



Red Hat AMQ Streams 2.2

Kafka 設定プロパティ

設定プロパティを使用して Kafka コンポーネントを設定する

Red Hat AMQ Streams 2.2 Kafka 設定プロパティ

設定プロパティを使用して Kafka コンポーネントを設定する

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Kafka_configuration_properties.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Kafka 設定プロパティを使用して、Kafka コンポーネントがどのように動作するかを最大限に活用します。

目次

多様性を受け入れるオープンソースの強化	3
第1章 ブローカー設定プロパティ	4
第2章 トピック設定プロパティ	47
第3章 コンシューマー設定プロパティ	53
第4章 プロデューサー設定プロパティ	69
第5章 管理クライアントの設定プロパティ	85
第6章 KAFKA CONNECT 設定プロパティ	96
第7章 KAFKA ストリームの設定プロパティ	114
付録A サブスクリプションの使用	123
アカウントへのアクセス	123
サブスクリプションのアクティベート	123
Zip および Tar ファイルのダウンロード	123
DNF を使用したパッケージのインストール	123

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

第1章 ブローカー設定プロパティ

advertised.listeners

タイプ: string

デフォルト: null

重要度: 高

動的更新: ブローカーごと

クライアントが使用する ZooKeeper にパブリッシュするリスナー (**listeners** 設定プロパティとは異なる場合)。IaaS 環境では、ブローカーをバインドするインターフェイスとは異ならなくてはならない場合があります。これが設定されていない場合、**listeners** の値が使用されます。**listeners** とは異なり、0.0.0.0 メタアドレスをアドバタイズすることは有効ではありません。また、**listeners** とは異なり、このプロパティにはポートを重複させ、あるリスナーを別のリスナーのアドレスをアドバタイズするように設定することができます。これは、外部ロードバランサーが使用される場合に役に立ちます。

auto.create.topics.enable

タイプ: boolean

デフォルト: true

重要度: 高

動的更新: 読み取り専用

サーバーでトピックの自動作成を有効にします。

auto.leader.rebalance.enable

タイプ: boolean

デフォルト: true

重要度: 高

動的更新: 読み取り専用

自動リーダーバランシングを有効にします。バックグラウンドスレッドは、**leader.imbalance.check.interval.seconds** によって設定可能な一定間隔でパーティションリーダーの分散をチェックします。リーダーインバランスが **leader.imbalance.per.broker.percentage** を超える場合、パーティションの優先リーダーへのリーダーのリバランスがトリガーされます。

background.threads

Type: int

Default: 10

Valid Values: [1,...]

Importance: high

Dynamic update: cluster-wide

さまざまなバックグラウンド処理タスクに使用するスレッドの数。

broker.id

タイプ: int

デフォルト: -1

重要度: 高

動的更新: 読み取り専用

このサーバーのブローカー ID。未設定の場合は、一意のブローカー ID が生成されます。zookeeper が生成するブローカー ID とユーザーが設定したブローカー ID の間の競合を避けるために、生成されたブローカー ID は `reserved.broker.max.id + 1` から開始します。

compression.type

タイプ: string

デフォルト: プロデューサー

重要度: 高

動的更新: クラスター全体

特定のトピックの最終的な圧縮タイプを指定します。この設定は、標準の圧縮コーデック ('gzip'、'snappy'、'lz4'、'zstd') を受け入れます。さらに、圧縮なしに相当する 'uncompressed' と、プロデューサーによって設定された元の圧縮コーデックを維持する 'producer' も使用できます。

control.plane.listener.name

タイプ: string

デフォルト: null

重要度: 高

動的更新: 読み取り専用

コントローラーとブローカー間の通信に使用されるリスナーの名前。ブローカーは control.plane.listener.name を使用して、リスナーリスト内のエンドポイントを特定し、コントローラーからの接続をリッスンします。たとえば、ブローカーの設定が :listeners = INTERNAL://192.1.1.8:9092, EXTERNAL://10.1.1.5:9093, CONTROLLER://192.1.1.8:9094 listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSL control.plane.listener.name = CONTROLLER On startup の場合、ブローカーはセキュリティープロトコル "SSL" を使用して "192.1.1.8:9094" でリッスンを開始します。コントローラー側で、zookeeper を介してブローカーの公開済みエンドポイントを検出すると、control.plane.listener.name を使用してブローカーへの接続を確立するために使用するエンドポイントを見つけます。たとえば、zookeeper 上のブローカーの公開済みのエンドポイントが : "endpoints"

["INTERNAL://broker1.example.com:9092", "EXTERNAL://broker1.example.com:9093", "CONTROLLER で、コントローラーの設定が : listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSL control.plane.listener.name = CONTROLLER の場合、コントローラーはセキュリティープロトコル "SSL" で "broker1.example.com:9094" を使用してブローカーに接続します。明示的に設定されていない場合、デフォルト値は null で、コントローラー接続専用のエンドポイントはありません。

controller.listener.names

タイプ: string

デフォルト: null

重要度: 高

動的更新: 読み取り専用

コントローラーによって使用されるリスナーの名前のコンマ区切りリスト。これは、KRaft モードで実行する場合に必要になります。コントローラークォーラムと通信する場合、ブローカーは常にこのリストの最初のリスナーを使用します。注: ZK ベースのコントローラーは、この設定を設定しないでください。

controller.quorum.election.backoff.max.ms

Type: int

Default: 1000 (1 second)

Importance: high

Dynamic update: read-only

新しいエлекションを開始するまでの最大時間 (ミリ秒単位)。これは、グリッドロックされたエлекションを防ぐ際に役立つバイナリー指数バックオフメカニズムで使用されます。

controller.quorum.election.timeout.ms

Type: int

Default: 1000 (1 second)

Importance: high

Dynamic update: read-only

新しいエレクションをトリガーする前に、リーダーからフェッチできずに待機する最大時間 (ミリ秒単位)。

controller.quorum.fetch.timeout.ms

Type: int

Default: 2000 (2 seconds)

Importance: high

Dynamic update: read-only

候補者になり、投票者のエレクションをトリガーする前に、現在のリーダーからの正常なフェッチがない最大時間。リーダーの新しいエポックがあるかどうかを確認する前に、クォーラムの大部分からフェッチを受け取らない最大時間。

controller.quorum.voters

タイプ: list

デフォルト: ""

有効な値: non-empty list

重要度: 高

動的更新: 読み取り専用

{id}@{host}:{port} エントリーのコンマ区切りリストの投票者のセットの ID/エンドポイントの情報のマッピング。たとえば、**1@localhost:9092,2@localhost:9093,3@localhost:9094** になります。

delete.topic.enable

タイプ: boolean

デフォルト: true

重要度: 高

動的更新: 読み取り専用

トピックの削除を有効にします。この設定をオフの場合、管理ツールを介してトピックを削除しても効果はありません。

leader.imbalance.check.interval.seconds

型: long

デフォルト: 300

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

パーティションリバランスチェックがコントローラーによってトリガーされる頻度。

leader.imbalance.per.broker.percentage

型: int

デフォルト: 10

重要度: 高

動的更新: 読み取り専用

ブローカーごとに許可されるリーダーのインバランスの比率。コントローラーは、ブローカーごとにこの値を超えると、リーダーのバランスをトリガーします。この値はパーセンテージで指定されます。

listeners

Type: string

Default: PLAINTEXT://:9092

Importance: high

Dynamic update: per-broker

リスナーリスト - リッスンする URI とリスナー名のコンマ区切りリストリスナー名がセキュリ

ティープロトコルではない場合は、**listener.security.protocol.map** も設定する必要があります。リスナー名とポート番号は一意でなければなりません。ホスト名を 0.0.0.0 として指定し、すべてのインターフェイスにバインドします。ホスト名を空白のままにして、デフォルトのインターフェイスにバインドします。正当なリスナーリストの例: PLAINTEXT://myhost:9092,SSL://:9091 CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093.

log.dir

タイプ: string

デフォルト: /tmp/kafka-logs

重要度: 高

動的更新: 読み取り専用

ログデータが保持されるディレクトリー (log.dirs プロパティーで補足)。

log.dirs

タイプ: string

デフォルト: null

重要度: 高

動的更新: 読み取り専用

ログデータが保持されるディレクトリー。設定されていない場合は、log.dir の値が使用されます。

log.flush.interval.messages

Type: long

Default: 9223372036854775807

Valid Values: [1,...]

Importance: high

Dynamic update: cluster-wide

メッセージがディスクにフラッシュされる前に、ログパーティションで累積されるメッセージの数。

log.flush.interval.ms

タイプ: long

デフォルト: null

重要度: 高

動的更新: クラスター全体

ディスクにフラッシュされる前に、トピックのメッセージがメモリーに保持される最大時間 (ミリ秒単位)。設定されていない場合は、log.flush.scheduler.interval.ms の値が使用されます。

log.flush.offset.checkpoint.interval.ms

Type: int

Default: 60000 (1 minute)

Valid Values: [0,...]

Importance: high

Dynamic update: read-only

ログリカバリーポイントとして動作する最後のフラッシュの永続レコードを更新する頻度。

log.flush.scheduler.interval.ms

型: long

デフォルト: 9223372036854775807

重要度: 高

動的更新: 読み取り専用

ログフラッシャーが、ログをディスクにフラッシュする必要があるかどうかをチェックする頻度 (ミリ秒単位)。

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

Type: int
Default: 60000 (1 minute)
Valid Values: [0,...]
Importance: high
Dynamic update: read-only
ログ開始オフセットの永続レコードを更新する頻度。

log.retention.bytes

タイプ: long
デフォルト: -1
重要度: 高
動的更新: クラスター全体
ログを削除する前のログの最大サイズ。

log.retention.hours

型: int
デフォルト: 168
重要度: 高
動的更新: 読み取り専用
ログファイルを削除する前にログファイルを保持する時間 (時間単位) (log.retention.ms プロパティに対してターシャリー)。

log.retention.minutes

タイプ: int
デフォルト: null
重要度: 高
動的更新: 読み取り専用
ログファイルを削除する前にログファイルを保持する分数 (分単位) (log.retention.ms プロパティに対してセカンダリー)。設定されていない場合は、log.retention.hours の値が使用されます。

log.retention.ms

タイプ: long
デフォルト: null
重要度: 高
動的更新: クラスター全体
ログファイルを削除する前にログファイルを保持する秒数 (ミリ秒単位)。設定されていない場合は、log.retention.minutes の値が使用されます。-1 に設定すると、時間制限は適用されません。

log.roll.hours

型: int
デフォルト: 168
有効な値: [1,...]
重要度: 高
動的更新: 読み取り専用
新しいログセグメントがロールアウトされるまでの最大時間 (時間単位) (log.roll.ms プロパティに対してセカンダリー)

log.roll.jitter.hours

型: int
デフォルト: 0

有効な値: [0,...]

重要度: 高

動的更新: 読み取り専用

logRollTimeMillis から差し引く最大ジッター (時間単位) (log.roll.jitter.ms プロパティに対してセカンダリー)

log.roll.jitter.ms

タイプ: long

デフォルト: null

重要度: 高

動的更新: クラスター全体

logRollTimeMillis から減算する最大ジッター (ミリ秒単位)。設定されていない場合は、log.roll.jitter.hours の値が使用されます。

log.roll.ms

タイプ: long

デフォルト: null

重要度: 高

動的更新: クラスター全体

新しいログセグメントがロールアウトされるまでの最大時間 (ミリ秒単位)。設定されていない場合は、log.roll.hours の値が使用されます。

log.segment.bytes

型: int

デフォルト: 1073741824 (1 ギビバイト)

有効な値: [14,...]

重要度: 高

動的更新: クラスター全体

1つのログファイルの最大サイズ。

log.segment.delete.delay.ms

型: long

デフォルト: 60000 (1分)

有効な値: [0,...]

重要度: 高

動的更新: クラスター全体

ファイルシステムからファイルを削除するまでの待機時間。

message.max.bytes

型: int

デフォルト: 1048588

有効な値: [0,...]

重要度: 高

動的更新: クラスター全体

Kafka によって許可される最大のレコードバッチサイズ (圧縮が有効な場合は圧縮後)。これが増加し、0.10.2 より古いコンシューマーが存在する場合、コンシューマーのフェッチサイズも増加して、このような大きなレコードバッチをフェッチできるようにする必要があります。最新のメッセージ形式バージョンでは、効率化のためにレコードは常にバッチにグループ化されます。以前のメッセージ形式のバージョンでは、圧縮されていないレコードはバッチにグループ化されず、この制限はその場合は1つのレコードにのみ適用されます。これは、トピックレベルの **max.message.bytes** 設定でトピックごとに設定できます。

metadata.log.dir

タイプ: string

デフォルト: null

重要度: 高

動的更新: 読み取り専用

この設定は、クラスターのメタデータログを KRaft モードで配置する場所を決めます。設定されていない場合、メタデータログは log.dirs の最初のログディレクトリーに配置されます。

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

型: long

デフォルト: 20971520

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

これは、最新のスナップショットと新しいスナップショットを生成する前に必要な最高レベルのウォーターマークとの間のログの最大バイト数です。

metadata.log.segment.bytes

型: int

デフォルト: 1073741824 (1 ギビバイト)

有効な値: [12,...]

重要度: 高

動的更新: 読み取り専用

単一のメタデータログファイルの最大サイズ。

metadata.log.segment.ms

型: long

デフォルト: 604800000 (7 日)

重要度: 高

動的更新: 読み取り専用

新しいメタデータログファイルがロールアウトされるまでの最大時間 (ミリ秒単位)。

metadata.max.retention.bytes

タイプ: long

デフォルト: -1

重要度: 高

動的更新: 読み取り専用

古いスナップショットおよびログファイルを削除する前に、メタデータログおよびスナップショットの最大合計サイズ。ログが削除される前に少なくとも1つのスナップショットが存在する必要があるため、これはソフトリミットです。

metadata.max.retention.ms

型: long

デフォルト: 604800000 (7 日)

重要度: 高

動的更新: 読み取り専用

削除する前にメタデータログファイルまたはスナップショットを保持するミリ秒数。ログが削除される前に少なくとも1つのスナップショットが存在する必要があるため、これはソフトリミットです。

min.insync.replicas

型: int

デフォルト: 1

有効な値: [1,...]

重要度: 高

動的更新: クラスター全体

プロデューサーが acks を "all" (または "-1") に設定すると、min.insync.replicas は、書き込みが正常に実行されるように書き込みを承認しなければならないレプリカの最小数を指定します。この最小値が満たされない場合、プロデューサーは例外 (NotEnoughReplicas または NotEnoughReplicasAfterAppend のいずれか) を発生させます。min.insync.replicas と acks を併用することで、より高い持続性が保証されます。一般的なシナリオでは、レプリケーション係数 3 のトピックを作成し、min.insync.replicas を 2 に設定し、acks を all にして生成します。これにより、レプリカの大部分が書き込みを受信しない場合に、プロデューサーは確実に例外を発生させます。

node.id

タイプ: int

デフォルト: -1

重要度: 高

動的更新: 読み取り専用

ロールに関連付けられたノード ID です。このプロセスは、**process.roles** が空ではない場合に再生します。これは、KRaft モードで実行する場合に必須の設定です。

num.io.threads

型: int

デフォルト: 8

有効な値: [1,...]

