法恢复的错误，则生成者可能报告无法发送记录，或者将记录添加到达到早期交付过期期限的批处理中。此配置的值应大于或等于 **request.timeout.ms** 和 **linger.ms** 的总和。

linger.ms

type: long
Default: 0
Valid Values: [0,...]
Importance: medium
生产者将请求传输之间到达单个批处理请求的任何记录组合在一起。通常，这只在记录到达速度快于发送时发生。然而，在某些情况下，客户端可能希望减少请求的数量，即使负载低下也是如此。此设置通过添加少量人工延迟来实现此目的，即不是立即发送记录，而是等待到给定延迟，允许发送其他记录。这可能被视为与 TCP 中的 Nagle 算法类似。此设置提供批处理延迟的上限：一旦我们获得 **批处理。** 无论此设置如何，无论此设置如何，它都会立即发送一次记录，但是如果我们为此分区累积了这个字节，我们将"linger"用于等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 **linger.ms=5** 会影响减少发送的请求数，但会在没有负载的情况下向发送的记录添加最多 5ms 的延迟。

max.block.ms

type: long
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: medium
这个配置控制 **KafkaProducer's `send()**, **partitionsFor()**, **initTransactions()**, **sendOffsetsToTransaction()**, **commitTransaction()** and **abortTransaction()** 方法将被阻塞多长时间。对于 **send （）**，这个超时会绑定了元数据获取和缓冲区分配的总时间（在用户提供的 serializers 或 partitioner 中不会计算这个超时）。对于 **partitionsFor （）**，这个超时会绑定等待元数据的时间（如果其不可用）。与事务相关的方法始终阻止，但如果事务协调器无法发现或没有在超时时间内响应，则可能会超时。

max.request.size

type: int
Default: 1048576
Valid Values: [0,...]
Importance: medium
请求的最大大小（以字节为单位）。此设置将限制制作者将在单个请求中发送的记录批处理数量，以避免发送大量请求。这也实际上是最大未压缩的记录批处理大小的上限。请注意，服务器在记录批处理大小上具有自己的上限（如果启用了压缩，则进行压缩后），这可能与此不同。

partitioner.class

Type: class
Default: null
Importance: medium
决定在生成记录时要向发送记录的分区。可用的选项有：

- 如果没有设置，则使用默认分区逻辑。此策略将记录发送到分区，直到至少对分区生成 batch.size 字节。它可用于策略：

> 1) If no partition is specified but a key is present, choose a partition based on a hash of the key.

> 2) If no partition or key is present, choose the sticky partition that changes when at least batch.size bytes are produced to the partition.
> * `org.apache.kafka.clients.producer.RoundRobinPartitioner`: A partitioning strategy where each record in a series of consecutive records is sent to a different partition, regardless of whether the 'key' is provided or not, until partitions run out and the process starts over again. Note: There's a known issue that will cause uneven distribution when a new batch is created. See KAFKA-9965 for more detail.

通过实施 **org.apache.kafka.clients.producer.Partitioner** 接口，您可以插入自定义分区程序。

partitioner.ignore.keys

Type: boolean
Default: false
Importance: medium
当设置为 'true' 时，生成者不会使用记录密钥来选择分区。如果 'false'，则生成者将在存在密钥时根据密钥的哈希选择一个分区。注意：如果使用自定义分区器，则此设置无效。

receive.buffer.bytes

type: int
Default: 32768 (32 kibibytes)
Valid Values: [-1,...]
Importance: medium
读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。如果值为 -1，则使用操作系统默认值。

request.timeout.ms

type: int
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: medium
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。这应该大于 **replica.lag.time.max.ms** （代理配置），以减少因为不必要的制作者重试而出现消息重复的可能性。

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 客户端回调处理程序类的完全限定名称。

sasl.jaas.config

Type: password
Default: null
Importance: medium
用于 SASL 连接的 JAAS 登录上下文参数，其格式供 JAAS 配置文件使用。JAAS 配置文件格式描述在此处。该值的格式是： **loginModuleClass controlFlag (optionName=optionValue)*;**。对于代理，配置必须在小写中带有监听前缀和 SASL 机制名称前缀。例如： listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;。

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 登录回调处理程序类的完全限定名称。对于代理，登录回调处理器配置必须以监听器前缀和 SASL 机制名称作为前缀作为前缀。例如：
listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler。

sasl.login.class

Type: class
Default: null
Importance: medium
实现登录接口的类的完全限定名称。对于代理，登录配置必须在小写中带有监听前缀和 SASL 机制名称前缀。For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
用于客户端连接的 SASL 机制。这可能是提供安全提供程序的任何机制。GSSAPI 是默认机制。

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
可以从中检索供应商的 JWKS (JSON Web Key Set) 的 OAuth/OIDC 供应商 URL。URL 可以是基于 HTTP (S)或基于文件的 URL。如果 URL 基于 HTTP (S)，则 JWKS 数据将通过代理启动时配置的 URL 从 OAuth/OIDC 供应商中检索。所有 then-current 密钥都将缓存在代理上以进行传入请求。如果为 JWT 收到了身份验证请求，其中包含缓存中尚未在缓存中的"kid"标头声明值，则将根据需要再次查询 JWKS 端点。但是，代理会轮询每个 sasl.oauthbearer.jwks.endpoint.refresh.ms 毫秒的 URL，以便在收到包含这些密钥的任何 JWT 请求前刷新缓存。如果 URL 基于文件，代理将在启动时从配置的位置加载 JWKS 文件。如果 JWT 包含没有在 JWKS 文件中的 "kid" 标头值，代理将拒绝 JWT 并且身份验证将失败。

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
OAuth/OIDC 身份提供程序的 URL。如果 URL 基于 HTTP (S)，这是签发者的令牌端点 URL，其请求将根据 sasl.jaas.config 中的配置登录。如果 URL 基于文件，它指定一个包含由 OAuth/OIDC 身份提供程序发布的访问令牌(JWT 序列化形式)的文件，以用于授权。

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: (case insensitive)[SASL_SSL, PLAINTEXT, SSL, SASL_PLAINTEXT]
Importance: medium
用于与代理通信的协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。

send.buffer.bytes

type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
发送数据时要使用的 TCP 发送缓冲区(SO_SNDBUF)的大小。如果值为 -1，则使用操作系统默认值。

socket.connection.setup.timeout.max.ms

type: long
Default: 30000 (30 秒)
Importance: medium
客户端等待套接字连接建立的最大时间。当每个连续连接失败时，连接设置超时会达到这个最大值。为避免连接差异，将把一个随机化因值应用于超时，导致计算值高于 20% 的随机范围。

socket.connection.setup.timeout.ms

type: long
Default: 10000 (10 seconds)
Importance: medium
客户端等待套接字连接建立的时间长度。如果在超时时间前没有构建连接，客户端将关闭套接字频道。这个值是初始 backoff 值，每个连续连接失败都会按指数增加，最高为 **socket.connection.setup.timeout.max.ms** 值。

ssl.enabled.protocols

type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
为 SSL 连接启用的协议列表。在使用 Java 11 或更新版本时，默认为 'TLSv1.2,TLSv1.3'，否则 'TLSv1.2'。使用 Java 11 的默认值，如果客户端和服务器同时支持 TLSv1.3，则客户端和服务器将首选 TLSv1.3，否则将回退到 TLSv1.2（假设两者都至少支持 TLSv1.2）。对于大多数情况，这个默认值应该可以正常工作。另请参阅 **ssl.protocol** 的配置文档。

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
密钥存储文件的文件格式。这对客户端是可选的。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

ssl.protocol

type: string
Default: TLSv1.3
Importance: medium
用于生成 SSLContext 的 SSL 协议。使用 Java 11 或更新版本运行时，默认为 'TLSv1.3'，否则 'TLSv1.2'。对于大多数用例，这个值应该可以正常工作。最近的 JVM 中允许的值为 'TLSv1.2' 和 'TLSv1.3'。旧的 JVM 可以支持 'TLS', 'TLSv1.1', 'SSLv2', 'SSLv2' 和 'SSLv3'，但由于已知的安全漏洞，

不建议使用它们。使用这个配置和 'ssl.enabled.protocols' 的默认值，如果服务器不支持 'TLSv1.3'，客户端将降级为 'TLSv1.2'。如果此配置被设置为 'TLSv1.2'，客户端将不会使用 'TLSv1.3'，即使它是 ssl.enabled.protocols 中的值之一，服务器只支持 'TLSv1.3'。

ssl.provider

Type: string
Default: null
Importance: medium
用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
信任存储文件的文件格式。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

acks

Type: string
Default: all
Valid Values: [all, -1, 0, 1]
Importance: low
在考虑请求完成前，生成者需要收到的确认数量。这控制发送的记录的持久性。允许以下设置：

- **acks=0** 如果设为零，则生成者不会等待来自服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。无法保证服务器已收到记录，**重试** 配置不会生效（因为客户端通常不知道任何失败）。每个记录给出的偏移始终设置为 **-1**。

- **acks=1** 意味着领导人将记录写入其本地日志，但不会等待所有后续人的完全确认。在这种情况下，领导机在确认记录后马上失败，但在后续者复制之前，记录将会丢失。

- **acks=all** 意味着领导机将等待一整组同步副本确认记录。这样可保证记录在至少一个同步副本仍然处于活动状态时不会丢失。这是最强的可用保证。这等同于 acks=-1 设置。
  请注意，启用 idempotence 需要此配置值为 'all'。如果设置了冲突配置，且没有显式启用 idempotence，则禁用 idempotence。

auto.include.jmx.reporter

Type: boolean
Default: true
Importance: low
已弃用。是否自动包含 JmxReporter，即使它在 **metric.reporters** 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 **metric.reporters** 中包含 **org.apache.kafka.common.metrics.JmxReporter**，以启用 JmxReporter。

enable.idempotence

Type: boolean
Default: true
Importance: low
当设置为 'true' 时，生成者将确保每个消息的确切副本在流中写入。如果 'false'，则因代理失败而重试的制作者可能会在流中写入重试消息的副本。请注意，启用 idempotence 需要 **max.in.flight.requests.per.connection** 小于或等于 5 （为任何允许值保留消息排序），**重试** 大于 0，ack **s** 必须是 'all'。

如果没有设置任何冲突配置，则默认启用 Idempence。如果设置了冲突配置，且没有显式启用 idempotence，则禁用 idempotence。如果明确启用 idempotence，并且设置了冲突的配置，则会抛出 **ConfigException**。

enable.metrics.push

Type: boolean
Default: true
Importance: low
是否启用将客户端指标推送到集群，如果集群具有与此客户端匹配的客户端指标订阅。

interceptor.classes

type: list
Default: ""
Valid Values: non-null string
Importance: low
用作拦截器的类列表。通过实施 **org.apache.kafka.clients.producer.ProducerInterceptor** 接口，您可以在生成者发布到 Kafka 集群前截获（并可能修改）制作者收到的记录。默认情况下，没有拦截器。

max.in.flight.requests.per.connection

type: int
Default: 5
Valid Values: [1,...]
Importance: low
客户端在阻止前在单个连接上发送的最大未确认请求数。请注意，如果将此配置设置为大于 1，并且 **enable.idempotence** 被设置为 false，则因为重试（例如，启用了重试）失败后有消息重新排序的风险（即，如果已启用重试）；如果禁用重试，或者将 **enable.idempotence** 设置为 true，则排序将保留。此外，启用 idempotence 要求此配置的值应小于或等于 5。如果设置了冲突配置，且没有显式启用 idempotence，则禁用 idempotence。

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。

metadata.max.idle.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [5000,...]
Importance: low
控制制作者将缓存闲置主题的元数据的时长。如果自上次生成了一个主题后经过的时间超过元数据空闲持续时间，则主题的元数据会被忘记，下一次访问将强制进行元数据获取请求。

metric.reporters

type: list
Default: ""
Valid Values: non-null string
Importance: low

用作指标报告器的类列表。实施 **org.apache.kafka.common.metrics.MetricsReporter** 接口，允许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,...]
Importance: low
为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low
计算指标示例的时间窗口。

partitioner.adaptive.partitioning.enable

Type: boolean
Default: true
Importance: low
当设置为 'true' 时，生成者将尝试适应代理性能，并为在更快代理上托管的分区生成更多消息。如果 'false'，则生成者将尝试统一分发消息。注意：如果使用自定义分区器，则此设置无效。

partitioner.availability.timeout.ms

type: long
Default: 0
Valid Values: [0,...]
Importance: low
如果代理无法为 **partitioner.availability.timeout.ms** 时间生成来自分区的请求，则分区器会将该分区视为不可用。如果值为 0，则禁用此逻辑。注：如果使用自定义分区器或 **partitioner.adaptive.partitioning.enable** 设置为 'false'，则此设置无效。

reconnect.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。

reconnect.backoff.ms

type: long

Default: 50
Valid Values: [0,...]
Importance: low
尝试重新连接到给定主机前等待的基本时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到客户端到代理的所有连接尝试。这个值是初始 backoff 值，并为每个连续连接失败指数增加，最高为 **reconnect.backoff.max.ms** 值。

retry.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
当重试请求重复失败的代理时，等待的最大时间（毫秒）。如果提供，每个客户端将为每个失败请求按指数增加 backoff，直到达到这个最大值。要防止所有客户端在重试时同步，一个随机的 jitter 会被应用到 backoff，从而导致 backoff 在以下 20 到 20% 之间的范围内，超过计算值。如果将 **retry.backoff.ms** 设置为大于 **retry.backoff.max.ms**，则 **retry.backoff.max.ms** 将用作开始的常量 backoff，而不会增加任何指数增加。

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 **retry.backoff.max.ms** 值。

sasl.kerberos.kinit.cmd

type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit 命令路径。

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
刷新尝试之间的登录线程睡眠时间。

sasl.kerberos.ticket.renew.jitter

type: double
Default: 0.05
Importance: low
添加到续订时间的随机 jitter 的百分比。

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。

sasl.login.connect.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商连接超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.read.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商读取超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
在刷新凭证时凭证过期前的缓冲时间（以秒为单位）。如果刷新情况比缓冲区秒数接近过期，则刷新将移动，以尽可能多地维护缓冲区时间。法律值介于 0 到 3600 (1 小时)之间；如果没有指定值，则使用默认值 300 (5 分钟)。如果这个值和 sasl.login.refresh.min.period.seconds 超过了凭证剩余的生命周期，则忽略这个值。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
登录刷新线程在刷新凭证前等待的时间（以秒为单位）。法律值介于 0 到 900 (15 分钟)之间；如果没有指定值，则使用默认值 60 (1 分钟)。如果这个值和 sasl.login.refresh.buffer.seconds 都会被忽略，如果它们的总和超过凭证的剩余生命周期。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.factor

type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
登录刷新线程将休眠到与凭证生命周期相关的指定窗口因素，此时将尝试刷新凭证。法律值介于 0.5 (50%)和 1.0 (100%)之间，如果没有指定值，则使用默认值 0.8 (80%)。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.jitter

type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
与凭证的生命周期相关的最大随机 jitter 量添加到登录刷新线程的睡眠时间中。法律值介于 0 到 0.25(25%)之间，如果没有指定值，则使用默认值 0.05(5%)。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.max.ms

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）登录尝试外部身份验证提供程序之间等待的最大等待值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
（可选）登录尝试外部身份验证提供程序期间初始等待的值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
（可选）值（以秒为单位），允许 OAuth/OIDC 身份提供程序和代理间的差别。

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
（可选）代理用逗号分隔的设置来验证是否已为其中一个预期的受众发出 JWT。JWT 将检查是否有标准的 OAuth "aud" 声明，如果设置了这个值，代理将与 JWT 的"aud"声明中的值匹配，以查看是否有完全匹配。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
代理的 （可选）用于验证 JWT 是否已由预期签发者创建的（可选）设置。JWT 将检查是否有标准 OAuth "iss" 声明，如果设置了这个值，代理会完全匹配 JWT 的"iss"声明中的内容。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
（可选）代理在刷新其 JWKS (JSON Web Key Set)缓存之间等待的值，其中包含键以验证 JWT 的签名。

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

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）尝试从外部身份验证提供程序检索 JWKS (JSON Web Key Set)之间的最大等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

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

Type: long
Default: 100
Importance: low
（可选）在 JWKS (JSON Web Key Set)检索外部身份验证提供程序尝试时的初始等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
范围的 OAuth 声明通常命名为 "scope"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以提供用于 JWT 有效负载声明中包含的范围的名称。

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
该主题的 OAuth 声明通常命名为 "sub"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以为 JWT 有效负载的声明中包含的主题提供不同的名称。

security.providers

Type: string
Default: null
Importance: low
每个返回供应商实施安全算法的可配置创建者类列表。这些类应实施 **org.apache.kafka.common.security.auth.SecurityProviderCreator** 接口。

ssl.cipher.suites

Type: list
Default: null
Importance: low
密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。默认情况下，支持所有可用的密码套件。

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
使用服务器证书验证服务器主机名的端点标识算法。

ssl.engine.factory.class

Type: class
Default: null
Importance: low
org.apache.kafka.common.security.auth.SslEngineFactory 类型的类提供 SSLEngine 对象。默认值为 org.apache.kafka.common.security.ssl.DefaultSslEngineFactory。或者，将其设置为 org.apache.kafka.common.security.ssl.CommonNameLoggingSslEngineFactory 将记录客户端用于与

日志级别 INFO 的任何代理进行身份验证的通用 SSL 证书名称。请注意，由于检查客户端提供的证书链，这会在建立从 mTLS 客户端到代理的新连接时造成小延迟。请注意，该实施还使用基于标准 Java 信任存储的自定义信任存储，因此由于标准版本不成熟，因此可能会被视为安全风险。

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
用于 SSL 加密操作的 SecureRandom PRNG 实施。

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。

transaction.timeout.ms

type: int
Default: 60000 (1 minute)
Importance: low
事务在协调者主动中止前保持打开的最大时间（以毫秒为单位）。事务的开头会在向其中添加第一个分区时设置。如果这个值大于代理中的 **transaction.max.timeout.ms** 设置，则请求将失败，并显示 **InvalidTxnTimeoutException** 错误。

transactional.id

Type: string
Default: null
Valid Values: non-empty string
Importance: low
用于事务发送的 TransactionalId。这可实现跨越多个制作者会话的可靠性语义，因为它允许客户端保证在启动任何新事务前使用相同的 TransactionalId 事务已完成。如果没有提供 TransactionalId，则生成者将限制为幂等交付。如果配置了 TransactionalId，则代表 **enable.idempotence**。默认情况下，TransactionId 没有配置，这意味着无法使用事务。请注意，默认情况下，事务需要至少三个代理集群，这是生产环境的建议设置；对于开发，您可以通过调整代理设置 **transaction.state.log.replication.factor**。

第 5 章 管理客户端配置属性

bootstrap.controllers

Type: list
Default: ""
Importance: high
用于建立到 KRaft 控制器仲裁的初始连接的主机/端口对列表。此列表的格式应为 **host1:port1,host2:port2,...**。

bootstrap.servers

Type: list
Default: ""
Importance: high
用于建立到 Kafka 集群的初始连接的主机/端口对列表。客户端将使用所有服务器与此处为引导指定的服务器无关 - 此列表仅影响用于发现完整服务器集的初始主机。此列表的格式应为 **host1:port1,host2:port2,...**。由于这些服务器仅用于初始连接来发现完整的群集成员身份（可能会动态更改），因此此列表不需要包含整组服务器（但是如果服务器停机，您可能需要多个服务器）。

ssl.key.password

Type: password
Default: null
Importance: high
密钥存储文件中私钥的密码或 'ssl.keystore.key' 中指定的 PEM 密钥。

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
使用由 'ssl.keystore.type" 指定的格式的证书链。默认 SSL 引擎工厂仅支持使用 X.509 证书列表的 PEM 格式。

ssl.keystore.key

Type: password
Default: null
Importance: high
使用 'ssl.keystore.type" 指定的格式的私钥。默认 SSL 引擎工厂只支持使用 PKCS#8 密钥的 PEM 格式。如果密钥加密，必须使用 'ssl.key.password' 指定密钥密码。

ssl.keystore.location

Type: string
Default: null
Importance: high
密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。

ssl.keystore.password

Type: password
Default: null
Importance: high
密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 'ssl.keystore.location' 时才需要。PEM 格式不支持密钥存储密码。

ssl.truststore.certificates

Type: password
Default: null
Importance: high
可信证书，格式为 'ssl.truststore.type'。默认 SSL 引擎工厂仅支持使用 X.509 证书使用 PEM 格式。

ssl.truststore.location

Type: string
Default: null
Importance: high
信任存储文件的位置。

ssl.truststore.password

Type: password
Default: null
Importance: high
信任存储文件的密码。如果没有设置密码，则仍然使用配置的信任存储文件，但禁用完整性检查。PEM 格式不支持信任存储密码。