重要度: 高

動的更新: クラスター全体

サーバーが要求の処理に使用するスレッドの数。これにはディスク I/O が含まれる場合があります。

num.network.threads

型: int

デフォルト: 3

有効な値: [1,...]

重要度: 高

動的更新: クラスター全体

サーバーがネットワークから要求を受信し、応答をネットワークに送信するために使用するスレッドの数。

num.recovery.threads.per.data.dir

型: int

デフォルト: 1

有効な値: [1,...]

重要度: 高

動的更新: クラスター全体

起動時のログリカバリーおよびシャットダウン時にフラッシュするために使用されるデータディレクトリーごとのスレッド数。

num.replica.alter.log.dirs.threads

タイプ: int

デフォルト: null

重要度: 高

動的更新: 読み取り専用

ログディレクトリー間でレプリカを移動できるスレッドの数。これにはディスク I/O が含まれる場合があります。

num.replica.fetchers

型: int

デフォルト: 1

重要度: 高

動的更新: クラスターワイド

ソースブローカーからメッセージを複製するために使用されるフェッチャースレッドの数。この値を増やすと、フォロワーブローカーの I/O 並列処理のレベルを増やすことができます。

offset.metadata.max.bytes

型: int

デフォルト: 4096 (4 ギビバイト)

重要度: 高

動的更新: 読み取り専用

オフセットコミットに関連付けられたメタデータエントリーの最大サイズ。

offsets.commit.required.acks

タイプ: short

デフォルト: -1

重要度: 高

動的更新: 読み取り専用

コミットを許可する前に必要な acks。通常、デフォルトの (-1) は上書きできません。

offsets.commit.timeout.ms

型: int

デフォルト: 5000 (5 秒)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

オフセットコミットは、オフセットトピックのすべてのレプリカがコミットを受信するか、またはこのタイムアウトに達するまで遅延します。これは、プロデューサーリクエストのタイムアウトと似ています。

offsets.load.buffer.size

型: int

デフォルト: 5242880

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

オフセットをキャッシュに読み込むときにオフセットセグメントから読み取るバッチサイズ (ソフトリミット。レコードが大きすぎるとオーバーライドされます)。

offsets.retention.check.interval.ms

型: long

デフォルト: 600000 (10 分)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

古いオフセットをチェックする頻度。

offsets.retention.minutes

型: int

デフォルト: 10080

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

コンシューマーグループがすべてのコンシューマーを失った後 (つまり空になった場合)、そのオフセットは破棄される前にこの保持期間は保持されます。(手動割り当てを使用する) スタンドアロンコンシューマーの場合、オフセットは最後のコミットの期間と、この保持期間後に期限切れになります。

offsets.topic.compression.codec

型: int

デフォルト: 0

重要度: 高

動的更新: 読み取り専用

オフセットトピックの圧縮コーデック - 圧縮は "atomic" コミットの達成に使用できます。

offsets.topic.num.partitions

型: int

デフォルト: 50

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

オフセットコミットトピックのパーティション数 (デプロイメント後には変更しないでください)。

offsets.topic.replication.factor

型: short

デフォルト: 3

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

オフセットトピックのレプリケーション係数 (可用性を確保するために高く設定します)。内部トピックの作成は、クラスタのサイズがこのレプリケーション係数の要件を満たすまで失敗します。

offsets.topic.segment.bytes

型: int

デフォルト: 104857600 (100 メビバイト)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

ログコンパクションとキャッシュの負荷をより高速にするために、オフセットトピックセグメントバイトは比較的小さく維持する必要があります。

process.roles

タイプ: list

デフォルト: ""

有効な値: [broker, controller]

重要度: 高

動的更新: 読み取り専用

このプロセスが果たすロールは、'broker'、'controller'、または 'broker,controller' (両方の場合)。この設定は、(Zoo Keeper ではなく) KRaft (Kafka Raft) モードのクラスターにのみ適用できます。Zookeeper クラスターの場合、この設定を未定義のままにするか、空のままにします。

queued.max.requests

型: int

デフォルト: 500

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

ネットワークスレッドをブロックする前に、データプレーンで許可されるキューに置かれた要求の数。

replica.fetch.min.bytes

型: int

デフォルト: 1

重要度: 高

動的更新: 読み取り専用

各フェッチ応答で想定される最小バイト数。十分なバイトがない場合は、**replica.fetch.wait.max.ms** (broker config) まで待機します。

replica.fetch.wait.max.ms

型: int

デフォルト: 500

重要度: 高

動的更新: 読み取り専用

フォロワーレプリカが発行する各フェッチャーリクエストの最大待機時間。低スループットトピックのISRが頻繁に縮小するのを防ぐために、この値は常に **replica.lag.time.max.ms** よりも小さくする必要があります。

replica.high.watermark.checkpoint.interval.ms

型: long

デフォルト: 5000 (5 秒)

重要度: 高

動的更新: 読み取り専用

最高レベルのウォーターマークがディスクに保存される頻度。

replica.lag.time.max.ms

型: long

デフォルト: 30000 (30 秒)

重要度: 高

動的更新: 読み取り専用

フォロワーがフェッチリクエストを送信していない場合や、リーダーのログ終了オフセットまで少なくともこの時間消費していない場合、リーダーはフォロワーを **isr** から削除します。

replica.socket.receive.buffer.bytes

型: int

デフォルト: 65536 (64 キビバイト)

重要度: 高

動的更新: 読み取り専用

ネットワーク要求のソケット受信バッファ。

replica.socket.timeout.ms

型: int

デフォルト: 30000 (30 秒)

重要度: 高

動的更新: 読み取り専用

ネットワーク要求のソケットタイムアウト。その値は、少なくとも replica.fetch.wait.max.ms である必要があります。

request.timeout.ms

型: int

デフォルト: 30000 (30 秒)

重要度: 高

動的更新: 読み取り専用

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

sasl.mechanism.controller.protocol

型: string

デフォルト: GSSAPI

重要度: 高

動的更新: 読み取り専用

コントローラーとの通信に使用される SASL メカニズム。デフォルトは GSSAPI です。

socket.receive.buffer.bytes

型: int

デフォルト: 102400 (100 キビバイト)

重要度: 高

動的更新: 読み取り専用

ソケットサーバーソケットの SO_RCVBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。

socket.request.max.bytes

型: int

デフォルト: 104857600 (100 メビバイト)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

ソケットリクエストの最大バイト数。

socket.send.buffer.bytes

型: int

デフォルト: 102400 (100 キビバイト)

重要度: 高

動的更新: 読み取り専用

ソケットサーバーソケットの SO_SNDBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。

transaction.max.timeout.ms

型: int

デフォルト: 900000 (15 分)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

トランザクションの最大許容タイムアウト。クライアントの要求されたトランザクション時間がこの値を超えると、ブローカーは `InitProducerIdRequest` でエラーを返します。これにより、クライアントがタイムアウトを大きくしすぎて、トランザクションに含まれるトピックからの読み取りをコンシューマーが停止する可能性がなくなります。

transaction.state.log.load.buffer.size

型: int

デフォルト: 5242880

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

プロデューサー ID とトランザクションをキャッシュにロードするときにトランザクションログセグメントから読み取るバッチサイズ (ソフトリミット。レコードが大きすぎると上書きされます)。

transaction.state.log.min.isr

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

トランザクショントピックの上書きされた `min.insync.replicas` 設定。

transaction.state.log.num.partitions

型: int

デフォルト: 50

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

トランザクショントピックのパーティション数 (デプロイメント後には変更しないでください)。

transaction.state.log.replication.factor

型: short

デフォルト: 3

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

トランザクショントピックのレプリケーション係数 (可用性を確保するために高く設定します)。内部トピックの作成は、クラスターのサイズがこのレプリケーション係数の要件を満たすまで失敗します。

transaction.state.log.segment.bytes

型: int

デフォルト: 104857600 (100 メビバイト)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

ログコンパクションとキャッシュの負荷をより高速にするために、トランザクショントピックセグメントバイトは比較的小さく維持する必要があります。

transactional.id.expiration.ms

型: int

デフォルト: 604800000 (7日)

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

トランザクションコーディネーターが、トランザクション ID の有効期限が切れる前に、現在のトランザクションのトランザクションステータスの更新を受信せずに待機する時間 (ミリ秒単位)。この設定は、プロデューサー ID の有効期限にも影響します。プロデューサー ID は、指定されたプロデューサー ID で最後に書き込んだ後、この時間が経過すると期限切れになります。トピックの保持設定が原因で、プロデューサー ID からの最後の書き込みが削除された場合、プロデューサー ID の有効期限が早くなる可能性があることに注意してください。

unclean.leader.election.enable

タイプ: boolean

デフォルト: false

重要度: 高

動的更新: クラスター全体

データが失われる可能性がある場合でも、ISR セットにないレプリカを最後の手段としてリーダーとして選出できるようにするかどうかを示します。

zookeeper.connect

タイプ: string

デフォルト: null

重要度: 高

動的更新: 読み取り専用

ZooKeeper 接続文字列を **hostname:port** の形式で指定します。ここで、ホストとポートは ZooKeeper サーバーのホストおよびポートになります。ZooKeeper マシンが停止しているときに他の ZooKeeper ノード経由で接続できるようにするには、**hostname1:port1,hostname2:port2,hostname3:port3** の形式で複数のホストを指定することもできます。また、サーバーには ZooKeeper 接続文字列の一部として ZooKeeper の chroot パスを含めることもできます。この文字列は、データをグローバル ZooKeeper namespace の一部のパスに配置します。たとえば、**/chroot/path** の chroot パスを指定するには、接続文字列に **hostname1:port1,hostname2:port2,hostname3:port3/chroot/path** を指定します。

zookeeper.connection.timeout.ms

タイプ: int

デフォルト: null

重要度: 高

動的更新: 読み取り専用

クライアントが zookeeper への接続を確立するまで待機する最大時間。設定されていない場合は、zookeeper.session.timeout.ms の値が使用されます。

zookeeper.max.in.flight.requests

型: int

デフォルト: 10

有効な値: [1,...]

重要度: 高

動的更新: 読み取り専用

ブロックする前にクライアントが Zookeeper に送信する確認されていないリクエストの最大数。

zookeeper.session.timeout.ms

型: int
デフォルト: 18000 (18 秒)
重要度: 高
動的更新: 読み取り専用
ZooKeeper セッションのタイムアウト。

zookeeper.set.acl

タイプ: boolean
デフォルト: false
重要度: 高
動的更新: 読み取り専用
セキュアな ACL を使用するようにクライアントを設定します。

broker.heartbeat.interval.ms

型: int
デフォルト: 2000 (2 秒)
重要度: 中
動的更新: 読み取り専用
ブローカーのハートビート間の時間の長さ (ミリ秒単位)。KRaft モードで実行する場合に使用されま
す。

broker.id.generation.enable

タイプ: boolean
デフォルト: true
重要度: 中
動的更新: 読み取り専用
サーバーでブローカー ID の自動生成を有効にします。これを有効にすると、reserved.broker.max.id
に設定された値を確認する必要があります。

broker.rack

タイプ: string
デフォルト: null
重要度: 中
動的更新: 読み取り専用
ブローカーのラック。これは、フォールトトレランスのためのラック対応レプリケーション割り当
てで使用されます。たとえば、**RACK1**、**us-east-1d** です。

broker.session.timeout.ms

型: int
デフォルト: 9000 (9 秒)
重要度: 中
動的更新: 読み取り専用
ハートビートが作成されない場合にブローカーのリースが続く時間の長さ (ミリ秒単位)。KRaft モー
ドで実行する場合に使用されます。

connections.max.idle.ms

型: long
デフォルト: 600000 (10 分)
重要度: 中
動的更新: 読み取り専用
アイドル接続のタイムアウト: サーバーソケットプロセッサスレッドは、これ以上にアイドル状態
にある接続を閉じます。

connections.max.reauth.ms

型: long

デフォルト: 0

重要度: 高

動的更新: 読み取り専用

正数 (デフォルトは正数ではなく 0) に明示的に設定されている場合、設定された値を超えないセッションライフタイムは、認証時に v2.2.0 以降のクライアントに通知されます。ブローカーは、セッションの有効期間内に再認証されなかったあらゆる接続を切断し、その後、これは再認証以外の目的で使用されます。設定名には、任意でリスナー接頭辞と SASL メカニズム名を小文字で接頭辞できます。たとえば、`listener.name.sasl_ssl.oauthbearer.connections.max.reauth.ms=3600000` などです。

controlled.shutdown.enable

タイプ: boolean

デフォルト: true

重要度: 中

動的更新: 読み取り専用

サーバーの制御されたシャットダウンを有効にします。

controlled.shutdown.max.retries

型: int

デフォルト: 3

重要度: 中

動的更新: 読み取り専用

制御されたシャットダウンは、複数の理由で失敗する可能性があります。これにより、このような障害が発生した場合の再試行回数が決まります。

controlled.shutdown.retry.backoff.ms

型: long

デフォルト: 5000 (5 秒)

重要度: 中

動的更新: 読み取り専用

それぞれの再試行の前に、システムは以前の障害を引き起こした状態 (コントローラーのフェイルオーバー、レプリカの遅延など) から回復する時間が必要です。この設定は、再試行までの待機時間を決定します。

controller.quorum.append.linger.ms

型: int

デフォルト: 25

重要度: 中

動的更新: 読み取り専用

書き込みをディスクにフラッシュする前に、それらの書き込みが蓄積するのをリーダーが待機する期間 (ミリ秒単位)。

controller.quorum.request.timeout.ms

型: int

デフォルト: 2000 (2 秒)

重要度: 中

動的更新: 読み取り専用

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

controller.socket.timeout.ms

型: int
デフォルト: 30000 (30 秒)
重要度: 中
動的更新: 読み取り専用
controller-to-broker チャンネルのソケットタイムアウト。

default.replication.factor

型: int
デフォルト: 1
重要度: 中
動的更新: 読み取り専用
自動的に作成されるトピックのデフォルトレプリケーション係数。

delegation.token.expiry.time.ms

型: long
デフォルト: 86400000 (1 日)
有効な値: [1,...]
重要度: 中
動的更新: 読み取り専用
トークンの更新が必要となるまでのトークンの有効時間 (ミリ秒単位)。デフォルト値は 1 日です。

delegation.token.master.key

タイプ: password
デフォルト: null
重要度: 中
動的更新: 読み取り専用
非推奨: この設定の代わりに使用する必要のある delegation.token.secret.key のエイリアス。

delegation.token.max.lifetime.ms

型: long
デフォルト: 604800000 (7 日)
有効な値: [1,...]
重要度: 中
動的更新: 読み取り専用
トークンの最大有効期間は、それを超えると更新できなくなります。デフォルト値は 7 日です。

delegation.token.secret.key

タイプ: password
デフォルト: null
重要度: 中
動的更新: 読み取り専用
委任トークンを生成および検証するための秘密鍵。すべてのブローカーで同じキーを設定する必要があります。キーが設定されていないか、空の文字列に設定されていると、ブローカーは委任トークンのサポートを無効にします。

delete.records.purgatory.purge.interval.requests

型: int
デフォルト: 1
重要度: 中
動的更新: 読み取り専用

delete records request purgatory のページ間隔 (リクエストの回数)。

fetch.max.bytes

型: int

デフォルト: 57671680 (55 メビバイト)

有効な値: [1024,...]