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
控制客户端如何使用 DNS 查找。如果设置为 **use_all_dns_ips**，请按顺序连接到每个返回的 IP 地址，直到成功建立连接。断开连接后，会使用下一个 IP。当所有 IP 都被一次使用后，客户端会再次解析主机名(JVM 和 OS 缓存 DNS 名称查找)中的 IP。如果设置为 **resolve_canonical_bootstrap_servers_only**，请将每个 bootstrap 地址解析为规范名称列表。bootstrap 阶段后，它的行为与 **use_all_dns_ips** 相同。

client.id

Type: string
Default: ""
Importance: medium
在发出请求时传递给服务器的 id 字符串。这样做的目的是通过允许逻辑应用程序名称包含在服务器端请求日志记录中，从而跟踪除 ip/port 以外的请求源。

connections.max.idle.ms

Type: long
Default: 300000 (5 minutes)
Importance: medium
在这个配置指定的毫秒数后关闭闲置连接。

default.api.timeout.ms

type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: medium
指定客户端 API 的超时时间（以毫秒为单位）。此配置用作没有指定 **timeout** 参数的所有客户端操作的默认超时时间。

receive.buffer.bytes

type: int
Default: 65536 (64 kibibytes)
Valid Values: [-1,...]
Importance: medium
读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。如果值为 -1，则使用操作系统默认值。

request.timeout.ms

type: int
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: medium
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 客户端回调处理程序类的完全限定名称。

sasl.jaas.config

Type: password
Default: null
Importance: medium
用于 SASL 连接的 JAAS 登录上下文参数，其格式供 JAAS 配置文件使用。JAAS 配置文件格式描述在此处。该值的格式是： **loginModuleClass controlFlag (optionName=optionValue)*;**。对于代理，配置必须在小写中带有监听前缀和 SASL 机制名称前缀。例如： listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;。

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 登录回调处理程序类的完全限定名称。对于代理，登录回调处理器配置必须以监听器前缀和 SASL 机制名称作为前缀作为前缀。例如：
listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler。

sasl.login.class

Type: class
Default: null
Importance: medium

实现登录接口的类的完全限定名称。对于代理，登录配置必须在小写中带有监听前缀和 SASL 机制名称前缀。For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
用于客户端连接的 SASL 机制。这可能是提供安全提供程序的任何机制。GSSAPI 是默认机制。

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
可以从中检索供应商的 JWKS (JSON Web Key Set) 的 OAuth/OIDC 供应商 URL。URL 可以是基于 HTTP (S)或基于文件的 URL。如果 URL 基于 HTTP (S)，则 JWKS 数据将通过代理启动时配置的 URL 从 OAuth/OIDC 供应商中检索。所有 then-current 密钥都将缓存在代理上以进行传入请求。如果为 JWT 收到了身份验证请求，其中包含缓存中尚未在缓存中的"kid"标头声明值，则将根据需要再次查询 JWKS 端点。但是，代理会轮询每个 sasl.oauthbearer.jwks.endpoint.refresh.ms 毫秒的 URL，以便在收到包含这些密钥的任何 JWT 请求前刷新缓存。如果 URL 基于文件，代理将在启动时从配置的位置加载 JWKS 文件。如果 JWT 包含没有在 JWKS 文件中的 "kid" 标头值，代理将拒绝 JWT 并且身份验证将失败。

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
OAuth/OIDC 身份提供程序的 URL。如果 URL 基于 HTTP (S)，这是签发者的令牌端点 URL，其请求将根据 sasl.jaas.config 中的配置登录。如果 URL 基于文件，它指定一个包含由 OAuth/OIDC 身份提供程序发布的访问令牌(JWT 序列化形式)的文件，以用于授权。

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: (case insensitive)[SASL_SSL, PLAINTEXT, SSL, SASL_PLAINTEXT]
Importance: medium
用于与代理通信的协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。

send.buffer.bytes

type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
发送数据时要使用的 TCP 发送缓冲区(SO_SNDBUF)的大小。如果值为 -1，则使用操作系统默认值。

socket.connection.setup.timeout.max.ms

type: long
Default: 30000 (30 秒)
Importance: medium
客户端等待套接字连接建立的最大时间。当每个连续连接失败时，连接设置超时会达到这个最大值。为避免连接差异，将把一个随机化因值应用于超时，导致计算值高于 20% 的随机范围。

socket.connection.setup.timeout.ms

type: long
Default: 10000 (10 seconds)
Importance: medium
客户端等待套接字连接建立的时间长度。如果在超时时间前没有构建连接，客户端将关闭套接字频道。这个值是初始 backoff 值，每个连续连接失败都会按指数增加，最高为 **socket.connection.setup.timeout.max.ms** 值。

ssl.enabled.protocols

type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
为 SSL 连接启用的协议列表。在使用 Java 11 或更新版本时，默认为 'TLSv1.2,TLSv1.3'，否则 'TLSv1.2'。使用 Java 11 的默认值，如果客户端和服务器同时支持 TLSv1.3，则客户端和服务器将首选 TLSv1.3，否则将回退到 TLSv1.2（假设两者都至少支持 TLSv1.2）。对于大多数情况，这个默认值应该可以正常工作。另请参阅 **ssl.protocol** 的配置文档。

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
密钥存储文件的文件格式。这对客户端是可选的。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

ssl.protocol

type: string
Default: TLSv1.3
Importance: medium
用于生成 SSLContext 的 SSL 协议。使用 Java 11 或更新版本运行时，默认为 'TLSv1.3'，否则 'TLSv1.2'。对于大多数用例，这个值应该可以正常工作。最近的 JVM 中允许的值为 'TLSv1.2' 和 'TLSv1.3'。旧的 JVM 可以支持 'TLS', 'TLSv1.1', 'SSLv2', 'SSLv2' 和 'SSLv3'，但由于已知的安全漏洞，不建议使用它们。使用这个配置和 'ssl.enabled.protocols' 的默认值，如果服务器不支持 'TLSv1.3'，客户端将降级为 'TLSv1.2'。如果此配置被设置为 'TLSv1.2'，客户端将不会使用 'TLSv1.3'，即使它是 ssl.enabled.protocols 中的值之一，服务器只支持 'TLSv1.3'。

ssl.provider

Type: string
Default: null
Importance: medium
用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
信任存储文件的文件格式。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

auto.include.jmx.reporter

Type: boolean
Default: true
Importance: low

已弃用。是否自动包含 JmxReporter，即使它在 **metric.reporters** 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 **metric.reporters** 中包含 **org.apache.kafka.common.metrics.JmxReporter**，以启用 JmxReporter。

enable.metrics.push

Type: boolean
Default: true
Importance: low
是否启用将客户端指标推送到集群，如果集群具有与此客户端匹配的客户端指标订阅。

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。

metric.reporters

Type: list
Default: ""
Importance: low
用作指标报告器的类列表。实施 **org.apache.kafka.common.metrics.MetricsReporter** 接口，允许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,...]
Importance: low
为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low
指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low
计算指标示例的时间窗口。

reconnect.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low

重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。

reconnect.backoff.ms

type: long
Default: 50
Valid Values: [0,...]
Importance: low
尝试重新连接到给定主机前等待的基本时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到客户端到代理的所有连接尝试。这个值是初始 backoff 值，并为每个连续连接失败指数增加，最高为 **reconnect.backoff.max.ms** 值。

retries

type: int
Default: 2147483647
Valid Values: [0,...,2147483647]
Importance: low
设置大于零的值将导致客户端重新发送失败并可能出现临时错误的请求。建议将值设为 0 或 **MAX_VALUE**，并使用对应的超时参数来控制客户端应重试请求的时长。

retry.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
当重试请求重复失败的代理时，等待的最大时间（毫秒）。如果提供，每个客户端将为每个失败请求按指数增加 backoff，直到达到这个最大值。要防止所有客户端在重试时同步，一个随机的 jitter 会被应用到 backoff，从而导致 backoff 在以下 20 到 20% 之间的范围内，超过计算值。如果将 **retry.backoff.ms** 设置为大于 **retry.backoff.max.ms**，则 **retry.backoff.max.ms** 将用作开始的常量 backoff，而不会增加任何指数增加。

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 **retry.backoff.max.ms** 值。

sasl.kerberos.kinit.cmd

type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit 命令路径。

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
刷新尝试之间的登录线程睡眠时间。

sasl.kerberos.ticket.renew.jitter

type: double
Default: 0.05
Importance: low
添加到续订时间的随机 jitter 的百分比。

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。

sasl.login.connect.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商连接超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.read.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商读取超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
在刷新凭证时凭证过期前的缓冲时间（以秒为单位）。如果刷新情况比缓冲区秒数接近过期，则刷新将移动，以尽可能多地维护缓冲区时间。法律值介于 0 到 3600 (1 小时)之间；如果没有指定值，则使用默认值 300 (5 分钟)。如果这个值和 sasl.login.refresh.min.period.seconds 超过了凭证剩余的生命周期，则忽略这个值。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
登录刷新线程在刷新凭证前等待的时间（以秒为单位）。法律值介于 0 到 900 (15 分钟)之间；如果没有指定值，则使用默认值 60 (1 分钟)。如果这个值和 sasl.login.refresh.buffer.seconds 都会被忽略，如果它们的总和超过凭证的剩余生命周期。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.factor

type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low

登录刷新线程将休眠到与凭证生命周期相关的指定窗口因素，此时将尝试刷新凭证。法律值介于 0.5 (50%)和 1.0 (100%)之间，如果没有指定值，则使用默认值 0.8 (80%)。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.jitter

type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
与凭证的生命周期相关的最大随机 jitter 量添加到登录刷新线程的睡眠时间中。法律值介于 0 到 0.25(25%)之间，如果没有指定值，则使用默认值 0.05(5%)。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.max.ms

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）登录尝试外部身份验证提供程序之间等待的最大等待值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
（可选）登录尝试外部身份验证提供程序期间初始等待的值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
（可选）值（以秒为单位），允许 OAuth/OIDC 身份提供程序和代理间的差别。

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
（可选）代理用逗号分隔的设置来验证是否已为其中一个预期的受众发出 JWT。JWT 将检查是否有标准的 OAuth "aud" 声明，如果设置了这个值，代理将与 JWT 的"aud"声明中的值匹配，以查看是否有完全匹配。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
代理的 （可选）用于验证 JWT 是否已由预期签发者创建的（可选）设置。JWT 将检查是否有标准 OAuth "iss" 声明，如果设置了这个值，代理会完全匹配 JWT 的"iss"声明中的内容。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
（可选）代理在刷新其 JWKS (JSON Web Key Set)缓存之间等待的值，其中包含键以验证 JWT 的签名。

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

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）尝试从外部身份验证提供程序检索 JWKS (JSON Web Key Set)之间的最大等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

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

Type: long
Default: 100
Importance: low
（可选）在 JWKS (JSON Web Key Set)检索外部身份验证提供程序尝试时的初始等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
范围的 OAuth 声明通常命名为 "scope"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以提供用于 JWT 有效负载声明中包含的范围的名称。

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
该主题的 OAuth 声明通常命名为 "sub"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以为 JWT 有效负载的声明中包含的主题提供不同的名称。

security.providers

Type: string
Default: null
Importance: low
每个返回供应商实施安全算法的可配置创建者类列表。这些类应实施 **org.apache.kafka.common.security.auth.SecurityProviderCreator** 接口。

ssl.cipher.suites

Type: list
Default: null
Importance: low
密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。默认情况下，支持所有可用的密码套件。

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
使用服务器证书验证服务器主机名的端点标识算法。

ssl.engine.factory.class

Type: class
Default: null
Importance: low
org.apache.kafka.common.security.auth.SslEngineFactory 类型的类提供 SSLEngine 对象。默认值为 org.apache.kafka.common.security.ssl.DefaultSslEngineFactory。或者，将其设置为 org.apache.kafka.common.security.ssl.CommonNameLoggingSslEngineFactory 将记录客户端用于与日志级别 INFO 的任何代理进行身份验证的通用 SSL 证书名称。请注意，由于检查客户端提供的证书链，这会在建立从 mTLS 客户端到代理的新连接时造成小延迟。请注意，该实施还使用基于标准 Java 信任存储的自定义信任存储，因此由于标准版本不成熟，因此可能会被视为安全风险。

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
用于 SSL 加密操作的 SecureRandom PRNG 实施。

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。

第 6 章 KAFKA CONNECT 配置属性

config.storage.topic

type: string
Importance: high
存储连接器配置的 Kafka 主题的名称。

group.id

type: string
Importance: high
标识此 worker 所属的 Connect 集群组的唯一字符串。

key.converter

type: class
Importance: high
转换器类用于在 Kafka Connect 格式和写入 Kafka 的序列化形式之间进行转换。这会控制写入或从 Kafka 读取的消息中的密钥格式，因为这独立于连接器，因此任何连接器都可以使用任何序列化格式。常见格式示例包括 JSON 和 Avro。

offset.storage.topic

type: string
Importance: high
存储源连接器偏移的 Kafka 主题的名称。

status.storage.topic

type: string
Importance: high
存储连接器和任务状态的 Kafka 主题的名称。

value.converter

type: class
Importance: high
转换器类用于在 Kafka Connect 格式和写入 Kafka 的序列化形式之间进行转换。这会控制写入或从 Kafka 读取的消息中的值格式，因为这独立于连接器，因此任何连接器都可以使用任何序列化格式。常见格式示例包括 JSON 和 Avro。

bootstrap.servers

type: list
Default: localhost:9092
Importance: high
用于建立到 Kafka 集群的初始连接的主机/端口对列表。客户端将使用所有服务器与此处为引导指定的服务器无关 - 此列表仅影响用于发现完整服务器集的初始主机。此列表的格式应为 **host1:port1,host2:port2,...**。由于这些服务器仅用于初始连接来发现完整的群集成员身份（可能会动态更改），因此此列表不需要包含整组服务器（但是如果服务器停机，您可能需要多个服务器）。

exactly.once.source.support

Type: string
Default: disabled
Valid Values: (case insensitive)[DISABLED, ENABLED, PREPARING]
Importance: high

是否在使用事务编写源记录及其源偏移前，通过主动隔离旧任务生成前，启用集群中源连接器的准确支持。要在新集群中启用准确的源支持，请将此属性设置为 'enabled'。要在现有集群中启用支持，首先在集群的每个 worker 上设置为 'preparing'，然后设置为 'enabled'。滚动升级可用于这两个更改。有关这个功能的更多信息，请参阅 准确源支持文档。

heartbeat.interval.ms

Type: int
Default: 3000 (3 秒)
Importance: high
使用 Kafka 的组管理功能时，与组协调器的心跳之间预期的时间。心跳用于确保 worker 的会话保持活动状态，并在新成员加入或离开组时促进重新平衡。该值必须小于 **session.timeout.ms**，但设置的值通常不应超过这个值的 1/3。它可以调整甚至较低，以控制正常重新平衡的预期时间。

rebalance.timeout.ms

type: int
Default: 60000 (1 minute)
Importance: high
每个 worker 允许的最大时间在重新平衡后加入组。这基本上是所有任务清除任何待处理数据和提交偏移所需的时间的限制。如果超过了超时，则 worker 将从组中删除，这会导致偏移提交失败。

session.timeout.ms

type: int
Default: 10000 (10 seconds)
Importance: high
用于检测 worker 失败的超时。worker 发送定期心跳以指示其存活度到代理。如果在这个会话超时前代理没有接收心跳，则代理将从组中删除 worker 并启动重新平衡。请注意，该值必须在允许范围内，如 **group.min.session.timeout.ms** 和 **group.max.session.timeout.ms** 中配置。

ssl.key.password

Type: password
Default: null
Importance: high
密钥存储文件中私钥的密码或 'ssl.keystore.key' 中指定的 PEM 密钥。

ssl.keystore.certificate.chain

Type: password
Default: null
Importance: high
使用由 'ssl.keystore.type" 指定的格式的证书链。默认 SSL 引擎工厂仅支持使用 X.509 证书列表的 PEM 格式。

ssl.keystore.key

Type: password
Default: null
Importance: high
使用 'ssl.keystore.type" 指定的格式的私钥。默认 SSL 引擎工厂只支持使用 PKCS#8 密钥的 PEM 格式。如果密钥加密，必须使用 'ssl.key.password' 指定密钥密码。

ssl.keystore.location

Type: string
Default: null
Importance: high
密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。

ssl.keystore.password

Type: password
Default: null
Importance: high
密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 'ssl.keystore.location' 时才需要。
PEM 格式不支持密钥存储密码。

ssl.truststore.certificates

Type: password
Default: null
Importance: high
可信证书，格式为 'ssl.truststore.type'。默认 SSL 引擎工厂仅支持使用 X.509 证书使用 PEM 格式。

ssl.truststore.location

Type: string
Default: null
Importance: high
信任存储文件的位置。

ssl.truststore.password

Type: password
Default: null
Importance: high
信任存储文件的密码。如果没有设置密码，则仍然使用配置的信任存储文件，但禁用完整性检查。
PEM 格式不支持信任存储密码。

client.dns.lookup

Type: string
Default: use_all_dns_ips
Valid Values: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]
Importance: medium
控制客户端如何使用 DNS 查找。如果设置为 **use_all_dns_ips**，请按顺序连接到每个返回的 IP 地址，直到成功建立连接。断开连接后，会使用下一个 IP。当所有 IP 都被一次使用后，客户端会再次解析主机名(JVM 和 OS 缓存 DNS 名称查找)中的 IP。如果设置为 **resolve_canonical_bootstrap_servers_only**，请将每个 bootstrap 地址解析为规范名称列表。bootstrap 阶段后，它的行为与 **use_all_dns_ips** 相同。

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: medium
在这个配置指定的毫秒数后关闭闲置连接。

connector.client.config.override.policy

Type: string
Default: All
Importance: medium
ConnectorClientConfigOverridePolicy 的类名称或别名实现。定义连接器可以覆盖哪些客户端配置。默认实现是 **All**，这意味着连接器配置可以覆盖所有客户端属性。框架中的其他可能策略包括 **None** 来禁止连接者覆盖客户端属性，和 **Principal** 来允许连接者只能覆盖客户端的主体。

receive.buffer.bytes

type: int
Default: 32768 (32 kibibytes)
Valid Values: [-1,...]
Importance: medium
读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。如果值为 -1，则使用操作系统默认值。

request.timeout.ms

type: int
Default: 40000 (40 seconds)
Valid Values: [0,...]
Importance: medium
配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

sasl.client.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 客户端回调处理程序类的完全限定名称。

sasl.jaas.config

Type: password
Default: null
Importance: medium
用于 SASL 连接的 JAAS 登录上下文参数，其格式供 JAAS 配置文件使用。JAAS 配置文件格式描述在此处。该值的格式是： **loginModuleClass controlFlag (optionName=optionValue)*;**。对于代理，配置必须在小写中带有监听前缀和 SASL 机制名称前缀。例如： listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;。

sasl.kerberos.service.name

Type: string
Default: null
Importance: medium
Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。

sasl.login.callback.handler.class

Type: class
Default: null
Importance: medium
实现 AuthenticateCallbackHandler 接口的 SASL 登录回调处理程序类的完全限定名称。对于代理，登录回调处理器配置必须以监听器前缀和 SASL 机制名称作为前缀作为前缀。例如：
listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler。

sasl.login.class

Type: class
Default: null
Importance: medium
实现登录接口的类的完全限定名称。对于代理，登录配置必须在小写中带有监听前缀和 SASL 机制名称前缀。For example, listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin.