重要度: 中

動的更新: 読み取り専用

フェッチリクエストに対して返す最大バイト数。1024 以上でなければなりません。

fetch.purgatory.purge.interval.requests

型: int

デフォルト: 1000

重要度: 中

動的更新: 読み取り専用

fetch request purgatory のページ間隔 (リクエストの回数)。

group.initial.rebalance.delay.ms

型: int

デフォルト: 3000 (3 秒)

重要度: 中

動的更新: 読み取り専用

グループコーディネーターが最初のリバランスを実行する前に、より多くのコンシューマーが新しいグループに参加するのを待つ時間。遅延が長くなると、リバランスが少なくなる可能性があります。処理が開始されるまでの時間が長くなります。

group.max.session.timeout.ms

型: int

デフォルト: 1800000 (30 分)

重要度: 中

動的更新: 読み取り専用

登録されたコンシューマーに許可される最大セッションタイムアウト。タイムアウトが長くなると、コンシューマーはハートビートの間にメッセージを処理する時間が長くなりますが、障害の検出に時間がかかります。

group.max.size

型: int

デフォルト: 2147483647

有効な値: [1,...]

重要度: 中

動的更新: 読み取り専用

1つのコンシューマーグループが対応できるコンシューマーの最大数。

group.min.session.timeout.ms

型: int

デフォルト: 6000 (6 秒)

重要度: 中

動的更新: 読み取り専用

登録されたコンシューマーに許可される最小セッションタイムアウト。タイムアウトを短くすると、障害の検出が速くなりますが、コンシューマーのハートビートが頻繁に発生し、ブローカーのリソースが対応できなくなる可能性があります。

initial.broker.registration.timeout.ms

型: int

デフォルト: 60000 (1分)

重要度: 中

動的更新: 読み取り専用

コントローラークォーラムで最初に登録する場合、失敗を宣言してブローカープロセスを終了するまで待機する時間 (ミリ秒単位)。

inter.broker.listener.name

タイプ: string

デフォルト: null

重要度: 中

動的更新: 読み取り専用

ブローカー間の通信に使用されるリスナーの名前。これが設定されていない場合、リスナー名は security.inter.broker.protocol によって定義されます。これと security.inter.broker.protocol プロパティを同時に設定するとエラーになります。

inter.broker.protocol.version

タイプ: 文字列

デフォルト: 3.2-IV0

Valid Values: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0]

重要度: 中

動的更新: 読み取り専用

使用する inter-broker プロトコルのバージョンを指定します。通常、これはすべてのブローカーが新しいバージョンにアップグレードされた後に表示されます。いくつかの有効な値の例は、0.8.0、0.8.1、0.8.1.1、0.8.2、0.8.2.0、0.8.2.1、0.9.0.0、0.9.0.1 です。完全なリストについては ApiVersion を確認してください。

log.cleaner.backoff.ms

型: long

デフォルト: 15000 (15 秒)

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

クリーニングするログがない場合のスリープ時間。

log.cleaner.dedupe.buffer.size

型: long

デフォルト: 134217728

重要度: 中

動的更新: クラスター全体

すべてのクリーナースレッドにおけるログの重複排除に使用されるメモリーの合計。

log.cleaner.delete.retention.ms

型: long

デフォルト: 86400000 (1日)

重要度: 中

動的更新: クラスター全体

ログ圧縮トピックの tombstone 削除マーカを保持する期間。また、この設定は、最終ステージの

有効なスナップショットを確実に取得するために、コンシューマーがオフセット 0 から開始する場合に読み取りを完了しなければならない時間にも制限を与えます (そうでない場合、スキャンを完了する前に、削除 tombstone が収集される可能性があります)。

log.cleaner.enable

タイプ: boolean

デフォルト: true

重要度: 中

動的更新: 読み取り専用

ログクリーナープロセスを有効にして、サーバー上で実行できるようにします。内部オフセットトピックを含む cleanup.policy=compact でトピックを使用する場合は、有効にする必要があります。無効にすると、これらのトピックは圧縮されず、サイズが継続的に大きくなります。

log.cleaner.io.buffer.load.factor

型: double

デフォルト: 0.9

重要度: 中

動的更新: クラスター全体

ログクリーナー重複排除バッファの負荷率。重複排除バッファの最大パーセンテージ。値を大きくすると、一度により多くのログをクリーンアップできますが、ハッシュ衝突が多くなります。

log.cleaner.io.buffer.size

型: int

デフォルト: 524288

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

すべてのクリーナースレッドにまたがってログクリーナー I/O バッファに使用されるメモリーの合計。

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

型: double

デフォルト: 1.7976931348623157E308

重要度: 中

動的更新: クラスター全体

ログクリーナーは、読み取りと書き込みの I/O の合計が平均してこの値よりも小さくなるようにロットリングされます。

log.cleaner.max.compaction.lag.ms

型: long

デフォルト: 9223372036854775807

重要度: 中

動的更新: クラスター全体

メッセージがログのコンパクションの対象外のままになる最大時間。圧縮されるログにのみ適用されます。

log.cleaner.min.cleanable.ratio

型: double

デフォルト: 0.5

重要度: 中

動的更新: クラスター全体

クリーニングの対象となるログの、合計ログに対するダーティログの最小比率。

log.cleaner.max.compaction.lag.ms または log.cleaner.min.compaction.lag.ms 設定も指定されている場合、ログコンパクターは次のいずれかの場合にすぐにログをコンパクションの対象と見なします。(i) ダーティー比率のしきい値に達し、ログに少なくとも log.cleaner.min.compaction.lag.ms 期間のダーティー (圧縮されていない) レコードがある場合、または (ii) ログに最大で log.cleaner.max.compaction.lag.ms 期間のダーティー (圧縮されていない) レコードがある場合。

log.cleaner.min.compaction.lag.ms

型: long

デフォルト: 0

重要度: 中

動的更新: クラスター全体

メッセージがログで圧縮されないままになる最小時間。圧縮されるログにのみ適用されます。

log.cleaner.threads

型: int

デフォルト: 1

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

ログのクリーニングに使用するバックグラウンドスレッドの数。

log.cleanup.policy

タイプ: list

デフォルト: delete

有効な値: [compact, delete]

重要度: 中

動的更新: クラスター全体

保持期間を超えたセグメントのデフォルトのクリーンアップポリシー。有効なポリシーのコンマ区切りリスト。有効なポリシーは "delete" と "compact" です。

log.index.interval.bytes

型: int

デフォルト: 4096 (4 キビバイト)

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

オフセットインデックスにエントリーを追加する間隔。

log.index.size.max.bytes

型: int

デフォルト: 10485760 (10 メビバイト)

有効な値: [4,...]

重要度: 中

動的更新: クラスター全体

オフセットインデックスの最大サイズ (バイト単位)。

log.message.format.version

型: string

デフォルト: 3.0-IV1

有効な値: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1,

3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0]

重要度: 中

動的更新: 読み取り専用

ブローカーがログにメッセージを追加するために使用するメッセージ形式バージョンを指定します。この値は有効な ApiVersion である必要があります。たとえば、0.8.2、0.9.0.0、0.10.0 です。詳細は ApiVersion を確認してください。特定のメッセージ形式のバージョンを設定することで、ユーザーは、ディスク上の既存のメッセージすべてが指定したバージョンよりも小さいか、または等しいことを認定します。この値を誤って設定すると、以前のバージョンを持つコンシューマーが、認識されない形式でメッセージを受信するため、破損します。

log.message.timestamp.difference.max.ms

型: long

デフォルト: 9223372036854775807

重要度: 中

動的更新: クラスター全体

ブローカーがメッセージを受信したときのタイムスタンプと、メッセージに指定されたタイムスタンプとの間の最大差。log.message.timestamp.type=CreateTime の場合、タイムスタンプの差がこのしきい値を超えると、メッセージが拒否されます。この設定は、log.message.timestamp.type=LogAppendTime の場合に無視されます。不必要に頻繁なログのローリングを回避するために、許可される最大タイムスタンプの差は log.retention.ms 以下にする必要があります。

log.message.timestamp.type

型: string

デフォルト: CreateTime

有効な値: [CreateTime, LogAppendTime]

重要度: 中

動的更新: クラスター全体

メッセージのタイムスタンプが、メッセージ作成時間かログの追加時間であるかを定義します。値は **CreateTime** または **LogAppendTime** である必要があります。

log.preallocate

タイプ: boolean

デフォルト: false

重要度: 中

動的更新: クラスター全体

新規セグメントの作成時にファイルを事前に割り当てる必要がありますか? Windows で Kafka を使用している場合は、true に設定する必要があります。

log.retention.check.interval.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [1,...]

重要度: 中

動的更新: 読み取り専用

ログクリーナーが、削除の対象となるログがあるかどうかを確認する頻度 (ミリ秒単位)。

max.connection.creation.rate

型: int

デフォルト: 2147483647

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

ブローカーでいつでも許可する最大接続作成率。リスナーレベルの制限は、設定名の前にリスナー接頭辞を付けて設定することもできます (例:

listener.name.internal.max.connection.creation.rate)。ブローカー全体の接続レート制限は、ブローカーの容量に基づいて設定する必要がありますが、リスナーの制限はアプリケーションの要件に基づいて設定する必要があります。inter-broker リスナーを除き、リスナーまたはブローカーの制限のいずれかに達すると、新しい接続はスロットリングされます。inter-broker リスナーの接続は、リスナーレベルのレート制限に達した場合にのみスロットリングされます。

max.connections

型: int

デフォルト: 2147483647

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

ブローカーでいつでも許可する最大接続数。この制限は、`max.connections.per.ip` を使用して設定された各 IP の制限に加えて適用されます。リスナーレベルの制限は、設定名の前にリスナー接頭辞を付けて設定することもできます (例: **listener.name.internal.max.connections**)。ブローカー全体の制限はブローカー容量に基づいて設定する必要がありますが、リスナー制限はアプリケーション要件に基づいて設定する必要があります。リスナーまたはブローカーの制限のいずれかに達すると、新しい接続はブロックされます。inter-broker リスナーの接続は、ブローカー全体の制限に到達しても許可されます。この場合、別のリスナーで一番最近使用されていない接続が閉じられます。

max.connections.per.ip

型: int

デフォルト: 2147483647

有効な値: [0,...]

重要度: 中

動的更新: クラスター全体

各 IP アドレスから許容される接続の最大数。これは、`max.connections.per.ip.overrides` プロパティを使用して上書きが設定されている場合は 0 に設定できます。制限に達すると、IP アドレスからの新規接続はドロップされます。

max.connections.per.ip.overrides

タイプ: string

デフォルト: ""

重要度: 中

動的更新: クラスター全体

ip ごとまたはホスト名のコンマ区切りリストは、デフォルトの最大接続数よりも優先されます。値の例は "hostName:100,127.0.0.1:200" です。

max.incremental.fetch.session.cache.slots

型: int

デフォルト: 1000

有効な値: [0,...]

重要度: 中

動的更新: 読み取り専用

維持する増分フェッチセッションの最大数。

num.partitions

型: int

デフォルト: 1

有効な値: [1,...]

重要度: 中

動的更新: 読み取り専用

トピックごとのログパーティションのデフォルト数。

password.encoder.old.secret

タイプ: password

デフォルト: null

重要度: 中

動的更新: 読み取り専用

動的に設定されたパスワードをエンコードするために使用されていた古いシークレット。これは、シークレットが更新される場合にのみ必要です。指定されている場合、動的にエンコードされたパスワードはすべて、この古いシークレットを使用してデコードされ、ブローカーの起動時に password.encoder.secret を使用して再エンコードされます。

password.encoder.secret

タイプ: password

デフォルト: null

重要度: 中

動的更新: 読み取り専用

このブローカーに動的に設定されたパスワードをエンコードするために使用されるシークレット。

principal.builder.class

タイプ: class

デフォルト: org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalBuilder

重要度: 中

動的更新: ブローカーごと

承認中に使用される KafkaPrincipal オブジェクトを構築するために使用される KafkaPrincipalBuilder インターフェイスを実装するクラスの完全修飾名。プリンシパルビルダーが定義されていない場合、デフォルトの動作は使用中のセキュリティープロトコルによって異なります。SSL 認証の場合、プリンシパルは、クライアント証明書 (提供されている場合) からの識別名に適用される **ssl.principal.mapping.rules** によって定義されたルールを使用して派生されます。それ以外の場合は、クライアント認証が不要な場合はプリンシパル名は ANONYMOUS になります。SASL 認証の場合、プリンシパルは、GSSAPI が使用されている場合は **sasl.kerberos.principal.to.local.rules** で定義されるルールを使用して、他のメカニズムを使用している場合は SASL 認証 ID を使用して派生されます。PLAINTEXT の場合、プリンシパルは ANONYMOUS になります。

producer.purgatory.purge.interval.requests

型: int

デフォルト: 1000

重要度: 中

動的更新: 読み取り専用

producer request purgatory のページ間隔 (リクエストの回数)。

queued.max.request.bytes

タイプ: long

デフォルト: -1

重要度: 中

動的更新: 読み取り専用

これ以上要求が読み取られなくなるまでに許可されるキューに入れられたバイト数。

replica.fetch.backoff.ms

型: int

デフォルト: 1000 (1 秒)
有効な値: [0,...]
重要度: 中
動的更新: 読み取り専用
パーティションのフェッチエラーが発生したときにスリープする時間。

replica.fetch.max.bytes

型: int
デフォルト: 1048576 (1 メビバイト)
有効な値: [0,...]
重要度: 中
動的更新: 読み取り専用
パーティションごとにフェッチを試みるメッセージのバイト数。これは絶対的な最大値ではありません。フェッチの最初の空ではないパーティションの最初のレコードバッチがこの値よりも大きい場合でも、確実に前進することができるようにレコードバッチが返されます。ブローカーによって許可される最大レコードバッチサイズは、**message.max.bytes** (ブローカー設定) または **max.message.bytes** (トピック設定) で定義されます。

replica.fetch.response.max.bytes

型: int
デフォルト: 10485760 (10 メビバイト)
有効な値: [0,...]
重要度: 中
動的更新: 読み取り専用
フェッチ応答全体に対して想定される最大バイト数。レコードはバッチで取得され、フェッチの最初の空ではないパーティションの最初のレコードバッチがこの値よりも大きい場合でも、確実に前進することができるようにレコードバッチが返されます。そのため、これは絶対的な最大値ではありません。ブローカーによって許可される最大レコードバッチサイズは、**message.max.bytes** (ブローカー設定) または **max.message.bytes** (トピック設定) で定義されます。

replica.selector.class

タイプ: string
デフォルト: null
重要度: 中
動的更新: 読み取り専用
ReplicaSelector を実装する完全修飾クラス名。これは、推奨される読み取りレプリカを見つけるためにブローカーによって使用されます。デフォルトでは、リーダーを返す実装を使用します。