sasl.mechanism

Type: string
Default: GSSAPI
Importance: medium
用于客户端连接的 SASL 机制。这可能是提供安全提供程序的任何机制。GSSAPI 是默认机制。

sasl.oauthbearer.jwks.endpoint.url

Type: string
Default: null
Importance: medium
可以从中检索供应商的 JWKS (JSON Web Key Set) 的 OAuth/OIDC 供应商 URL。URL 可以是基于 HTTP (S)或基于文件的 URL。如果 URL 基于 HTTP (S)，则 JWKS 数据将通过代理启动时配置的 URL 从 OAuth/OIDC 供应商中检索。所有 then-current 密钥都将缓存在代理上以进行传入请求。如果为 JWT 收到了身份验证请求，其中包含缓存中尚未在缓存中的"kid"标头声明值，则将根据需要再次查询 JWKS 端点。但是，代理会轮询每个 sasl.oauthbearer.jwks.endpoint.refresh.ms 毫秒的 URL，以便在收到包含这些密钥的任何 JWT 请求前刷新缓存。如果 URL 基于文件，代理将在启动时从配置的位置加载 JWKS 文件。如果 JWT 包含没有在 JWKS 文件中的 "kid" 标头值，代理将拒绝 JWT 并且身份验证将失败。

sasl.oauthbearer.token.endpoint.url

Type: string
Default: null
Importance: medium
OAuth/OIDC 身份提供程序的 URL。如果 URL 基于 HTTP (S)，这是签发者的令牌端点 URL，其请求将根据 sasl.jaas.config 中的配置登录。如果 URL 基于文件，它指定一个包含由 OAuth/OIDC 身份提供程序发布的访问令牌(JWT 序列化形式)的文件，以用于授权。

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: (case insensitive)[SASL_SSL, PLAINTEXT, SSL, SASL_PLAINTEXT]
Importance: medium
用于与代理通信的协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。

send.buffer.bytes

type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,...]
Importance: medium
发送数据时要使用的 TCP 发送缓冲区(SO_SNDBUF)的大小。如果值为 -1，则使用操作系统默认值。

ssl.enabled.protocols

type: list
Default: TLSv1.2,TLSv1.3
Importance: medium
为 SSL 连接启用的协议列表。在使用 Java 11 或更新版本时，默认为 'TLSv1.2,TLSv1.3'，否则 'TLSv1.2'。使用 Java 11 的默认值，如果客户端和服务器同时支持 TLSv1.3，则客户端和服务器将首选 TLSv1.3，否则将回退到 TLSv1.2（假设两者都至少支持 TLSv1.2）。对于大多数情况，这个默认值应该可以正常工作。另请参阅 **ssl.protocol** 的配置文档。

ssl.keystore.type

Type: string
Default: JKS
Importance: medium
密钥存储文件的文件格式。这对客户端是可选的。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

ssl.protocol

type: string
Default: TLSv1.3
Importance: medium
用于生成 SSLContext 的 SSL 协议。使用 Java 11 或更新版本运行时，默认为 'TLSv1.3'，否则 'TLSv1.2'。对于大多数用例，这个值应该可以正常工作。最近的 JVM 中允许的值为 'TLSv1.2' 和 'TLSv1.3'。旧的 JVM 可以支持 'TLS', 'TLSv1.1', 'SSLv2', 'SSLv2' 和 'SSLv3'，但由于已知的安全漏洞，不建议使用它们。使用这个配置和 'ssl.enabled.protocols' 的默认值，如果服务器不支持 'TLSv1.3'，客户端将降级为 'TLSv1.2'。如果此配置被设置为 'TLSv1.2'，客户端将不会使用 'TLSv1.3'，即使它是 ssl.enabled.protocols 中的值之一，服务器只支持 'TLSv1.3'。

ssl.provider

Type: string
Default: null
Importance: medium
用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。

ssl.truststore.type

Type: string
Default: JKS
Importance: medium
信任存储文件的文件格式。默认 **ssl.engine.factory.class** 当前支持的值为 [JKS、PKCS12、PEM]。

worker.sync.timeout.ms

Type: int
Default: 3000 (3 秒)
Importance: medium
当 worker 与其他 worker 不同步且需要重新同步配置时，请在放弃前等待这个时间，保留组，并在重新加入前等待 backoff 周期。

worker.unsync.backoff.ms

Type: int
Default: 300000 (5 minutes)
Importance: medium
当 worker 没有与其他 worker 同步，且无法在 worker.sync.timeout.ms 中捕获时，在重新加入前保留 Connect 集群。

access.control.allow.methods

Type: string
Default: ""
Importance: low
通过设置 Access-Control-Allow-Methods 标头来设置跨原始请求支持的方法。Access-Control-Allow-Methods 标头的默认值允许对 GET、POST 和 HEAD 的跨原始请求。

access.control.allow.origin

Type: string
Default: ""
Importance: low
将 Access-Control-Allow-Origin 标头设置为 REST API 请求的值。要启用跨原始访问，将其设置为允许访问 API 的应用程序的域，或 '*' 允许从任何域进行访问。默认值只允许从 REST API 的域访问。

admin.listeners

type: list
Default: null
Valid Values: comma- separated URL 列表，例如： http://localhost:8080,https://localhost:8443.
Importance: low
Admin REST API 将侦听的、以逗号分隔的 URI 列表。支持的协议有 HTTP 和 HTTPS。空字符串或空字符串将禁用此功能。默认行为是使用常规监听程序（由 'listeners' 属性指定）。

auto.include.jmx.reporter

Type: boolean
Default: true
Importance: low
已弃用。是否自动包含 JmxReporter，即使它在 **metric.reporters** 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 **metric.reporters** 中包含 **org.apache.kafka.common.metrics.JmxReporter**，以启用 JmxReporter。

client.id

Type: string
Default: ""
Importance: low
在发出请求时传递给服务器的 id 字符串。这样做的目的是通过允许逻辑应用程序名称包含在服务器端请求日志记录中，从而跟踪除 ip/port 以外的请求源。

config.providers

Type: list
Default: ""
Importance: low
以逗号分隔的 **ConfigProvider** 类名称，加载并按指定顺序使用。通过实施接口 **ConfigProvider**，您可以替换连接器配置中的变量引用，如用于外部化 secret。

config.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the Kafka 集群中的代理数，或 -1 使用代理的默认
导入： low
创建配置存储主题时使用的复制因素。

connect.protocol

Type: string
Default: sessioned
Valid Values: [eager, compatible, sessioned]
Importance: low
Kafka Connect 协议的兼容性模式。

header.converter

type: class
Default: org.apache.kafka.connect.storage.SimpleHeaderConverter
Importance: low
HeaderConverter 类，用于在 Kafka Connect 格式和写入 Kafka 的序列化表单之间进行转换。这控制写入或从 Kafka 读取的消息中的标头值格式，因为这独立于连接器，因此任何连接器都可以使用任何序列化格式。常见格式示例包括 JSON 和 Avro。默认情况下，simpleHeaderConverter 用于将标头值序列化为字符串，并通过推断模式来反序列化它们。

inter.worker.key.generation.algorithm

Type: string
Default: HmacSHA256
Valid Values: worker JVM
Importance: low 支持的任何 KeyGenerator 算法
用于生成内部请求密钥的算法。算法 'HmacSHA256' 将在支持它的 JVM 上用作默认值；在其他 JVM 中，不会使用默认值，且必须在 worker 配置中手动指定此属性的值。

inter.worker.key.size

type: int
Default: null
Importance: low
用于签署内部请求的密钥大小（以位为单位）。如果为 null，则使用密钥生成算法的默认密钥大小。

inter.worker.key.ttl.ms

type: int
Default: 3600000 (1 hour)
Valid Values: [0,...,2147483647]
Importance: low
用于内部请求验证生成的会话密钥的 TTL （以毫秒为单位）。

inter.worker.signature.algorithm

Type: string
Default: HmacSHA256
Valid Values: worker JVM
Importance: low 支持的任何 MAC 算法
用于签署内部请求 The algorithm 'inter.worker.signature.algorithm' 的算法将用作支持它的 JVM 的默认；在其他 JVM 中，不使用默认值，必须在 worker 配置中手动指定此属性的值。

inter.worker.verification.algorithms

Type: list
Default: HmacSHA256
Valid Values: 一个或多个 MAC 算法列表，每个算法都由 worker JVM
Importance: low 支持

验证内部请求的允许算法列表，其中必须包含用于 inter.worker.signature.algorithm 属性的算法。算法 '[HmacSHA256]' 将用作提供它们的 JVM 中的默认算法；在其他 JVM 中，不会使用默认值，且必须在 worker 配置中手动指定此属性的值。

监听器

type: list
Default: http://:8083
Valid Values: comma- separated URL 列表，例如： http://localhost:8080,https://localhost:8443.
Importance: low
REST API 将要侦听的、以逗号分隔的 URI 列表。支持的协议有 HTTP 和 HTTPS。将 hostname 指定为 0.0.0.0 以绑定到所有接口。将 hostname 留空，以绑定到默认接口。法律侦听器列表示例：HTTP://myhost:8083,HTTPS://myhost:8084。

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,...]
Importance: low
即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。

metric.reporters

Type: list
Default: ""
Importance: low
用作指标报告器的类列表。实施 **org.apache.kafka.common.metrics.MetricsReporter** 接口，允许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,...]
Importance: low
为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG]
Importance: low
指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low
计算指标示例的时间窗口。

offset.flush.interval.ms

type: long
Default: 60000 (1 minute)
Importance: low
尝试为任务提交偏移的时间间隔。

offset.flush.timeout.ms

Type: long
Default: 5000 (5 seconds)
Importance: low
在取消进程并恢复偏移数据以在以后尝试提交的偏移数据前，等待记录刷新和分区偏移数据的最大毫秒数。此属性对使用准确支持运行的源连接器没有影响。

offset.storage.partitions

Type: int
Default: 25
Valid Values: Positive number, 或 -1 使用代理的默认
导入： low
创建偏移存储主题时使用的分区数量。

offset.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the Kafka 集群中的代理数，或 -1 使用代理的默认
导入： low
创建偏移存储主题时使用的复制因素。

plugin.discovery

Type: string
Default: hybrid_warn
Valid Values: (case insensitive)[ONLY_SCAN, SERVICE_LOAD, HYBRID_WARN, HYBRID_FAIL]
Importance: low
用于发现 classpath 和 plugin.path 配置中存在的插件的方法。这可以是带有以下含义的多个值之一：* only_scan: Discover plugins only by reflection。ServiceLoader 无法发现的插件不会影响 worker 启动。* hybrid_warn: 发现插件，反映了 ServiceLoader 和 ServiceLoader。ServiceLoader 无法发现的插件会在 worker 启动过程中打印警告。* hybrid_fail: 发现插件，反映在 ServiceLoader 和 ServiceLoader。ServiceLoader 无法发现的插件将导致 worker 启动失败。* service_load: 仅由 ServiceLoader 发现插件。比其他模式更快启动。无法被 ServiceLoader 发现的插件可能无法使用。

plugin.path

Type: list
Default: null
Importance: low
使用逗号(,)分隔的路径列表，其中包含插件(connectors, converters, transformations)。该列表应当由包含任何组合的顶级目录组成，这些目录会立即包含插件及其依赖项 b 的 jar，以及带有插件及其依赖项 c 的 uber-jars，即立即包含插件类别及其依赖项结构的软件包目录结构及其依赖项 注意：符号链接将跟进来发现依赖项或插件。示例：
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors Do not use config provider variables，因为 worker 的扫描程序在配置供应商被初始化并用来替换变量。

reconnect.backoff.max.ms

type: long
Default: 1000 (1 second)