reserved.broker.max.id

型: int
デフォルト: 1000
有効な値: [0,...]
重要度: 中
動的更新: 読み取り専用
broker.id に使用できる最大数。

sasl.client.callback.handler.class

タイプ: class
デフォルト: null
重要度: 中
動的更新: 読み取り専用

AuthenticateCallbackHandler インターフェイスを実装する SASL クライアントコールバックハンドラークラスの完全修飾名。

sasl.enabled.mechanisms

型: list

デフォルト: GSSAPI

重要度: 中

動的更新: per-broker

Kafka サーバーで有効になっている SASL メカニズムの一覧。このリストには、セキュリティープロバイダーを利用できるメカニズムが含まれている場合があります。GSSAPI のみがデフォルトで有効になっています。

sasl.jaas.config

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

JAAS 設定ファイルで使用される形式の SASL 接続の JAAS ログインコンテキストパラメーター。JAAS 設定ファイルの形式は、[こちら](#) で説明されています。値の形式は **loginModuleClass controlFlag (optionName=optionValue)***; です。ブローカーの場合、設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;` などです。

sasl.kerberos.kinit.cmd

タイプ: string

デフォルト: /usr/bin/kinit

重要度: 中

動的更新: ブローカーごと

Kerberos kinit コマンドパス。

sasl.kerberos.min.time.before.relogin

型: long

デフォルト: 60000

重要度: 中

動的更新: per-broker

更新試行間のログインスレッドのスリープ時間。

sasl.kerberos.principal.to.local.rules

型: list

デフォルト: DEFAULT

重要度: 中

動的更新: ブローカーごと

プリンシパル名から短縮名 (通常はオペレーティングシステムのユーザー名) にマッピングするためのルールの一覧です。ルールは順番に評価され、プリンシパル名と一致する最初のルールは、これを短縮名にマップするために使用されます。一覧の後続のルールは無視されます。デフォルトでは、`{username}/{hostname}@{REALM}` 形式のプリンシパル名は `{username}` にマッピングされません。形式の詳細は、[security authorization and acls](#) を参照してください。この設定は、**principal.builder.class** 設定によって `KafkaPrincipalBuilder` のエクステンションが提供される場合は無視されます。

sasl.kerberos.service.name

タイプ: string

デフォルト: null

重要度: 中

動的更新: ブローカーごと

Kafka が実行される Kerberos プリンシパル名。これは、Kafka の JAAS 設定または Kafka の設定で定義できます。

sasl.kerberos.ticket.renew.jitter

型: double

デフォルト: 0.05

重要度: 中

動的更新: per-broker

更新時間に追加されたランダムなジッターの割合。

sasl.kerberos.ticket.renew.window.factor

型: double

デフォルト: 0.8

重要度: 中

動的更新: per-broker

ログインスレッドは、最後の更新からチケットの有効期限までの指定された時間のウィンドウファクターに達するまでスリープし、その時点でチケットの更新を試みます。

sasl.login.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

動的更新: 読み取り専用

AuthenticateCallbackHandler インターフェイスを実装する SASL ログインコールバックハンドラークラスの完全修飾名。ブローカーの場合、ログインコールバックハンドラー設定の前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler` などです。

sasl.login.class

タイプ: class

デフォルト: null

重要度: 中

動的更新: 読み取り専用

Login インターフェイスを実装するクラスの完全修飾名。ブローカーの場合、ログイン設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin` です。

sasl.login.refresh.buffer.seconds

型: short

デフォルト: 300

重要度: 中

動的更新: per-broker

クレデンシャルを更新するときに維持するクレデンシャルの有効期限が切れるまでのバッファ時間 (秒単位)。更新が、バッファ秒数よりも有効期限に近いときに発生する場合、更新は、可能な限り多くのバッファ時間を維持するために上に移動されます。設定可能な値は 0 から 3600 (1 時間) です。値を指定しない場合は、デフォルト値の 300 (5 分) が使用されます。この値および `sasl.login.refresh.min.period.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合にどちらも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.min.period.seconds

型: short

デフォルト: 60

重要度: 中

動的更新: per-broker

ログイン更新スレッドがクレデンシャルを更新する前に待機する最小時間 (秒単位)。設定可能な値は 0 から 900 (15 分) です。値を指定しない場合は、デフォルト値の 60 (1 分) が使用されます。この値および `sasl.login.refresh.buffer.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合に両方とも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.factor

型: double

デフォルト: 0.8

重要度: 中

動的更新: per-broker

ログイン更新スレッドは、クレデンシャルの有効期間との関連で指定の期間ファクターに達するまでスリープ状態になり、達成した時点でクレデンシャルの更新を試みます。設定可能な値は 0.5(50%) から 1.0(100%) までです。値が指定されていない場合、デフォルト値の 0.8(80%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.jitter

型: double

デフォルト: 0.05

重要度: 中

動的更新: per-broker

ログイン更新スレッドのスリープ時間に追加されるクレデンシャルの存続期間に関連するランダムジッターの最大量。設定可能な値は 0 から 0.25(25%) です。値が指定されていない場合は、デフォルト値の 0.05(5%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.mechanism.inter.broker.protocol

型: string

デフォルト: GSSAPI

重要度: 中

動的更新: per-broker

inter-broker 通信に使用される SASL メカニズム。デフォルトは GSSAPI です。

sasl.oauthbearer.jwks.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

動的更新: 読み取り専用

プロバイダーの JWKS (JSON Web Key Set) を取得できる OAuth/OIDC プロバイダーの URL。URL は、HTTP(S) ベースまたはファイルベースにすることができます。URL が HTTP(S) ベースの場合、JWKS データは、ブローカーの起動時に設定された URL を介して OAuth/OIDC プロバイダーから取得されます。その時点で最新のすべてのキーは、受信リクエストのためにブローカーにキャッシュされます。まだキャッシュにない kid ヘッダークレーム値を含む JWT の認証リクエストを受信した場合、JWKS エンドポイントはオンデマンドで再度クエリーされます。ただし、ブローカーは、`sasl.oauthbearer.jwks.endpoint.refresh.ms` ミリ秒ごとに URL をポーリングして、それらを含む JWT リクエストを受信される前に、今後のキーでキャッシュを更新します。URL がファイルベースの場合、ブローカーは起動時に設定された場所から JWKS ファイルをロードします。JWT に JWKS ファイルにない kid ヘッダー値が含まれている場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.token.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

動的更新: 読み取り専用

OAuth/OIDCID プロバイダーの URL。URL が HTTP(S) ベースの場合、sasl.jaas.config の設定に基づいてログインリクエストが行われるのは、issuer のトークンエンドポイント URL です。URL がファイルベースの場合、認証に使用するために OAuth/OIDC ID プロバイダーによって発行されたアクセストークン (JWT シリアル化形式) を含むファイルを指定します。

sasl.server.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

動的更新: 読み取り専用

AuthenticateCallbackHandler インターフェイスを実装する SASL サーバーコールバックハンドラークラスの完全修飾名。サーバーコールバックハンドラーの前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.plain.sasl.server.callback.handler.class=com.example.CustomPlainCallbackHandler` などです。

security.inter.broker.protocol

型: string

デフォルト: PLAINTEXT

重要度: 中

動的更新: 読み取り専用

inter-broker 通信に使用されるセキュリティープロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。これと `inter.broker.listener.name` プロパティを同時に設定するとエラーになります。

socket.connection.setup.timeout.max.ms

型: long

デフォルト: 30000 (30 秒)

重要度: 中

動的更新: 読み取り専用

ソケット接続が確立されるまでクライアントが待機する最大時間。接続設定のタイムアウトは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。接続ストームを回避するために、タイムアウトに 0.2 のランダム化ファクターが適用され、計算された値よりも 20% 未満と 20% 超の間にランダムな範囲が発生します。

socket.connection.setup.timeout.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 中

動的更新: 読み取り専用

ソケット接続が確立されるまでクライアントが待機する時間。タイムアウトが経過する前に接続が構築されない場合、クライアントはソケットチャンネルを閉じます。

socket.listen.backlog.size

型: int

デフォルト: 50

有効な値: [1,...]

重要度: 中

動的更新: 読み取り専用

ソケットで保留中の接続の最大数。Linux では、設定を有効にするために、それに応じて

somaxconn および **tcp_max_syn_backlog** カーネルパラメーターを設定する必要がある場合もあります。

ssl.cipher.suites

タイプ: list

デフォルト: ""

重要度: 中

動的更新: ブローカーごと

暗号化スイートの一覧。これは、TLS または SSL ネットワークプロトコルを使用してネットワーク接続のセキュリティ設定をネゴシエートするために使用される認証、暗号化、MAC、および鍵交換アルゴリズムの名前付きの組み合わせです。デフォルトでは、利用可能なすべての暗号スイートがサポートされます。

ssl.client.auth

タイプ: string

デフォルト: none

有効な値: [required, requested, none]

重要度: 中

動的更新: ブローカーごと

クライアント認証を要求する kafka ブローカーを設定します。以下の設定は一般的な設定です。

- **ssl.client.auth=required:** 必要なクライアント認証が必要な場合
- **ssl.client.auth=requested:** これは、クライアント認証が任意であることを意味します。required とは異なり、このオプションが設定されている場合、クライアントは自身に関する認証情報を提供しないことを選択できます。
- **ssl.client.auth=none:** これは、クライアント認証が不要であることを意味します。

ssl.enabled.protocols

型: list

デフォルト: TLSv1.2, TLSv1.3

重要度: 中

動的更新: per-broker

SSL 接続で有効なプロトコルの一覧。Java 11 以降で実行する場合、デフォルトは 'TLSv1.2,TLSv1.3' で、それ以外の場合は 'TLSv1.2' になります。Java 11 のデフォルト値では、クライアントとサーバーの両方が TLSv1.3 をサポートする場合は TLSv1.3 を優先し、そうでない場合は TLSv1.2 にフォールバックします (両方が少なくとも TLSv1.2 をサポートすることを前提とします)。ほとんどの場合、このデフォルトは問題ありません。 **ssl.protocol** の設定ドキュメントも参照してください。

ssl.key.password

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

キーストアファイルの秘密鍵または 'ssl.keystore.key' で指定した PEM キーのパスワード。これは、双方向認証が設定されている場合にのみ、クライアントに必要です。

ssl.keymanager.algorithm

型: string

デフォルト: SunX509

重要度: 中

動的更新: per-broker

SSL 接続のキーマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたキーマネージャーファクトリーアルゴリズムです。

ssl.keystore.certificate.chain

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

'ssl.keystore.type' で指定された形式の証明書チェーン。デフォルトの SSL エンジンファクトリーは、X.509 証明書の一覧を含む PEM 形式のみをサポートします。

ssl.keystore.key

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

ssl.keystore.type で指定された形式の秘密鍵。デフォルトの SSL エンジンファクトリーは、PKCS#8 キーを持つ PEM 形式のみをサポートします。鍵が暗号化されている場合は、'ssl.key.password' を使用して鍵のパスワードを指定する必要があります。

ssl.keystore.location

タイプ: string

デフォルト: null

重要度: 中

動的更新: ブローカーごと

キーストアファイルの場所。これはクライアントではオプションで、クライアントの双方向認証に使用できます。

ssl.keystore.password

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

キーストアファイルのストアパスワード。これはクライアントではオプションで、'ssl.keystore.location' が設定されている場合にのみ必要です。キーストアのパスワードは PEM 形式ではサポートされません。

ssl.keystore.type

型: string

デフォルト: JKS

重要度: 中

動的更新: ブローカーごと

キーストアファイルのファイル形式。これはクライアントにとってオプションになります。

ssl.protocol

型: string

デフォルト: TLSv1.3

重要度: 中

動的更新: per-broker

SSLContext の生成に使用される SSL プロトコル。Java 11 以降で実行する場合、デフォルトは 'TLSv1.3' になります。それ以外の場合は 'TLSv1.2' になります。この値は、ほとんどのユースケースで問題ありません。最近の JVM で許可される値は 'TLSv1.2' および 'TLSv1.3' です。'TLS'、'TLSv1.1'、'SSL'、'SSLv2'、および 'SSLv3' は古い JVM でサポートされる可能性があります。既知のセキュリティ脆弱性のために使用は推奨されません。この設定のデフォルト値および 'ssl.enabled.protocols' では、サーバーが 'TLSv1.3' に対応していない場合は、クライアントは 'TLSv1.2' にダウングレードされます。この設定が 'TLSv1.2' に設定されている場合、'TLSv1.3' が ssl.enabled.protocols の値の1つで、サーバーが 'TLSv1.3' のみをサポートする場合でも、クライアントは 'TLSv1.3' を使用しません。

ssl.provider

タイプ: string

デフォルト: null

重要度: 中

動的更新: ブローカーごと

SSL 接続に使用されるセキュリティプロバイダーの名前。デフォルト値は JVM のデフォルトのセキュリティプロバイダーです。

ssl.trustmanager.algorithm

型: string

デフォルト: PKIX

重要度: 中

動的更新: ブローカーごと

SSL 接続のトラストマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたトラストマネージャーファクトリーアルゴリズムです。

ssl.truststore.certificates

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

'ssl.truststore.type' で指定された形式の信頼できる証明書。デフォルトの SSL エンジンファクトリーは、X.509 証明書を使用する PEM 形式のみをサポートします。

ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 中

動的更新: ブローカーごと

トラストストアファイルの場所。

ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 中

動的更新: ブローカーごと

トラストストアファイルのパスワード。パスワードが設定されていない場合、設定されたトラストストアファイルは引き続き使用されますが、整合性チェックは無効になります。トラストストアのパスワードは PEM 形式ではサポートされません。

ssl.truststore.type

型: string

デフォルト: JKS
重要度: 中
動的更新: ブローカーごと
トラストストアファイルのファイル形式。

zookeeper.clientCnxnSocket

タイプ: string
デフォルト: null
重要度: 中
動的更新: 読み取り専用
通常、ZooKeeper への TLS 接続を使用する場合は **org.apache.zookeeper.ClientCnxnSocketNetty** に設定されます。同じ名前が付けられた **zookeeper.clientCnxnSocket** システムプロパティを介して設定された明示的な値を上書きします。

zookeeper.ssl.client.enable

タイプ: boolean
デフォルト: false
重要度: 中
動的更新: 読み取り専用
ZooKeeper への接続時に TLS を使用するようにクライアントを設定します。明示的な値は、**zookeeper.client.secure** システムプロパティを介して設定された値を上書きします (別の名前に注意してください)。どちらも設定されていない場合は false に設定されます。true の場合は **zookeeper.clientCnxnSocket** を設定する必要があります (通常は **org.apache.zookeeper.ClientCnxnSocketNetty** に設定)。設定するその他の値には、**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** などがあります。

zookeeper.ssl.keystore.location

タイプ: string
デフォルト: null
重要度: 中
動的更新: 読み取り専用
ZooKeeper への TLS 接続でクライアント側の証明書を使用する場合のキーストアの場所。**zookeeper.ssl.keystore.location** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。

zookeeper.ssl.keystore.password

タイプ: password
デフォルト: null
重要度: 中
動的更新: 読み取り専用
ZooKeeper への TLS 接続でクライアント側の証明書を使用する場合のキーストアパスワード。**zookeeper.ssl.keystore.password** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。ZooKeeper はキーストアパスワードとは異なるキーパスワードをサポートしないため、キーストアのキーパスワードをキーストアのパスワードと同じ値に設定してください。そうしないと、ZooKeeper への接続は失敗します。

zookeeper.ssl.keystore.type

タイプ: string
デフォルト: null

重要度: 中

動的更新: 読み取り専用

ZooKeeper への TLS 接続でクライアント側の証明書を使用する場合のキーストアタイプ。 **zookeeper.ssl.keyStore.type** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。 **null** のデフォルト値は、キーストアのファイル名のエクステンションに基づいて、タイプが自動検出されることを意味します。

zookeeper.ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 中

動的更新: 読み取り専用

ZooKeeper への TLS 接続を使用する場合のトラストストアの場所。 **zookeeper.ssl.trustStore.location** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。

zookeeper.ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 中

動的更新: 読み取り専用

ZooKeeper への TLS 接続を使用する場合のトラストストアパスワード。 **zookeeper.ssl.trustStore.password** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。

zookeeper.ssl.truststore.type

タイプ: string

デフォルト: null

重要度: 中

動的更新: 読み取り専用

ZooKeeper への TLS 接続を使用する場合のトラストストアタイプ。 **zookeeper.ssl.trustStore.type** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。デフォルト値の **null** は、タイプがトラストストアのファイル名のエクステンションに基づいて自動検出されることを意味します。

alter.config.policy.class.name

タイプ: class

デフォルト: null

重要度: 低

動的更新: 読み取り専用

検証に使用する必要のある alter configs ポリシークラス。クラスは **org.apache.kafka.server.policy.AlterConfigPolicy** インターフェイスを実装する必要があります。

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

型: int

デフォルト: 11

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

ログディレクトリーレプリケーションクォータを変更するためにメモリー内に保持するサンプル数。

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

型: int

デフォルト: 1

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

ログディレクトリーレプリケーションクォータを変更する各サンプルの期間。

authorizer.class.name

タイプ: string

デフォルト: ""

重要度: 低

動的更新: 読み取り専用

承認のためにブローカーによって使用される、`org.apache.kafka.server.authorizer.Authorizer` インターフェイスを実装するクラスの完全修飾名。

client.quota.callback.class

タイプ: class

デフォルト: null

重要度: 低

動的更新: 読み取り専用

ClientQuotaCallback インターフェイスを実装するクラスの完全修飾名。これは、クライアント要求に適用されるクォータ制限を決定するために使用されます。デフォルトでは、ZooKeeper に保存されている `<user>`, `<client-id>`, `<user>` または `<client-id>` クォータが適用されます。指定されたリクエストでは、セッションのユーザープリンシパルと、リクエストのクライアント ID と一致する最も具体的なクォータが適用されます。

connection.failed.authentication.delay.ms

型: int

デフォルト: 100

有効な値: [0,...]

重要度: 低

動的更新: 読み取り専用

認証に失敗した場合の connection close 遅延: これは、認証に失敗した場合に connection close が遅延する時間 (ミリ秒単位) です。これは、接続のタイムアウトを防ぐために、connections.max.idle.ms 未満になるように設定する必要があります。

controller.quorum.retry.backoff.ms

型: int

デフォルト: 20

重要度: 低

動的更新: 読み取り専用

特定のトピックパーティションに対して失敗したリクエストを再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

controller.quota.window.num

型: int

デフォルト: 11

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

コントローラーの変更クォータに対してメモリー内に保持するサンプル数。

controller.quota.window.size.seconds

型: int
デフォルト: 1
有効な値: [1,...]
重要度: 低
動的更新: 読み取り専用
コントローラーの変更クォータの各サンプルでの期間。

create.topic.policy.class.name

タイプ: class
デフォルト: null
重要度: 低
動的更新: 読み取り専用
検証に使用する必要があるトピックポリシークラスを作成します。クラスは **org.apache.kafka.server.policy.CreateTopicPolicy** インターフェイスを実装する必要があります。

delegation.token.expiry.check.interval.ms

型: long
デフォルト: 3600000 (1時間)
有効な値: [1,...]
重要度: 低
動的更新: 読み取り専用
期限切れの委任トークンを削除する間隔をスキャンします。

kafka.metrics.polling.interval.secs

型: int
デフォルト: 10
有効な値: [1,...]
重要度: 低
動的更新: 読み取り専用
kafka.metrics.reporters 実装で使用できるメトリクスポーリング間隔 (秒単位)。

kafka.metrics.reporters

タイプ: list
デフォルト: ""
重要度: 低
動的更新: 読み取り専用
Yammer メトリクスカスタムレポーターとして使用するクラスの一覧。レポーターは **kafka.metrics.KafkaMetricsReporter** トレイトを実装する必要があります。クライアントがカスタムレポーターで JMX 操作を公開する必要がある場合、カスタムレポーターは **kafka.metrics.KafkaMetricsReporterMBean** トレイトを拡張する MBean トレイトを追加で実装し、登録された MBean が標準の MBean 規則に準拠するようにする必要があります。

listener.security.protocol.map

型: string
デフォルト:
PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
重要度: low
動的更新: per-broker
リスナー名とセキュリティープロトコル間のマッピング。これは、複数のポートまたは IP で同じセキュリティープロトコルを使用できるように定義する必要があります。たとえば、SSL が両方に必要であっても、内部トラフィックと外部トラフィックは分離することができます。具体的には、

ユーザーは INTERNAL および EXTERNAL という名前のリスナーを定義し、このプロパティを **INTERNAL:SSL,EXTERNAL:SSL** のように定義できます。表示されているように、キーと値はコロンで区切られ、マップエントリはコマンドで区切ります。各リスナー名はマップで一度だけ表示されます。設定名に正規化された接頭辞 (リスナー名は小文字) を追加して、各リスナーに異なるセキュリティ (SSL および SASL) 設定を設定できます。たとえば、INTERNAL リスナーに異なるキーストアを設定するには、**listener.name.internal.ssl.keystore.location** という名前の設定が設定されます。リスナー名の設定が設定されていない場合、設定は汎用設定 (**ssl.keystore.location** など) にフォールバックします。KRaft では、明示的なマッピングが提供されておらず、他のセキュリティプロトコルが使用されていない場合、**controller.listener.names** で定義されたリスナー名から PLAINTEXT へのデフォルトのマッピングが想定されることに注意してください。

log.message.downconversion.enable

タイプ: boolean

デフォルト: true

重要度: 低

動的更新: クラスター全体

この設定は、消費リクエストを満たすためにメッセージ形式のダウンコンバートを有効にするかどうかを制御します。**false** に設定すると、ブローカーは古いメッセージ形式を想定しているコンシューマーに対してダウンコンバートを実行しません。ブローカーは、そのような古いクライアントからの消費リクエストに対して、**UNSUPPORTED_VERSION** エラーを返します。この設定は、フォロワーへのレプリケーションに必要なメッセージ形式の変換には適用されません。

metric.reporters

タイプ: list

デフォルト: ""

重要度: 低

動的更新: クラスター全体

メトリクスレポーターとして使用するクラスの一

覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string

デフォルト: INFO

重要度: 低

動的更新: 読み取り専用

メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

メトリクスサンプルが計算される時間枠。

password.encoder.cipher.algorithm

型: string

デフォルト: AES/CBC/PKCS5Padding

重要度: 低

動的更新: 読み取り専用

動的に設定されたパスワードをエンコードする際に使用する暗号アルゴリズム。

password.encoder.iterations

型: int

デフォルト: 4096

有効な値: [1024,...]

重要度: 低

動的更新: 読み取り専用

動的に設定されたパスワードのエンコードに使用される反復カウント。

password.encoder.key.length

型: int

デフォルト: 128

有効な値: [8,...]

重要度: 低

動的更新: 読み取り専用

動的に設定されたパスワードをエンコードするために使用されるキーの長さ。

password.encoder.keyfactory.algorithm

タイプ: string

デフォルト: null

重要度: 低

動的更新: 読み取り専用

動的に設定されたパスワードをエンコードするのに使用される SecretKeyFactory アルゴリズム。それ以外の場合は、デフォルトで PBKDF2WithHmacSHA512 および PBKDF2WithHmacSHA1 になります。

quota.window.num

型: int

デフォルト: 11

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

クライアントクォータのメモリー内に保持するサンプル数。

quota.window.size.seconds

型: int

デフォルト: 1

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

クライアントクォータの各サンプルの時間の長さ。

replication.quota.window.num

型: int
デフォルト: 11
有効な値: [1,...]
重要度: 低
動的更新: 読み取り専用
レプリケーションクォータのメモリーに保持するサンプル数。

replication.quota.window.size.seconds

型: int
デフォルト: 1
有効な値: [1,...]
重要度: 低
動的更新: 読み取り専用
レプリケーションクォータの各サンプルの期間。

sasl.login.connect.timeout.ms

Type: int
Default: null
Importance: 低
Dynamic update: 読み取り専用
外部認証プロバイダーの接続タイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.read.timeout.ms

Type: int
Default: null
Importance: 低
Dynamic update: 読み取り専用
外部認証プロバイダーの読み取りタイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.max.ms

型: long
デフォルト: 10000 (10 秒)
重要度: 低
動的更新: 読み取り専用
外部認証プロバイダーへのログイン試行間の最大待機時間のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は sasl.login.retry.backoff.ms 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は sasl.login.retry.backoff.max.ms 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.ms

型: long
デフォルト: 100
重要度: 低
動的更新: 読み取り専用
外部認証プロバイダーへのログイン試行間の最初の待機のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は sasl.login.retry.backoff.ms 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は sasl.login.retry.backoff.max.ms 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

sasl.oauthbearer.clock.skew.seconds

型: int

デフォルト: 30

重要度: 低

動的更新: 読み取り専用

OAuth/OIDC ID プロバイダーとブローカーの時間の差を考慮した秒単位の (オプションの) 値。

sasl.oauthbearer.expected.audience

タイプ: list

デフォルト: null

重要度: 低

動的更新: 読み取り専用

JWT が想定される対象者の1つに対して発行されたことを確認するためにブローカーが使用する (オプションの) コンマ区切りの設定です。JWT は標準的な OAuth の aud クレームについて検査され、この値が設定されている場合、ブローカーは JWT の aud クレームから値が完全に一致するかどうかを確認するために照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.expected.issuer

タイプ: string

デフォルト: null

重要度: 低

動的更新: 読み取り専用

ブローカーが JWT が想定される issuer によって作成されたことを確認するために使用する (オプションの) 設定。JWT は標準の OAuth iss クレームについて検査され、この値が設定されている場合、ブローカーは JWT の iss クレームにあるものと正確に照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.jwks.endpoint.refresh.ms

型: long

デフォルト: 3600000 (1 hour)

重要度: 低

動的更新: 読み取り専用

ブローカーが JWT の署名を検証するためのキーを含む JWKS (JSON Web Key Set) キャッシュを更新するまで待機するミリ秒単位の (オプションの) 値。

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

型: long

デフォルト: 10000 (10 秒)

重要度: 低

動的更新: 読み取り専用

外部認証プロバイダーから JWKS (JSON Web Key Set) を取得する試行間の最大待機時間のミリ秒単位の (オプションの) 値。JWKS の取得では、sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 設定で指定された最大待機時間まで試行間の待機時間が2倍になります。

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

型: long

デフォルト: 100

重要度: 低

動的更新: 読み取り専用

外部認証プロバイダーからの JWKS (JSON Web Key Set) 取得する試行間の最初の待機のミリ秒単

位の (オプションの) 値。JWKS の取得では、`sasl.oauthbearer.jwks.endpoint.retry.backoff.ms` 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、`sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms` 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

sasl.oauthbearer.scope.claim.name

Type: 文字列

Default: スコープ

Importance: 低

Dynamic update: 読み取り専用

スコープの OAuth クレームはスコープと呼ばれることがよくありますが、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、この (オプションの) 設定は JWT ペイロードのクレームに含まれるスコープに使用する別の名前を提供できます。

sasl.oauthbearer.sub.claim.name

Type: string

Default: sub

Importance: 低

Dynamic update: 読み取り専用

サブジェクトの OAuth クレームは sub と呼ばれることがよくありますが、この (オプションの) 設定は、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、JWT ペイロードのクレームに含まれるサブジェクトに使用する別の名前を提供できます。

security.providers

タイプ: string

デフォルト: null

重要度: 低

動的更新: 読み取り専用

設定可能なクリエイタークラスのリストで、それぞれがセキュリティーアルゴリズムを実装するプロバイダーを返します。これらのクラスは

org.apache.kafka.common.security.auth.SecurityProviderCreator インターフェイスを実装する必要があります。

ssl.endpoint.identification.algorithm

タイプ: string

デフォルト: https

重要度: 低

動的更新: ブローカーごと

サーバー証明書を使用してサーバーのホスト名を検証するエンドポイント識別アルゴリズム。

ssl.engine.factory.class

タイプ: class

デフォルト: null

重要度: 低

動的更新: ブローカーごと

SSL Engine オブジェクトを提供する `org.apache.kafka.common.security.auth.SslEngineFactory` タイプのクラス。デフォルト値は `org.apache.kafka.common.security.ssl.DefaultSslEngineFactory` です。

ssl.principal.mapping.rules

型: string

デフォルト: DEFAULT

重要度: 低

動的更新: 読み取り専用

クライアント証明書の識別名から短縮名にマッピングするためのルールの一覧。ルールは順番に評価され、プリンシパル名と一致する最初のルールは、これを短縮名にマップするために使用されます。一覧の後続のルールは無視されます。デフォルトでは、X.500 証明書の識別名はプリンシパルになります。形式の詳細は、[security authorization and acls](#) を参照してください。この設定は、**principal.builder.class** 設定によって KafkaPrincipalBuilder のエクステンションが提供される場合は無視されます。

ssl.secure.random.implementation

タイプ: string

デフォルト: null

重要度: 低

動的更新: ブローカーごと

SSL 暗号操作に使用する SecureRandom PRNG 実装。

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

型: int

デフォルト: 10000 (10 秒)

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

タイムアウトしたトランザクションをロールバックする間隔。

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

型: int

デフォルト: 3600000 (1 時間)

有効な値: [1,...]