Valid Values: [0,...]
Importance: low
重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。

reconnect.backoff.ms

type: long
Default: 50
Valid Values: [0,...]
Importance: low
尝试重新连接到给定主机前等待的基本时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到客户端到代理的所有连接尝试。这个值是初始 backoff 值，并为每个连续连接失败指数增加，最高为 **reconnect.backoff.max.ms** 值。

response.http.headers.config

Type: string
Default: ""
Valid Values: Comma- separated header rules，其中每个标头规则都是 '[action] [header name]: [header value]'，如果标头规则的任何部分包含逗号
导入： low
REST API HTTP 响应标头规则。

rest.advertised.host.name

Type: string
Default: null
Importance: low
如果设置了，这是将分配给其他要连接的 worker 的主机名。

rest.advertised.listener

Type: string
Default: null
Importance: low
设置公告的监听程序(HTTP 或 HTTPS)，它将提供给其他 worker 使用。

rest.advertised.port

type: int
Default: null
Importance: low
如果设置了此项，这是将分配给其他要连接的 worker 的端口。

rest.extension.classes

Type: list
Default: ""
Importance: low
以逗号分隔的 **ConnectRestExtension** 类名称，按指定顺序加载并调用。通过实现接口 **ConnectRestExtension**，您可以注入 Connect 的 REST API 用户定义的资源，如过滤器。通常用于添加自定义功能，如日志记录、安全性等。

retry.backoff.max.ms

type: long

Default: 1000 (1 second)
Valid Values: [0,...]
Importance: low
当重试请求重复失败的代理时，等待的最大时间（毫秒）。如果提供，每个客户端将为每个失败请求按指数增加 backoff，直到达到这个最大值。要防止所有客户端在重试时同步，一个随机的 jitter 会被应用到 backoff，从而导致 backoff 在以下 20 到 20% 之间的范围内，超过计算值。如果将 **retry.backoff.ms** 设置为大于 **retry.backoff.max.ms**，则 **retry.backoff.max.ms** 将用作开始的常量 backoff，而不会增加任何指数增加。

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low
尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 **retry.backoff.max.ms** 值。

sasl.kerberos.kinit.cmd

type: string
Default: /usr/bin/kinit
Importance: low
Kerberos kinit 命令路径。

sasl.kerberos.min.time.before.relogin

Type: long
Default: 60000
Importance: low
刷新尝试之间的登录线程睡眠时间。

sasl.kerberos.ticket.renew.jitter

type: double
Default: 0.05
Importance: low
添加到续订时间的随机 jitter 的百分比。

sasl.kerberos.ticket.renew.window.factor

Type: double
Default: 0.8
Importance: low
登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。

sasl.login.connect.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商连接超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.read.timeout.ms

type: int
Default: null
Importance: low
（可选）外部身份验证供应商读取超时的值（以毫秒为单位）。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.buffer.seconds

Type: short
Default: 300
Valid Values: [0,...,3600]
Importance: low
在刷新凭证时凭证过期前的缓冲时间（以秒为单位）。如果刷新情况比缓冲区秒数接近过期，则刷新将移动，以尽可能多地维护缓冲区时间。法律值介于 0 到 3600 (1 小时)之间；如果没有指定值，则使用默认值 300 (5 分钟)。如果这个值和 sasl.login.refresh.min.period.seconds 超过了凭证剩余的生命周期，则忽略这个值。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.min.period.seconds

Type: short
Default: 60
Valid Values: [0,...,900]
Importance: low
登录刷新线程在刷新凭证前等待的时间（以秒为单位）。法律值介于 0 到 900 (15 分钟)之间；如果没有指定值，则使用默认值 60 (1 分钟)。如果这个值和 sasl.login.refresh.buffer.seconds 都会被忽略，如果它们的总和超过凭证的剩余生命周期。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.factor

type: double
Default: 0.8
Valid Values: [0.5,...,1.0]
Importance: low
登录刷新线程将休眠到与凭证生命周期相关的指定窗口因素，此时将尝试刷新凭证。法律值介于 0.5 (50%)和 1.0 (100%)之间，如果没有指定值，则使用默认值 0.8 (80%)。目前仅适用于 OAUTHBEARER。

sasl.login.refresh.window.jitter

type: double
Default: 0.05
Valid Values: [0.0,...,0.25]
Importance: low
与凭证的生命周期相关的最大随机 jitter 量添加到登录刷新线程的睡眠时间中。法律值介于 0 到 0.25(25%)之间，如果没有指定值，则使用默认值 0.05(5%)。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.max.ms

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）登录尝试外部身份验证提供程序之间等待的最大等待值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.login.retry.backoff.ms

Type: long
Default: 100
Importance: low
（可选）登录尝试外部身份验证提供程序期间初始等待的值（以毫秒为单位）。登录使用基于 sasl.login.retry.backoff.ms 设置的初始等待的指数 backoff 算法，并在尝试到 sasl.login.retry.backoff.max.ms 设置指定的最大等待长度之间加倍。目前仅适用于 OAUTHBEARER。

sasl.oauthbearer.clock.skew.seconds

Type: int
Default: 30
Importance: low
（可选）值（以秒为单位），允许 OAuth/OIDC 身份提供程序和代理间的差别。

sasl.oauthbearer.expected.audience

Type: list
Default: null
Importance: low
（可选）代理用逗号分隔的设置来验证是否已为其中一个预期的受众发出 JWT。JWT 将检查是否有标准的 OAuth "aud" 声明，如果设置了这个值，代理将与 JWT 的"aud"声明中的值匹配，以查看是否有完全匹配。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.expected.issuer

Type: string
Default: null
Importance: low
代理的 （可选）用于验证 JWT 是否已由预期签发者创建的（可选）设置。JWT 将检查是否有标准 OAuth "iss" 声明，如果设置了这个值，代理会完全匹配 JWT 的"iss"声明中的内容。如果没有匹配项，代理将拒绝 JWT，身份验证将失败。

sasl.oauthbearer.jwks.endpoint.refresh.ms

Type: long
Default: 3600000 (1 hour)
Importance: low
（可选）代理在刷新其 JWKS (JSON Web Key Set)缓存之间等待的值，其中包含键以验证 JWT 的签名。

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

type: long
Default: 10000 (10 seconds)
Importance: low
（可选）尝试从外部身份验证提供程序检索 JWKS (JSON Web Key Set)之间的最大等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大等待长度之间加倍。

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

Type: long
Default: 100
Importance: low
（可选）在 JWKS (JSON Web Key Set)检索外部身份验证提供程序尝试时的初始等待值（以毫秒为单位）。JWKS 检索使用基于 sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 设置的初始等待算法的指数 backoff 算法，并在尝试到 sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 设置所指定的最大

等待长度之间加倍。

sasl.oauthbearer.scope.claim.name

Type: string
Default: scope
Importance: low
范围的 OAuth 声明通常命名为 "scope"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以提供用于 JWT 有效负载声明中包含的范围的名称。

sasl.oauthbearer.sub.claim.name

Type: string
Default: sub
Importance: low
该主题的 OAuth 声明通常命名为 "sub"，但如果 OAuth/OIDC 供应商为该声明使用不同的名称，则此（可选）设置可以为 JWT 有效负载的声明中包含的主题提供不同的名称。

scheduled.rebalance.max.delay.ms

Type: int
Default: 300000 (5 minutes)
Valid Values: [0,...,2147483647]
Importance: low
调度的最大延迟，以便在重新平衡和将连接器和任务重新分配到组前等待一个或多个分离的 worker 返回。在此期间，未分配的 worker 连接器和任务保持不变。

socket.connection.setup.timeout.max.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low
客户端等待套接字连接建立的最大时间。当每个连续连接失败时，连接设置超时会达到这个最大值。为避免连接差异，将把一个随机化因值应用于超时，导致计算值高于 20% 的随机范围。

socket.connection.setup.timeout.ms

type: long
Default: 10000 (10 秒)
Valid Values: [0,...]
Importance: low
客户端等待套接字连接建立的时间长度。如果在超时时间前没有构建连接，客户端将关闭套接字频道。这个值是初始 backoff 值，每个连续连接失败都会按指数增加，最高为 **socket.connection.setup.timeout.max.ms** 值。

ssl.cipher.suites

Type: list
Default: null
Importance: low
密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。默认情况下，支持所有可用的密码套件。

ssl.client.auth

Type: string
Default: none

Valid Values: [required, requested, none]
Importance: low
配置 kafka 代理以请求客户端身份验证。以下设置是通用的：