重要度: 低

動的更新: 読み取り専用

transactional.id.expiration.ms を渡すために期限切れとなったトランザクションを削除する間隔。

zookeeper.ssl.cipher.suites

タイプ: list

デフォルト: null

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS negotiation(csv) で使用される有効な暗号スイートを指定しま

す。**zookeeper.ssl.ciphersuites** システムプロパティを介して設定された明示的な値を上書きします ("ciphersuites" という単語に注意)。デフォルト値の **null** は、有効な暗号スイートのリストが、使用される Java ランタイムによって決定されることを意味します。

zookeeper.ssl.crl.enable

タイプ: boolean

デフォルト: false

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS プロトコルで、証明書失効リストを有効にするかどうかを指定しま

す。**zookeeper.ssl.crl** システムプロパティを介して設定された明示的な値を上書きします (短い名前に注意)。

zookeeper.ssl.enabled.protocols

タイプ: list

デフォルト: null

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS negotiation(csv) で有効なプロトコルを指定しま

す。 **zookeeper.ssl.enabledProtocols** システムプロパティを介して設定された明示的な値を上書きします (camelCase に注意)。デフォルト値の **null** は、有効化されたプロトコルが

zookeeper.ssl.protocol 設定プロパティの値になることを意味します。

zookeeper.ssl.endpoint.identification.algorithm

型: string

デフォルト: HTTPS

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS ネゴシエーションプロセスのホスト名検証を有効にするかどうかを指定します。この場合 (大文字小文字を区別しない) "https" は ZooKeeper ホスト名の検証が有効で、明示的な空白値は無効であることを意味します (無効にすることはテスト目的でのみ推奨されます)。明示的な値は、 **zookeeper.ssl.hostnameVerification** システムプロパティを介して設定される "true" または "false" の値を上書きします (異なる名前と値に注意してください。 true は https を意味し、 false は空白を意味します)。

zookeeper.ssl.ocsp.enable

タイプ: boolean

デフォルト: false

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS プロトコルで Online Certificate Status Protocol を有効にするかどうかを指定します。 **zookeeper.ssl.ocsp** システムプロパティを介して設定した明示的な値を上書きします (短い名前に注意)。

zookeeper.ssl.protocol

型: string

デフォルト: TLSv1.2

重要度: 低

動的更新: 読み取り専用

ZooKeeper TLS ネゴシエーションで使用されるプロトコルを指定します。明示的な値は、同じ名前が付けられた **zookeeper.ssl.protocol** システムプロパティで設定された値を上書きします。

第2章 トピック設定プロパティー

cleanup.policy

タイプ: list
デフォルト: delete
有効な値: [compact, delete]
サーバーデフォルトプロパティー: log.cleanup.policy
重要度: 中

"delete" または "compact" のいずれか、または両方である文字列。この文字列は、古いログセグメントで使用する保持ポリシーを指定します。デフォルトのポリシー ("delete") は、保持期間またはサイズ制限に達すると古いセグメントを破棄します。"compact" 設定は、トピックで [ログコンパクション](#) を有効にします。

compression.type

タイプ: string
デフォルト: producer
有効な値: [uncompressed, zstd, lz4, snappy, gzip, producer]
サーバーデフォルトプロパティー: compression.type
重要度: 中

特定のトピックの最終的な圧縮タイプを指定します。この設定は、標準の圧縮コーデック ('gzip'、'snappy'、'lz4'、'zstd') を受け入れます。さらに、圧縮なしに相当する 'uncompressed' と、プロデューサーによって設定された元の圧縮コーデックを維持する 'producer' も使用できます。

delete.retention.ms

型: long
デフォルト: 86400000 (1日)
有効な値: [0,...]
サーバーのデフォルトプロパティー: log.cleaner.delete.retention.ms
重要度: 中

[ログ圧縮](#) トピックの tombstone 削除マーカーを保持する期間。また、この設定は、最終ステージの有効なスナップショットを確実に取得するために、コンシューマーがオフセット 0 から開始する場合に読み取りを完了しなければならない時間にも制限を与えます (そうでない場合、スキャンを完了する前に、削除 tombstone が収集される可能性があります)。

file.delete.delay.ms

型: long
デフォルト: 60000 (1分)
有効な値: [0,...]
サーバーのデフォルトプロパティー: log.segment.delete.delay.ms
重要度: 中
ファイルシステムからファイルを削除するまでの待機時間。

flush.messages

型: long
デフォルト: 9223372036854775807
有効な値: [0,...]
サーバーのデフォルトプロパティー: log.flush.interval.messages
重要度: 中

この設定により、ログに書き込まれたデータの fsync を強制する間隔を指定できます。たとえば、これが 1 に設定されている場合、すべてのメッセージの後に fsync を実行します。5 に設定されている場合は、5 つのメッセージごとに fsync します。一般的には、これを設定せず、永続性のためにレ

アプリケーションを使用し、より効率的であるため、オペレーティングシステムのバックグラウンドフラッシュ機能を使用することが推奨されます。この設定は、トピックごとに上書きできます ([トピックごとの設定セクション](#) を参照)。

flush.ms

型: long

デフォルト: 9223372036854775807

有効な値: [0,...]

サーバーのデフォルトプロパティ: log.flush.interval.ms

重要度: 中

この設定により、ログに書き込まれるデータの fsync を強制する時間間隔を指定できます。たとえば、これが 1000 に設定されていた場合は、1000 ミリ秒が経過した後に fsync を実行します。一般的には、これを設定せず、永続性のためにレプリケーションを使用し、より効率的であるため、オペレーティングシステムのバックグラウンドフラッシュ機能を使用することが推奨されます。

follower.replication.throttled.replicas

型: list

デフォルト: ""

有効な値: [partitionId]:[brokerId],[partitionId]:[brokerId],...

サーバーのデフォルトプロパティ: follower.replication.throttled.replicas

重要度: 中

ログレプリケーションをフォロワー側でスロットリングする必要があるレプリカの一覧。リストには、レプリカのセットを [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... の形式で記述する必要があります。または、ワイルドカード * を使用して、このトピックのすべてのレプリカを調整できます。

index.interval.bytes

型: int

デフォルト: 4096 (4 キビバイト)

有効な値: [0,...]

サーバーのデフォルトプロパティ: log.index.interval.bytes

重要度: 中

この設定は、Kafka がインデックスエントリをオフセットインデックスに追加する頻度を制御します。デフォルト設定では、およそ 4096 バイトごとにメッセージを確実にインデックス化します。インデックス化を増やすことで、読み取りをログ内の正確な位置に近づけることができますが、インデックスは大きくなりますおそらくこれを変更する必要はありません。

leader.replication.throttled.replicas

型: list

デフォルト: ""

有効な値: [partitionId]:[brokerId],[partitionId]:[brokerId],...

サーバーのデフォルトプロパティ: leader.replication.throttled.replicas

重要度: 中

ログレプリケーションをリーダー側でスロットリングする必要があるレプリカの一覧。リストには、レプリカのセットを [PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:... の形式で記述する必要があります。または、ワイルドカード * を使用して、このトピックのすべてのレプリカを調整できます。

max.compaction.lag.ms

型: long

デフォルト: 9223372036854775807

有効な値: [1,...]

サーバーのデフォルトプロパティ: log.cleaner.max.compaction.lag.ms

重要度: 中

メッセージがログのコンパクションの対象外のままになる最大時間。圧縮されるログにのみ適用されます。

max.message.bytes

型: int

デフォルト: 1048588

有効な値: [0,...]

サーバーのデフォルトプロパティ: message.max.bytes

重要度: 中

Kafka によって許可される最大のレコードバッチサイズ (圧縮が有効な場合は圧縮後)。これが増加し、0.10.2 より古いコンシューマーが存在する場合、コンシューマーのフェッチサイズも増加して、このような大きなレコードバッチをフェッチできるようにする必要があります。最新のメッセージ形式バージョンでは、効率化のためにレコードは常にバッチにグループ化されます。以前のメッセージ形式のバージョンでは、圧縮されていないレコードはバッチにグループ化されず、この制限はその場合の単一レコードにのみ適用されます

message.format.version

型: string

デフォルト: 3.0-IV1

有効な値: [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0]

サーバーのデフォルトプロパティ: log.message.format.version

Importance: medium

[非推奨] ブローカーがログにメッセージを追加するために使用するメッセージ形式バージョンを指定します。**inter.broker.protocol.version** が 3.0 以上の場合、この設定の値は常に **3.0** であると想定されます (実際の設定値は無視されます)。それ以外の場合は、この値は有効な ApiVersion である必要があります。たとえば、0.10.0、1.1、2.8、3.0 です。特定のメッセージ形式のバージョンを設定することで、ユーザーは、ディスク上の既存のメッセージすべてが指定したバージョンよりも小さいか、または等しいことを認定します。この値を誤って設定すると、以前のバージョンを持つコンシューマーが、認識されない形式でメッセージを受信するため、破損します。

message.timestamp.difference.max.ms

型: long

デフォルト: 9223372036854775807

有効な値: [0,...]

サーバーのデフォルトプロパティ: log.message.timestamp.difference.max.ms

重要度: 中

ブローカーがメッセージを受信したときのタイムスタンプと、メッセージに指定されたタイムスタンプとの間の最大差。message.timestamp.type=CreateTime の場合、タイムスタンプの相違点がこのしきい値を超えるとメッセージが拒否されます。この設定は、message.timestamp.type=LogAppendTime の場合は無視されます。

message.timestamp.type

型: string

デフォルト: CreateTime

有効な値: [CreateTime, LogAppendTime]

サーバーのデフォルトプロパティ: log.message.timestamp.type

重要度: medium

メッセージのタイムスタンプが、メッセージ作成時間かログの追加時間であるかを定義します。値は **CreateTime** または **LogAppendTime** である必要があります。

min.cleanable.dirty.ratio

型: double
 デフォルト: 0.5
 有効な値: [0,...,1]
 サーバーのデフォルトプロパティ: log.cleaner.min.cleanable.ratio
 重要度: 中

この設定は、ログコンパクターがログのクリーンアップを試行する頻度を制御します (**ログコンパクション** が有効になっている場合)。デフォルトでは、50% を超えるログが圧縮されているログのクリーニングは回避されます。この比率は、重複によって無駄になるログの最大スペースを制限します (50% の場合、最大でログの 50% が重複している可能性があります)。比率が高いほど、より少ない、より効率的なクリーニングを意味しますが、ログ内の無駄なスペースが多くなることを意味します。max.compaction.lag.ms または min.compaction.lag.ms 設定も指定されている場合、ログコンパクターは、次のいずれかの場合にすぐにログをコンパクションの対象と見なします。(i) ダーティー比率のしきい値に達し、ログに少なくとも min.compaction.lag.ms duration 期間のダーティー (圧縮されていない) レコードがある場合、または (ii) ログに最大で max.compaction.lag.ms 期間のダーティー (圧縮されていない) レコードがある場合。

min.compaction.lag.ms

型: long
 デフォルト: 0
 有効な値: [0,...]
 サーバーのデフォルトプロパティ: log.cleaner.min.compaction.lag.ms
 重要度: 中

メッセージがログで圧縮されないままになる最小時間。圧縮されるログにのみ適用されます。

min.insync.replicas

型: int
 デフォルト: 1
 有効な値: [1,...]
 サーバーのデフォルトプロパティ: min.insync.replicas
 重要度: 中

プロデューサーが acks を "all" (または "-1") に設定すると、この設定は、書き込みが成功したとみなされるために書き込みを確認する必要があるレプリカの最小数を指定します。この最小値が満たされない場合、プロデューサーは例外 (NotEnoughReplicas または NotEnoughReplicasAfterAppend のいずれか) を発生させます。 **min.insync.replicas** と **acks** を併用することで、より高い持続性が保証されます。一般的なシナリオでは、レプリケーション係数 3 のトピックを作成し、 **min.insync.replicas** を 2 に設定し、 **acks** を all にして生成します。これにより、レプリカの大部分が書き込みを受信しない場合に、プロデューサーは確実に例外を発生させます。

preallocate

タイプ: boolean
 デフォルト: false
 サーバーデフォルトプロパティ: log.preallocate
 重要度: 中

新規ログセグメントの作成時にファイルをディスクに事前割り当てする場合は True。

retention.bytes

タイプ: long
 デフォルト: -1
 サーバーデフォルトプロパティ: log.retention.bytes

重要度: 中

この設定は、"delete" 保持ポリシーを使用している場合に、領域を解放するために古いログセグメントを破棄する前に、パーティション (ログセグメントで設定される) が成長できる最大サイズを制御します。デフォルトでは、サイズ制限はなく、時間制限のみがあります。この制限はパーティションレベルで適用されるため、これにパーティションの数を掛けて、トピックの保持をバイト単位で計算します。

retention.ms

型: long

デフォルト: 604800000 (7 日)

有効な値: [-1,...]

サーバーのデフォルトプロパティ: log.retention.ms

重要度: 中

この設定は、"delete" 保持ポリシーを使用している場合に、古いログセグメントを破棄して領域を解放する前にログを保持する最大時間を制御します。これは、コンシューマーがどれだけ早くデータを読み込まなければならないかという SLA を意味します。-1 に設定すると、時間制限は適用されません。

segment.bytes

型: int

デフォルト: 1073741824 (1 ギビバイト)

有効な値: [14,...]

サーバーのデフォルトプロパティ: log.segment.bytes

重要度: 中

この設定は、ログのセグメントファイルサイズを制御します。保持とクリーニングは常に 1 ファイルずつ行われるため、セグメントサイズが大きくなると、ファイルは少なくなります。保持の制御が細かくできなくなります。

segment.index.bytes

型: int

デフォルト: 10485760 (10 mebibytes)

有効な値: [0,...]

サーバーのデフォルトプロパティ: log.index.size.max.bytes

重要度: 中

この設定では、オフセットをファイルの位置にマップするインデックスのサイズを制御します。このインデックスファイルを事前割り当てし、ログロールの後にのみ縮小します。通常、この設定を変更する必要はありません。

segment.jitter.ms

型: long

デフォルト: 0

有効な値: [0,...]

サーバーのデフォルトプロパティ: log.roll.jitter.ms

重要度: 中

セグメントローリングの大規模な集約を避けるために、スケジュールされたセグメントロール時間から差し引いたランダムなジッターの最大値。

segment.ms

型: long

デフォルト: 604800000 (7 日)

有効な値: [1,...]

サーバーのデフォルトプロパティ: log.roll.ms

重要度: 中

この設定は、セグメントファイルがいっぱいでない場合でも、Kafka がログを強制的にロールして、古いデータを削除または圧縮できるようにする期間を制御します。

unclean.leader.election.enable

タイプ: boolean

デフォルト: false

サーバーデフォルトプロパティ: unclean.leader.election.enable

重要度: 中

データが失われる可能性がある場合でも、ISR セットにないレプリカを最後の手段としてリーダーとして選出できるようにするかどうかを示します。

message.downconversion.enable

タイプ: boolean

デフォルト: true

サーバーデフォルトプロパティ: log.message.downconversion.enable

重要度: 低

この設定は、消費リクエストを満たすためにメッセージ形式のダウンコンバートを有効にするかどうかを制御します。**false** に設定すると、ブローカーは古いメッセージ形式を想定しているコンシューマーに対してダウンコンバートを実行しません。ブローカーは、そのような古いクライアントからの消費リクエストに対して、**UNSUPPORTED_VERSION** エラーを返します。この設定は、フォロワーへのレプリケーションに必要なメッセージ形式の変換には適用されません。

第3章 コンシューマー設定プロパティー

key.deserializer

タイプ: class

重要度: 高

`org.apache.kafka.common.serialization.Deserializer` インターフェイスを実装するキーのデシリアライザークラス。

value.deserializer

タイプ: class

重要度: 高

`org.apache.kafka.common.serialization.Deserializer` インターフェイスを実装する値のデシリアライザークラス。

bootstrap.servers

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 高

Kafka クラスターへの最初の接続を確立するために使用されるホストとポートのペアの一覧。クライアントは、ブートストラップ用にここで指定されたサーバーに関係なく、すべてのサーバーを利用します。この一覧は、サーバーのフルセットを検出するために使用される最初のホストにのみ影響します。この一覧は、`host1:port1,host2:port2,...` の形式にする必要があります。これらのサーバーは、(動的に変更される可能性がある) 完全なクラスターメンバーシップを検出するための最初の接続にだけ使用されるため、このリストにはサーバーの完全なセットを含める必要はありません (ただし、サーバーがダウンした場合に備えて、複数のサーバーが必要になる場合があります)。

fetch.min.bytes

型: int

デフォルト: 1

有効な値: [0,...]

重要度: 高

サーバーがフェッチ要求に対して返す必要のあるデータの最小量。利用可能なデータが不十分な場合、リクエストは、リクエストに回答する前に、十分なデータが蓄積されるのを待ちます。デフォルト設定の1バイトは、1バイトのデータが使用可能になるか、データの到着を待機してフェッチ要求がタイムアウトするとすぐに、フェッチ要求に回答することを意味します。これを1を超える値に設定すると、サーバーは大量のデータが蓄積されるのを待機することになり、追加の遅延を犠牲にしてサーバーのスループットを少し向上させることができます。

group.id

タイプ: string

デフォルト: null

重要度: 高

このコンシューマーが属するコンシューマーグループを識別する一意の文字列。このプロパティーは、コンシューマーが `subscribe(topic)` を使用したグループ管理機能または Kafka ベースのオフセット管理ストラテジーのいずれかを使用する場合に必要です。

heartbeat.interval.ms

型: int

デフォルト: 3000 (3 秒)

重要度: 高

Kafka のグループ管理機能を使用する場合の、ハートビートからコンシューマーコーディネーター間

の想定される時間。ハートビートは、コンシューマーのセッションがアクティブな状態を維持し、新しいコンシューマーがグループに参加したり離脱したりする際のリバランスを促進するために使用されます。この値は **session.timeout.ms** よりも低く設定する必要がありますが、通常はその値の 1/3 以下に設定する必要があります。さらに低く調整することで、通常のリバランスの予想時間を制御することもできます。

max.partition.fetch.bytes

型: int

デフォルト: 1048576 (1 メビバイト)

有効な値: [0,...]

重要度: 高

サーバーが返すパーティションごとのデータの最大量。レコードはコンシューマーによってバッチでフェッチされます。フェッチの最初の空でないパーティションの最初のレコードバッチがこの制限よりも大きい場合でも、コンシューマーが確実に前進することができるようにバッチが返されません。ブローカーによって許可される最大レコードバッチサイズは、**message.max.bytes** (ブローカー設定) または **max.message.bytes** (トピック設定) で定義されます。コンシューマーのリクエストサイズを制限する場合は、**fetch.max.bytes** を参照してください。

session.timeout.ms

型: int

デフォルト: 45000 (45 秒)

重要度: 高

Kafka のグループ管理機能の使用時にクライアント障害を検出するために使用されるタイムアウト。クライアントは定期的にハートビートを送信し、liveness をブローカーに示します。このセッションタイムアウトの期限切れ前にブローカーによってハートビートが受信されない場合、ブローカーはグループからこのクライアントを削除し、リバランスを開始します。この値は、**group.min.session.timeout.ms** および **group.max.session.timeout.ms** によってブローカー設定で設定される許容範囲内である必要があることに注意してください。

ssl.key.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルの秘密鍵または 'ssl.keystore.key' で指定した PEM キーのパスワード。これは、双方向認証が設定されている場合にのみ、クライアントに必要です。

ssl.keystore.certificate.chain

タイプ: password

デフォルト: null

重要度: 高

'ssl.keystore.type' で指定された形式の証明書チェーン。デフォルトの SSL エンジンファクトリーは、X.509 証明書の一覧を含む PEM 形式のみをサポートします。

ssl.keystore.key

タイプ: password

デフォルト: null

重要度: 高

ssl.keystore.type で指定された形式の秘密鍵。デフォルトの SSL エンジンファクトリーは、PKCS#8 キーを持つ PEM 形式のみをサポートします。鍵が暗号化されている場合は、'ssl.key.password' を使用して鍵のパスワードを指定する必要があります。

ssl.keystore.location

タイプ: string

デフォルト: null

重要度: 高

キーストアファイルの場所。これはクライアントではオプションで、クライアントの双方向認証に使用できます。

ssl.keystore.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルのストアパスワード。これはクライアントではオプションで、'ssl.keystore.location' が設定されている場合にのみ必要です。キーストアのパスワードは PEM 形式ではサポートされません。

ssl.truststore.certificates

タイプ: password

デフォルト: null

重要度: 高

'ssl.truststore.type' で指定された形式の信頼できる証明書。デフォルトの SSL エンジンファクトリーは、X.509 証明書を使用する PEM 形式のみをサポートします。

ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 高

トラストストアファイルの場所。

ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 高

トラストストアファイルのパスワード。パスワードが設定されていない場合、設定されたトラストストアファイルは引き続き使用されますが、整合性チェックは無効になります。トラストストアのパスワードは PEM 形式ではサポートされません。

allow.auto.create.topics

タイプ: boolean

デフォルト: true

重要度: 中

トピックをサブスクライブまたは割り当てるときに、ブローカーでのトピックの自動作成を許可します。サブスクライブするトピックは、ブローカーが **auto.create.topics.enable** ブローカー設定を使用して許可されている場合にのみ自動的に作成されます。この設定は、0.11.0 より古いブローカーを使用する場合は、**false** に設定する必要があります。

auto.offset.reset

タイプ: string

デフォルト: latest

有効な値: [latest, earliest, none]

重要度: 中

Kafka に初期オフセットがない場合や、現在のオフセットがサーバー上に存在しない場合 (例: データが削除された場合など) の対処方法。

- `earliest`: 自動的にオフセットを最も古いオフセットにリセットする
- `latest`: 自動的にオフセットを最新のオフセットにリセットする
- `none`: コンシューマーのグループに以前のオフセットが見つからない場合は、コンシューマーに例外を出力する
- `anything else`: コンシューマーに例外を出力する

`client.dns.lookup`

タイプ: string

デフォルト: `use_all_dns_ips`

有効な値: [`use_all_dns_ips`, `resolve_canonical_bootstrap_servers_only`]

重要度: 中

クライアントが DNS ルックアップを使用する方法を制御します。`use_all_dns_ips` に設定すると、正常な接続が確立されるまで、返された各 IP アドレスに順番に接続します。切断後に、次の IP が使用されます。すべての IP が一度使用されると、クライアントはホスト名から IP を再度解決します (ただし、JVM と OS の両方は DNS 名の検索をキャッシュします)。`resolve_canonical_bootstrap_servers_only` に設定すると、各ブートストラップアドレスを正規名の一覧に解決します。ブートストラップフェーズ後、これは `use_all_dns_ips` と同じように動作します。

`connections.max.idle.ms`

型: long

デフォルト: 540000 (9 分)

重要度: 中

この設定で指定された期間 (ミリ秒単位) の後にアイドル状態の接続を閉じます。

`default.api.timeout.ms`

型: int

デフォルト: 60000 (1 分)

有効な値: [0,...]

重要度: 中

クライアント API のタイムアウト (ミリ秒単位) を指定します。この設定は、`timeout` パラメーターを指定しないすべてのクライアント操作のデフォルトタイムアウトとして使用されます。

`enable.auto.commit`

タイプ: boolean

デフォルト: true

重要度: 中

true の場合、コンシューマーのオフセットは定期的にバックグラウンドでコミットされます。

`exclude.internal.topics`

タイプ: boolean

デフォルト: true

重要度: 中

サブスクライブされたパターンと一致する内部トピックをサブスクリプションから除外するべきかどうか。常に内部トピックに明示的にサブスクライブできます。

`fetch.max.bytes`

型: int

デフォルト: 52428800 (50 メビバイト s)

有効な値: [0,...]

重要度: 中

サーバーがフェッチリクエストに対して返す必要のあるデータの最大量。レコードはコンシューマーによってバッチで取得され、フェッチの最初の空ではないパーティションの最初のレコードバッチがこの値よりも大きい場合でも、コンシューマーが確実に前進することができるようにレコードバッチが返されます。そのため、絶対的な最大値ではありません。ブローカーによって許可される最大レコードバッチサイズは、**message.max.bytes** (ブローカー設定) または **max.message.bytes** (トピック設定) で定義されます。コンシューマーは複数のフェッチを並行して実行することに注意してください。

group.instance.id

タイプ: string

デフォルト: null

重要度: 中

エンドユーザーが提供するコンシューマーインスタンスの一意的識別子。non-empty strings のみが許可されます。設定されている場合、コンシューマーは静的メンバーとして扱われます。つまり、常に、この ID を持つ1つのインスタンスのみがコンシューマーグループで許可されます。これは、より大きなセッションタイムアウトと組み合わせて使用して、一時的な利用不可 (プロセス再起動など) によるグループのリバランスを回避します。設定しないと、コンシューマーは従来の動作である動的メンバーとしてグループに参加します。

isolation.level

タイプ: string

デフォルト: read_uncommitted

有効な値: [read_committed, read_uncommitted]

重要度: 中

トランザクションで書き込まれたメッセージを読み取る方法を制御します。**read_committed** に設定すると、`consumer.poll()` はコミットされたトランザクションメッセージのみを返します。**read_uncommitted** (デフォルト) に設定すると、`consumer.poll()` は中断されたトランザクションメッセージも含めて、すべてのメッセージを返します。非トランザクションメッセージは、どちらのモードでも無条件に返されます。

メッセージは常にオフセットの順序で返されます。そのため、**read_committed** モードでは、`consumer.poll()` は最初のオープントランザクションのオフセットよりも1つ小さい最後の安定したオフセット (LSO) までのメッセージのみを返します。特に、継続中のトランザクションに属するメッセージの後に表示されるメッセージは、関連するトランザクションが完了するまで保留されません。その結果、**read_committed** コンシューマーはフライトトランザクションがある場合に高基準値を読み取ることができません。

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

max.poll.interval.ms

型: int

デフォルト: 300000 (5分)

有効な値: [1,...]

重要度: 中

コンシューマーグループ管理を使用する場合の `poll()` の呼び出し間の最大遅延。これにより、コンシューマーがさらにレコードをフェッチする前にアイドル状態になることができる時間に上限が設定されます。このタイムアウトの期限が切れる前に `poll()` が呼び出されない場合、コンシューマーは失敗とみなされ、グループはパーティションを別のメンバーに再割り当てするためにリバランスします。null 以外の **group.instance.id** を使用しているコンシューマーがこのタイムアウトに達した

場合、パーティションは即座に再割り当てされません。代わりに、コンシューマーはハートビートの送信を停止し、**session.timeout.ms** の期限が切れるとパーティションが再割り当てされます。これは、シャットダウンした静的コンシューマーの動作を反映します。

max.poll.records

型: int

デフォルト: 500

有効な値: [1,...]

重要度: 中

poll() への単一の呼び出しで返される最大レコード数。**max.poll.records** は、基礎となるフェッチ動作には影響しません。コンシューマーは各フェッチリクエストからのレコードをキャッシュし、各ポーリングからインクリメンタルにこれらを返します。

partition.assignment.strategy

タイプ: list

デフォルト: class org.apache.kafka.clients.consumer.RangeAssignor, class org.apache.kafka.clients.consumer.CooperativeStickyAssignor

有効な値: non-null string

重要度: 中

グループ管理が使用されている場合に、クライアントがコンシューマーインスタンス間でパーティションの所有権を分散するために使用する、サポートされているパーティション割り当て戦略のクラス名またはクラスタイプのリスト (優先度順)。利用可能なオプションは以下の通りです。

- **org.apache.kafka.clients.consumer.RangeAssignor**: トピックごとにパーティションを割り当てます。
- **org.apache.kafka.clients.consumer.RoundRobinAssignor**: パーティションをラウンドロビン方式でコンシューマーに割り当てます。
- **org.apache.kafka.clients.consumer.StickyAssignor**: できるだけ多くの既存のパーティション割り当てを保持しながら、最大限にバランスの取れた割り当てを保証します。
- **org.apache.kafka.clients.consumer.CooperativeStickyAssignor**: 同じ StickyAssignor ロジックに従いますが、Cooperative Rebalancing を許可します。
デフォルトのアサイナーは [RangeAssignor, CooperativeStickyAssignor] で、デフォルトで RangeAssignor を使用しますが、リストから RangeAssignor を削除する 1 回のローリングバウンズで CooperativeStickyAssignor にアップグレードできます。

org.apache.kafka.clients.consumer.ConsumerPartitionAssignor インターフェイスを実装すると、カスタム割り当てストラテジーでプラグインできます。

receive.buffer.bytes

型: int

デフォルト: 65536 (64 キビバイト)

有効な値: [-1,...]

重要度: 中

データの読み取り時に使用する TCP 受信バッファー (SO_RCVBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

request.timeout.ms

型: int

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 中

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

sasl.client.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL クライアントコールバックハンドラークラスの完全修飾名。

sasl.jaas.config

タイプ: password

デフォルト: null

重要度: 中

JAAS 設定ファイルで使用される形式の SASL 接続の JAAS ログインコンテキストパラメーター。JAAS 設定ファイルの形式は、[こちら](#) で説明されています。値の形式は **loginModuleClass controlFlag (optionName=optionValue)***; です。ブローカーの場合、設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;` などです。

sasl.kerberos.service.name

タイプ: string

デフォルト: null

重要度: 中

Kafka が実行される Kerberos プリンシパル名。これは、Kafka の JAAS 設定または Kafka の設定で定義できます。

sasl.login.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL ログインコールバックハンドラークラスの完全修飾名。ブローカーの場合、ログインコールバックハンドラー設定の前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler` などです。

sasl.login.class

タイプ: class

デフォルト: null

重要度: 中

Login インターフェイスを実装するクラスの完全修飾名。ブローカーの場合、ログイン設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin` です。

sasl.mechanism

型: string

デフォルト: GSSAPI

重要度: 中

クライアント接続に使用される SASL メカニズム。これは、セキュリティープロバイダーが利用可能な任意のメカニズムである可能性があります。GSSAPI はデフォルトのメカニズムです。

sasl.oauthbearer.jwks.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

プロバイダーの JWKS (JSON Web Key Set) を取得できる OAuth/OIDC プロバイダーの URL。URL は、HTTP(S) ベースまたはファイルベースにすることができます。URL が HTTP(S) ベースの場合、JWKS データは、ブローカーの起動時に設定された URL を介して OAuth/OIDC プロバイダーから取得されます。その時点で最新のすべてのキーは、受信リクエストのためにブローカーにキャッシュされます。まだキャッシュにない kid ヘッダークレーム値を含む JWT の認証リクエストを受信した場合、JWKS エンドポイントはオンデマンドで再度クエリーされます。ただし、ブローカーは、sasl.oauthbearer.jwks.endpoint.refresh.ms ミリ秒ごとに URL をポーリングして、それらを含む JWT リクエストが受信される前に、今後のキーでキャッシュを更新します。URL がファイルベースの場合、ブローカーは起動時に設定された場所から JWKS ファイルをロードします。JWT に JWKS ファイルにない kid ヘッダー値が含まれている場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.token.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

OAuth/OIDCID プロバイダーの URL。URL が HTTP(S) ベースの場合、sasl.jaas.config の設定に基づいてログインリクエストが行われるのは、issuer のトークンエンドポイント URL です。URL がファイルベースの場合、認証に使用するために OAuth/OIDC ID プロバイダーによって発行されたアクセストークン (JWT シリアル化形式) を含むファイルを指定します。

security.protocol

型: string

デフォルト: PLAINTEXT

重要度: 中

ブローカーとの通信に使用されるプロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。

send.buffer.bytes

型: int

デフォルト: 131072 (128 キビバイト)

有効な値: [-1,...]