- 如果设为所需的客户端身份验证，则需要 **ssl.client.auth=required**。
- **ssl.client.auth=requested** 意味着客户端身份验证是可选的。与需要不同，如果设置此选项，则可以选择不提供有关其自身的身份验证信息
- **SSL.client.auth=none** 意味着不需要客户端身份验证。

ssl.endpoint.identification.algorithm

Type: string
Default: https
Importance: low
使用服务器证书验证服务器主机名的端点标识算法。

ssl.engine.factory.class

Type: class
Default: null
Importance: low
org.apache.kafka.common.security.auth.SslEngineFactory 类型的类提供 SSLEngine 对象。默认值为 org.apache.kafka.common.security.ssl.DefaultSslEngineFactory。或者，将其设置为 org.apache.kafka.common.security.ssl.CommonNameLoggingSslEngineFactory 将记录客户端用于与日志级别 INFO 的任何代理进行身份验证的通用 SSL 证书名称。请注意，由于检查客户端提供的证书链，这会在建立从 mTLS 客户端到代理的新连接时造成小延迟。请注意，该实施还使用基于标准 Java 信任存储的自定义信任存储，因此由于标准版本不成熟，因此可能会被视为安全风险。

ssl.keymanager.algorithm

Type: string
Default: SunX509
Importance: low
密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。

ssl.secure.random.implementation

Type: string
Default: null
Importance: low
用于 SSL 加密操作的 SecureRandom PRNG 实施。

ssl.trustmanager.algorithm

Type: string
Default: PKIX
Importance: low
信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。

status.storage.partitions

Type: int
Default: 5
Valid Values: Positive number, 或 -1 使用代理的默认
导入： low
创建状态存储主题时使用的分区数量。

status.storage.replication.factor

Type: short
Default: 3
Valid Values: Positive number not larger than the Kafka 集群中的代理数，或 -1 使用代理的默认
导入： low
创建状态存储主题时使用的复制因素。

task.shutdown.graceful.timeout.ms

Type: long
Default: 5000 (5 seconds)
Importance: low
正常等待任务关闭的时间长度。这是总时间，而不是每个任务。所有任务都触发了关闭，然后按顺序等待它们。

topic.creation.enable

Type: boolean
Default: true
Importance: low
是否允许自动创建源连接器使用的主题，当源连接器配置了 **topic.creation.** 属性时。每个任务都将使用管理员客户端创建其主题，而不依赖于 Kafka 代理自动创建主题。

topic.tracking.allow.reset

Type: boolean
Default: true
Importance: low
如果设置为 true，则允许用户请求为每个连接器重置活跃主题的集合。

topic.tracking.enable

Type: boolean
Default: true
Importance: low
在运行时，启用跟踪每个连接器的活跃主题集合。

第 7 章 KAFKA STREAMS 配置属性

application.id

type: string
Importance: high
流处理应用的标识符。在 Kafka 集群中必须是唯一的。它被用作默认 client-id 前缀，即 2）用于成员资格管理的 group-id，3） changelog 主题前缀。

bootstrap.servers

type: list
Importance: high
用于建立到 Kafka 集群的初始连接的主机/端口对列表。客户端将使用所有服务器与此处为引导指定的服务器无关 - 此列表仅影响用于发现完整服务器集的初始主机。此列表的格式应为 **host1:port1,host2:port2,...**。由于这些服务器仅用于初始连接来发现完整的群集成员身份（可能会动态更改），因此此列表不需要包含整组服务器（但是如果服务器停机，您可能需要多个服务器）。

num.standby.replicas

Type: int
Default: 0
Importance: high
每个任务的待机副本数。

state.dir

type: string
Default: /tmp/kafka-streams
Importance: high
状态存储的目录位置。这个路径对于共享同一底层文件系统的每个流实例都必须是唯一的。

acceptable.recovery.lag

type: long
Default: 10000
Valid Values: [0,...]
Importance: medium
可接受的最大滞后（要捕获的偏移数），客户端被视为足以接收活跃的任务分配。在分配后，它仍然会在处理前恢复更改日志的其余部分。为了避免在重新平衡过程中暂停处理，这个配置应该在一个给定工作负载的一分钟内与恢复时间相对应。必须至少为 0。

cache.max.bytes.buffering

type: long
Default: 10485760
Valid Values: [0,...]
Importance: medium
在所有线程之间用于缓冲的最大内存字节数。

client.id

Type: string
Default: ""
Importance: medium
一个 ID 前缀字符串，用于内部 [main-|restore-|global-]consumer, producer, 和 admin 客户端，格式为 < **client.id>-[Global]StreamThread[-<threadSequenceNumber$gt;]-<consumer|producer|restore-consumer|global-consumer** >。

default.deserialization.exception.handler

type: class
Default: org.apache.kafka.streams.errors.LogAndFailExceptionHandler
Importance: medium
实现 **org.apache.kafka.streams.errors.DeserializationExceptionHandler** 接口的异常处理类。

default.key.serde

Type: class
Default: null
Importance: medium
用于实现 **org.apache.kafka.common.serialization.Serde** 接口的密钥的默认序列化器/反序列化器类。请注意，当使用窗口的 serde 类时，还需要设置实现 **org.apache.kafka.common.serialization.Serde** 接口（通过 'default.windowed.key.serde.inner' 或 'default.windowed.value.serde.inner'）的 inner serde 类。

default.list.key.serde.inner

Type: class
Default: null
Importance: medium
默认内部类是实施 **org.apache.kafka.common.serialization.Serde** 接口的关键。只有在 **default.key.serde** 配置被设置为 **org.apache.kafka.common.serialization.Serdes.ListSerde** 时，才会读取此配置。

default.list.key.serde.type

Type: class
Default: null
Importance: medium
用于实施 **java.util.List** 接口的密钥的默认类。当使用列表 serde 类时，只有 **default.key.serde** 配置被设置为 **org.apache.kafka.common.serialization.Serdes.ListSerde** s.ListSerde，才会读取此配置，它需要设置实现 **org.apache.kafka.common.serialization.Serde** 接口的 inner serde 类。

default.list.value.serde.inner

Type: class
Default: null
Importance: medium
默认内部类是实施 **org.apache.kafka.common.serialization.Serde** 接口的值。只有在 **default.value.serde** 配置被设置为 **org.apache.kafka.common.serialization.Serdes.ListSerde** 时，才会读取此配置。

default.list.value.serde.type

Type: class
Default: null
Importance: medium
用于实现 **java.util.List** 接口的值的默认类。只有在 **default.value.serde** 配置且只有这个设置被设置为 **org.apache.kafka.common.serialization.Serdes.ListSerde** 是才会从这个配置中读取。请注意，当使用 list serde 类时，一个需要设置 inner serde 类，它实现了 **org.apache.kafka.common.serialization.Serde** 接口（通过 'default.list.value.serde.inner'）。

default.production.exception.handler

type: class
Default: org.apache.kafka.streams.errors.DefaultProductionExceptionHandler
Importance: medium
实现 **org.apache.kafka.streams.errors.ProductionExceptionHandler** 接口的异常处理类。

default.timestamp.extractor

type: class
Default: org.apache.kafka.streams.processor.FailOnInvalidTimestamp
Importance: medium
实现 **org.apache.kafka.streams.processor.TimestampExtractor** 接口的默认时间戳提取器类。

default.value.serde

Type: class
Default: null
Importance: medium
默认序列化器/反序列化器类用于实现 **org.apache.kafka.common.serialization.Serde** 接口的值。请注意，当使用窗口的 serde 类时，还需要设置实现 **org.apache.kafka.common.serialization.Serde** 接口（通过 'default.windowed.key.serde.inner' 或 'default.windowed.value.serde.inner'）的 inner serde 类。

max.task.idle.ms

Type: long
Default: 0
Importance: medium
此配置控制加入和合并是否可能会产生超出顺序的结果。当流任务完全在某些（但不全部）输入分区上等待生成者发送额外记录并避免在多个输入流间处理时，配置值是流任务的最大时间（以毫秒为单位）。默认（零）不会等待生成者发送更多记录，但它会等待代理上已存在的数据。此默认值意味着对于已在代理上存在的记录，流将以时间戳顺序处理它们。设置为 -1 可完全禁用闲置并完全处理任何本地可用数据，即使这样做可能会生成没有顺序处理。

max.warmup.replicas

type: int
Default: 2
Valid Values: [1,...]
Importance: medium
一次可以一次分配的 warmup 副本的最大数量（除配置的 num.standbys 之外），保持任务在一个实例上可用，同时将其重新分配给另一个实例。用于节流额外的代理流量和集群状态可用于高可用性。必须至少为 1.Note，一个 warmup 副本对应于一个流任务。另外，请注意，每个温副本只能在重新平衡期间提升到活跃任务（通常在所谓的探测重新平衡过程中，这会在 **probing.rebalance.interval.ms** 配置指定的频率时发生）。这意味着，活跃任务可以从一个 Kafka Streams 实例迁移到另一个实例的最大速率由(**max.warmup.replicas** / **probing.rebalance.interval.ms**)决定。

num.stream.threads

Type: int
Default: 1
Importance: medium
执行流处理的线程数量。

processing.guarantee

Type: string
Default: at_least_once
Valid Values: [at_least_once, exactly_once, exactly_once_beta, exactly_once_v2]

Importance: medium
处理保证应使用。可能的值有 **at_least_once** （默认）和 **exactly_once_v2**（需要代理版本 2.5 或更高版本）。弃用的选项为 **exactly_once**（需要代理版本 0.11.0 或更高版本）和 **exactly_once_beta**（需要代理版本 2.5 或更高版本）。请注意，完全处理需要至少三个代理的集群，这是生产的建议设置；通过调整代理设置 **transaction.state.log.replication.factor** 和 **transaction.state.log.min.isr**。

rack.aware.assignment.non_overlap_cost

type: int
Default: null
Importance: medium
与从现有分配移动任务相关的成本。这个 config 和 **rack.aware.assignment.assignment.traffic_cost** 控制优化算法是否首选最小化跨机架流量，或最小化现有分配中的任务移动。如果设置较大的值 **org.apache.kafka.streams.processor.internals.assignment.RackAwareTaskAssignor** 将优化以维护现有分配。默认值为 null，这意味着它将在不同的分配器中使用默认的 non_overlap 成本值。

rack.aware.assignment.strategy

Type: string
Default: none
Valid Values: [none, min_traffic, balance_subtopology]
Importance: medium
我们用于机架感知分配的策略。在分配任务以最小化跨机架流量时，机架感知分配将把 **client.rack** 和 **racks** 纳入考量。**有效设置是：none （默认），它将禁用机架感知分配； min_traffic 将计算最小跨机架流量分配； balance_subtopology，它将计算最小的跨机架流量，并尝试在不同客户端之间平衡相同子策略的任务。**

rack.aware.assignment.tags

type: list
Default: ""
Valid Values: List contains maximum 5 elements
Importance: medium

用于在 Kafka Streams 实例之间分发备用副本的客户端标签密钥列表。配置后，Kafka Streams 将有一个 best-effort 来在每个客户端标签维度上分发待机任务。

rack.aware.assignment.traffic_cost

type: int
Default: null
Importance: medium

与跨机架流量相关的成本。此 config 和 rack.aware.assignment.non_overlap_cost 控制优化算法是否首选最小化跨机架流量，或最小化现有分配中的任务移动。如果设置较大的值 org.apache.kafka.streams.processor.internals.assignment.RackAwareTaskAssignor 将优化以最小化跨机架流量。默认值为 null，这意味着它将在不同的分配器中使用默认流量成本值。

replication.factor

type: int
Default: -1
Importance: medium

更改由流处理应用程序创建的日志主题和重新分区主题的复制因素。默认值 -1 （主题：使用代理默认复制因素）需要代理版本 2.4 或更新版本。

security.protocol

Type: string
Default: PLAINTEXT
Valid Values: (case insensitive)[SASL_SSL, PLAINTEXT, SSL, SASL_PLAINTEXT]
Importance: medium

用于与代理通信的协议。有效值为： PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL。

statestore.cache.max.bytes

type: long
Default: 10485760 (10 mebibytes)
Valid Values: [0,…]
Importance: medium

在所有线程中用于 statestore 缓存的最大内存字节数。

task.timeout.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,…]
Importance: medium

任务因为内部错误而停止的最大时间（以毫秒为单位），并重试直到引发错误为止。对于超时为 0ms，任务会引发第一个内部错误的错误。对于大于 0ms 的任何超时，任务将在引发错误前至少重试一次。

topology.optimization

type: string

Default: none
Valid Values:
org.apache.kafka.streams.StreamsConfig$$Lambda$31/0x00007f269c00d000@7e32c033
Importance: medium

如果应该优化拓扑以及要应用的优化，则告知 Kafka Streams 的配置。可接受值为："NO_OPTIMIZATION", "OPTIMIZE", "OPTIMIZE"，或以逗号分隔的特定优化列表："REUSE_KTABLE_SOURCE_TOPICS", "MERGE_REPARTITION_TOPICS" + "SINGLE_STORE_SELF_JOIN+"）."NO_OPTIMIZATION", "MERGE_REPARTITION_TOPATION"

application.server

Type: string
Default: ""
Importance: low

host:port 对指向用户定义的端点，可用于此 KafkaStreams 实例上的状态存储发现和交互式查询。

auto.include.jmx.reporter

Type: boolean
Default: true
Importance: low

已弃用。是否自动包含 JmxReporter，即使它在 metric.reporters 中未列出。此配置将在 Kafka 4.0 中删除，用户应在 metric.reporters 中包含 org.apache.kafka.common.metrics.JmxReporter，以启用 JmxReporter。

buffered.records.per.partition

type: int
Default: 1000
Importance: low

每个分区要缓冲的最大记录数。

built.in.metrics.version

Type: string

Default: latest
Valid Values: [latest]
Importance: low

要使用的内置指标的版本。

commit.interval.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low

提交处理进度的频率（以毫秒为单位）。对于 at-least-once 处理，提交方法可以保存处理器的位置（如偏移）。对于精确处理，需要提交包含保存位置的事务，并使输出主题中的提交数据对具有隔离级别 read_committed 的用户可见。（请注意，如果 processing.guarantee 设置为 exactly_once_v2, exactly_once，则默认值为 100，否则默认值为 30000。

connections.max.idle.ms

Type: long
Default: 540000 (9 minutes)
Importance: low

在这个配置指定的毫秒数后关闭闲置连接。

default.client.supplier

Type: class
Default: org.apache.kafka.streams.processor.internals.DefaultKafkaClientSupplier
Importance: low

实现 org.apache.kafka.streams.KafkaClientSupplier 接口的客户端供应商类。

default.dsl.store

type: string
Default: rocksDB
Valid Values: [rocksDB, in_memory]

Importance: low

DSL 运算符使用的默认状态存储类型。

dsl.store.suppliers.class

Type: class
Default: org.apache.kafka.streams.state.BuiltInDslStoreSuppliers$RocksDBDslStoreSuppliers
Importance: low

定义将要插入的实施存储在 DSL 运算符中。必须实施 org.apache.kafka.streams.state.DslStoreSuppliers 接口。

enable.metrics.push

Type: boolean
Default: true
Importance: low

如果集群具有与客户端匹配的客户端指标，是否启用推送内部 [main-|restore-|global]consumer, producer, 和 admin 客户端指标。

metadata.max.age.ms

Type: long
Default: 300000 (5 minutes)
Valid Values: [0,…]
Importance: low

即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。

metric.reporters

Type: list
Default: ""
Importance: low

用作指标报告器的类列表。实施 org.apache.kafka.common.metrics.MetricsReporter 接口，允

许插入将收到新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。

metrics.num.samples

type: int
Default: 2
Valid Values: [1,…]
Importance: low

为计算指标维护的示例数量。

metrics.recording.level

Type: string
Default: INFO
Valid Values: [INFO, DEBUG, TRACE]
Importance: low

指标的最高记录级别。

metrics.sample.window.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,…]
Importance: low

计算指标示例的时间窗口。

poll.ms

Type: long
Default: 100
Importance: low

阻止等待输入的时间（以毫秒为单位）。

probing.rebalance.interval.ms

Type: long

Default: 600000 (10 minutes)
Valid Values: [60000,…]
Importance: low

触发重新平衡以探测到探测到回收完成并准备好激活的温副本前等待的时间（以毫秒为单位）。在分配平衡前，将继续触发重新平衡。必须至少为 1 分钟。

receive.buffer.bytes

type: int
Default: 32768 (32 kibibytes)
Valid Values: [-1,…]
Importance: low

读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。如果值为 -1，则使用操作系统默认值。

reconnect.backoff.max.ms

type: long
Default: 1000 (1 second)
Valid Values: [0,…]
Importance: low

重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。

reconnect.backoff.ms

type: long
Default: 50
Valid Values: [0,…]
Importance: low

尝试重新连接到给定主机前等待的基本时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到客户端到代理的所有连接尝试。这个值是初始 backoff 值，并为每个连续连接失败指数增加，最高为 reconnect.backoff.max.ms 值。

repartition.purge.interval.ms

type: long
Default: 30000 (30 秒)
Valid Values: [0,...]
Importance: low

从重新分区主题中删除完全消耗的记录的频率（以毫秒为单位）。清除将在最后一次清除以来至少在这个值后进行，但可能会延迟直到之后为止。（请注意，与 commit.interval.ms 不同，当 处理时，这个值的默认值保持不变。guarantee 被设置为 exactly_once_v2。

request.timeout.ms

type: int
Default: 40000 (40 seconds)
Valid Values: [0,...]
Importance: low

配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。

retries

type: int
Default: 0
Valid Values: [0,...,2147483647]
Importance: low

设置大于零的值将导致客户端重新发送失败并可能出现临时错误的请求。建议将值设为 0 或 MAX_VALUE，并使用对应的超时参数来控制客户端应重试请求的时长。

retry.backoff.ms

Type: long
Default: 100
Valid Values: [0,...]
Importance: low

尝试重试对给定主题分区失败的请求前等待的时间。这可避免在某些故障场景中的紧密循环中重复发送请求。这个值是初始 backoff 值，每个失败请求都会按指数增加，最多为 retry.backoff.ms 值。

rocksdb.config.setter

Type: class
Default: null
Importance: low

实现 org.apache.kafka.streams.state.RocksDBConfigSetter 接口的 Rocks DB 配置 setter 类或类名称。

send.buffer.bytes

type: int
Default: 131072 (128 kibibytes)
Valid Values: [-1,…]
Importance: low

发送数据时要使用的 TCP 发送缓冲区(SO_SNDBUF)的大小。如果值为 -1，则使用操作系统默认值。

state.cleanup.delay.ms

Type: long
Default: 600000 (10 minutes)
Importance: low

分区迁移时，在删除状态前等待的时间（以毫秒为单位）。只有尚未修改至少 状态的目录。cleanup.delay.ms 将被删除。

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, 3.3, 3.4, 3.5, 3.6]
Importance: low

允许以向后兼容的方式进行升级。从 [0.10.0.0, 1.1] 升级到 2.0+，或者从 [2.0, 2.3] 升级到 2.4+ 时需要。当从 3.3 升级到更新的版本时，不需要指定此配置。默认为 null。接受的值为 "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", "3.3", "3.5", "3.5", "3.5", "3.6", "3.6" 的值。

window.size.ms

Type: long
Default: null
Importance: low

为反序列化器设置窗口大小，以计算窗口结束时间。

windowed.inner.class.serde

Type: string
Default: null
Importance: low

窗口记录的内类的默认序列化器/反序列化器。必须实施 org.apache.kafka.common.serialization.Serde 接口。请注意，在 KafkaStreams 应用程序中设置此配置会导致错误，因为它只用于从 Plain consumer 客户端中使用。

windowstore.changelog.additional.retention.ms

type: long
Default: 86400000 (1 day)
Importance: low

向窗口添加 maintainMs，以确保不会预先从日志中删除数据。允许时钟偏移。默认值为 1 天。

附录 A. 使用您的订阅

Apache Kafka 的流通过软件订阅提供。要管理您的订阅，请访问红帽客户门户中的帐户。

访问您的帐户

1. 转至 access.redhat.com。
2. 如果您还没有帐户，请创建一个帐户。
3. 登录到您的帐户。

激活订阅

1. 转至 access.redhat.com。
2. 导航到 My Subscriptions。
3. 导航到 激活订阅 并输入您的 16 位激活号。

下载 Zip 和 Tar 文件

要访问 zip 或 tar 文件，请使用客户门户网站查找下载的相关文件。如果您使用 RPM 软件包，则不需要这一步。

1. 打开浏览器并登录红帽客户门户网站 产品下载页面，网址为 access.redhat.com/downloads。
2. 在 INTEGRATION AND AUTOMATION 目录中找到 Apache Kafka for Apache Kafka 的流。
3. 选择 Apache Kafka 产品所需的流。此时会打开 Software Downloads 页面。

4. 单击组件的 **Download** 链接。

使用 DNF 安装软件包

要安装软件包以及所有软件包的依赖软件包，请使用：

```
dnf install <package_name>
```

要从本地目录中安装之前下载的软件包，请使用：

```
dnf install <path_to_download_package>
```

更新于 2024-04-30