重要度: 中

データの送信時に使用する TCP 送信バッファ (SO_SNDBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

socket.connection.setup.timeout.max.ms

型: long

デフォルト: 30000 (30 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する最大時間。接続設定のタイムアウトは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。接続ストームを回避するために、タイムアウトに 0.2 のランダム化ファクターが適用され、計算された値よりも 20% 未満と 20% 超の間にランダムな範囲が発生します。

socket.connection.setup.timeout.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する時間。タイムアウトが経過する前に接続が構築されない場合、クライアントはソケットチャネルを閉じます。

ssl.enabled.protocols

型: list

デフォルト: TLSv1.2、TLSv1.3

重要度: 中

SSL 接続で有効なプロトコルの一覧。Java 11 以降で実行する場合、デフォルトは 'TLSv1.2,TLSv1.3' で、それ以外の場合は 'TLSv1.2' になります。Java 11 のデフォルト値では、クライアントとサーバーの両方が TLSv1.3 をサポートする場合は TLSv1.3 を優先し、そうでない場合は TLSv1.2 にフォールバックします (両方が少なくとも TLSv1.2 をサポートすることを前提とします)。ほとんどの場合、このデフォルトは問題ありません。**ssl.protocol** の設定ドキュメントも参照してください。

ssl.keystore.type

型: string

デフォルト: JKS

重要度: 中

キーストアファイルのファイル形式。これはクライアントにとってオプションになります。

ssl.protocol

型: string

デフォルト: TLSv1.3

重要度: 中

SSLContext の生成に使用される SSL プロトコル。Java 11 以降で実行する場合、デフォルトは 'TLSv1.3' になります。それ以外の場合は 'TLSv1.2' になります。この値は、ほとんどのユースケースで問題ありません。最近の JVM で許可される値は 'TLSv1.2' および 'TLSv1.3' です。'TLS'、'TLSv1.1'、'SSL'、'SSLv2'、および 'SSLv3' は古い JVM でサポートされる可能性がありますが、既知のセキュリティ脆弱性のために使用は推奨されません。この設定のデフォルト値および 'ssl.enabled.protocols' では、サーバーが 'TLSv1.3' に対応していない場合は、クライアントは 'TLSv1.2' にダウングレードされます。この設定が 'TLSv1.2' に設定されている場合、'TLSv1.3' が ssl.enabled.protocols の値の1つで、サーバーが 'TLSv1.3' のみをサポートする場合でも、クライアントは 'TLSv1.3' を使用しません。

ssl.provider

タイプ: string

デフォルト: null

重要度: 中

SSL 接続に使用されるセキュリティプロバイダーの名前。デフォルト値は JVM のデフォルトのセキュリティプロバイダーです。

ssl.truststore.type

型: string

デフォルト: JKS

重要度: 中

トラストストアファイルのファイル形式。

auto.commit.interval.ms

型: int

デフォルト: 5000 (5 秒)

有効な値: [0,...]

重要度: 低

`enable.auto.commit` が `true` に設定されている場合にコンシューマーオフセットが Kafka に自動コミットされる頻度 (ミリ秒単位)。

check.crcs

タイプ: boolean

デフォルト: true

重要度: 低

消費されたレコードの CRC32 を自動的に確認します。これにより、メッセージのネットワーク上またはディスク上の破損が発生しなくなります。このチェックはオーバーヘッドを追加するため、極端なパフォーマンスを求める場合は無効になる可能性があります。

client.id

タイプ: string

デフォルト: ""

重要度: 低

要求の実行時にサーバーに渡す ID 文字列。この目的は、サーバー側の要求ロギングに論理アプリケーション名を含めることを許可することにより、ip/port の他にもリクエストのソースを追跡できるようにすることです。

client.rack

タイプ: string

デフォルト: ""

重要度: 低

このクライアントのラック識別子。これには、このクライアントが物理的に存在する場所を示す任意の文字列値を指定できます。ブローカー設定 'broker.rack' に対応します。

fetch.max.wait.ms

型: int

デフォルト: 500

有効な値: [0,...]

重要度: 低

`fetch.min.bytes` で指定された要件をすぐに満たすために十分なデータがない場合に、サーバーがフェッチリクエストに応答するまでにブロックする最大時間。

interceptor.classes

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 低

インターセプターとして使用するクラスの一

覧。`org.apache.kafka.clients.consumer.ConsumerInterceptor` インターフェイスを実装すると、コンシューマーによって受信される (場合によっては変更) レコードを傍受できます。デフォルトでは、インターセプターはありません。

metadata.max.age.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [0,...]

重要度: low

新しいブローカーまたはパーティションをプロアクティブに検出するためのパーティションリーダーシップの変更がない場合でも、メタデータの更新を強制するまでの期間 (ミリ秒単位)。

metric.reporters

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 低

メトリクスレポーターとして使用するクラスの一

覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 低

メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string

デフォルト: INFO

有効な値: [INFO, DEBUG, TRACE]

重要度: 低

メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

メトリクスサンプルが計算される時間枠。

reconnect.backoff.max.ms

型: long

デフォルト: 1000 (1 秒)

有効な値: [0,...]

重要度: 低

接続に繰り返し失敗したブローカーへの再接続時に待機する最大時間 (ミリ秒単位)。これが指定されている場合、ホストごとのバックオフは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。バックオフの増加を計算した後、コネクションストームを回避するために 20% のランダムなジッターが追加されます。

reconnect.backoff.ms

型: long

デフォルト: 50

有効な値: [0,...]

重要度: 低

特定のホストへの再接続を試みる前に待機するベース時間。これにより、タイトなループでホストに繰り返し接続することを回避します。このバックオフは、クライアントによるブローカーへのすべての接続試行に適用されます。

retry.backoff.ms

型: long

デフォルト: 100

有効な値: [0,...]

重要度: 低

特定のトピックパーティションに対して失敗したリクエストを再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

sasl.kerberos.kinit.cmd

タイプ: string

デフォルト: /usr/bin/kinit

重要度: 低

Kerberos kinit コマンドパス。

sasl.kerberos.min.time.before.relogin

型: long

デフォルト: 60000

重要度: 低

更新試行間のログインスレッドのスリープ時間。

sasl.kerberos.ticket.renew.jitter

型: double

デフォルト: 0.05

重要度: 低

更新時間に追加されたランダムなジッターの割合。

sasl.kerberos.ticket.renew.window.factor

型: double

デフォルト: 0.8

重要度: 低

ログインスレッドは、最後の更新からチケットの有効期限までの指定された時間のウィンドウファクターに達するまでスリープし、その時点でチケットの更新を試みます。

sasl.login.connect.timeout.ms

タイプ: int

デフォルト: null

重要度: 低

外部認証プロバイダーの接続タイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.read.timeout.ms

タイプ: int

デフォルト: null

重要度: 低

外部認証プロバイダーの読み取りタイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.buffer.seconds

型: short

デフォルト: 300

有効な値: [0,...,3600]

重要度: 低

クレデンシャルを更新するときに維持するクレデンシャルの有効期限が切れるまでのバッファ時間 (秒単位)。更新が、バッファ秒数よりも有効期限に近いときに発生する場合、更新は、可能な限り多くのバッファ時間を維持するために上に移動されます。設定可能な値は 0 から 3600 (1 時間) です。値を指定しない場合は、デフォルト値の 300 (5 分) が使用されます。この値および `sasl.login.refresh.min.period.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合にどちらも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.min.period.seconds

型: short

デフォルト: 60

有効な値: [0,...,900]

重要度: 低

ログイン更新スレッドがクレデンシャルを更新する前に待機する最小時間 (秒単位)。設定可能な値は 0 から 900 (15 分) です。値を指定しない場合は、デフォルト値の 60 (1 分) が使用されます。この値および `sasl.login.refresh.buffer.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合に両方とも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.factor

型: double

デフォルト: 0.8

有効な値: [0.5,...,1.0]

重要度: 低

ログイン更新スレッドは、クレデンシャルの有効期間との関連で指定の期間ファクターに達するまでスリープ状態になり、達成した時点でクレデンシャルの更新を試みます。設定可能な値は 0.5(50%) から 1.0(100%) までです。値が指定されていない場合、デフォルト値の 0.8(80%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.jitter

型: double

デフォルト: 0.05

有効な値: [0.0,...,0.25]

重要度: 低

ログイン更新スレッドのスリープ時間に追加されるクレデンシャルの存続期間に関連するランダムジッターの最大量。設定可能な値は 0 から 0.25(25%) です。値が指定されていない場合は、デフォルト値の 0.05(5%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.max.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーへのログイン試行間の最大待機時間のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms` 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.ms

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーへのログイン試行間の最初の待機のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms`

設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、`OAUTHBEARER` にのみ適用されます。

`sasl.oauthbearer.clock.skew.seconds`

型: int

デフォルト: 30

重要度: 低

OAuth/OIDC ID プロバイダーとブローカーの時間の差を考慮した秒単位の (オプションの) 値。

`sasl.oauthbearer.expected.audience`

タイプ: list

デフォルト: null

重要度: 低

JWT が想定される対象者の 1 つに対して発行されたことを確認するためにブローカーが使用する (オプションの) コンマ区切りの設定です。JWT は標準的な OAuth の `aud` クレームについて検査され、この値が設定されている場合、ブローカーは JWT の `aud` クレームから値が完全に一致するかどうかを確認するために照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

`sasl.oauthbearer.expected.issuer`

タイプ: string

デフォルト: null

重要度: 低

ブローカーが JWT が想定される issuer によって作成されたことを確認するために使用する (オプションの) 設定。JWT は標準の OAuth `iss` クレームについて検査され、この値が設定されている場合、ブローカーは JWT の `iss` クレームにあるものと正確に照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

`sasl.oauthbearer.jwks.endpoint.refresh.ms`

型: long

デフォルト: 3600000 (1 時間)

重要度: 低

ブローカーが JWT の署名を検証するためのキーを含む JWKS (JSON Web Key Set) キャッシュを更新するまで待機するミリ秒単位の (オプションの) 値。

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

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーから JWKS (JSON Web Key Set) を取得する試行間の最大待機時間のミリ秒単位の (オプションの) 値。JWKS の取得では、`sasl.oauthbearer.jwks.endpoint.retry.backoff.ms` 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、`sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms` 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

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

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーからの JWKS (JSON Web Key Set) 取得する試行間の最初の待機のミリ秒単位の (オプションの) 値。JWKS の取得では、`sasl.oauthbearer.jwks.endpoint.retry.backoff.ms` 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、

`sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms` 設定で指定された最大待機時間まで試行間の待機時間が2倍になります。

`sasl.oauthbearer.scope.claim.name`

Type: 文字列

Type: スコープ

Importance: 低

スコープの OAuth クレームはスコープと呼ばれることがよくありますが、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、この (オプションの) 設定は JWT ペイロードのクレームに含まれるスコープに使用する別の名前を提供できます。

`sasl.oauthbearer.sub.claim.name`

Type: 文字列

Default: sub

Importance: 低

サブジェクトの OAuth クレームは sub と呼ばれることがよくありますが、この (オプションの) 設定は、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、JWT ペイロードのクレームに含まれるサブジェクトに使用する別の名前を提供できます。

`security.providers`

タイプ: string

デフォルト: null

重要度: 低

設定可能なクリエータークラスのリストで、それぞれがセキュリティーアルゴリズムを実装するプロバイダーを返します。これらのクラスは

`org.apache.kafka.common.security.auth.SecurityProviderCreator` インターフェイスを実装する必要があります。

`ssl.cipher.suites`

タイプ: list

デフォルト: null

重要度: 低

暗号化スイートの一覧。これは、TLS または SSL ネットワークプロトコルを使用してネットワーク接続のセキュリティー設定をネゴシエートするために使用される認証、暗号化、MAC、および鍵交換アルゴリズムの名前付きの組み合わせです。デフォルトでは、利用可能なすべての暗号スイートがサポートされます。

`ssl.endpoint.identification.algorithm`

タイプ: string

デフォルト: https

重要度: 低

サーバー証明書を使用してサーバーのホスト名を検証するエンドポイント識別アルゴリズム。

`ssl.engine.factory.class`

タイプ: class

デフォルト: null

重要度: 低

SSL Engine オブジェクトを提供する `org.apache.kafka.common.security.auth.SslEngineFactory` タイプのクラス。デフォルト値は `org.apache.kafka.common.security.ssl.DefaultSslEngineFactory` です。

`ssl.keymanager.algorithm`

型: string

デフォルト: SunX509

重要度: 低

SSL 接続のキーマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたキーマネージャーファクトリーアルゴリズムです。

ssl.secure.random.implementation

タイプ: string

デフォルト: null

重要度: 低

SSL 暗号操作に使用する SecureRandom PRNG 実装。

ssl.trustmanager.algorithm

型: string

デフォルト: PKIX

重要度: 低

SSL 接続のトラストマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたトラストマネージャーファクトリーアルゴリズムです。

第4章 プロデューサー設定プロパティー

key.serializer

タイプ: class

重要度: 高

org.apache.kafka.common.serialization.Serializer インターフェイスを実装するキーのシリアライザークラス。

value.serializer

タイプ: class

重要度: 高

org.apache.kafka.common.serialization.Serializer インターフェイスを実装する値のシリアライザークラス。

bootstrap.servers

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 高

Kafka クラスターへの最初の接続を確立するために使用されるホストとポートのペアの一覧。クライアントは、ブートストラップ用にここで指定されたサーバーに関係なく、すべてのサーバーを利用します。この一覧は、サーバーのフルセットを検出するために使用される最初のホストにのみ影響します。この一覧は、**host1:port1,host2:port2,...** の形式にする必要があります。これらのサーバーは、(動的に変更される可能性がある) 完全なクラスターメンバーシップを検出するための最初の接続にだけ使用されるため、このリストにはサーバーの完全なセットを含める必要はありません(ただし、サーバーがダウンした場合に備えて、複数のサーバーが必要になる場合があります)。

buffer.memory

型: long

デフォルト: 33554432

有効な値: [0,...]

重要度: 高

プロデューサーが、サーバーへの送信を待機しているレコードをバッファリングするために使用できるメモリーの合計バイト数。レコードの送信がサーバーへの配信よりも速い場合、プロデューサーは **max.block.ms** の間ブロックし、その後例外を出力します。

この設定は、プロデューサーが使用する合計メモリーにおおまかに対応する必要がありますが、プロデューサーが使用するすべてのメモリーがバッファリングに使用されるわけではないため、ハードバウンドではありません。一部の追加メモリーは、圧縮(圧縮が有効な場合)やインフライトリクエストの維持に使用されます。

compression.type

タイプ: string

デフォルト: none

重要度: 高

プロデューサーによって生成されたすべてのデータの圧縮タイプ。デフォルトは none (圧縮なし) です。有効な値は、**none**、**gzip**、**snappy**、**lz4**、または **zstd** です。圧縮はデータの完全なバッチであるため、バッチ処理の有効性は圧縮率にも影響します(バッチ処理が多いほど圧縮率が高くなります)。

retries

型: int

デフォルト: 2147483647

有効な値: [0,...,2147483647]

重要度: 高

ゼロより大きい値を設定すると、クライアントは、一時的なエラーの可能性により送信に失敗したレコードを再送信します。この再試行は、クライアントがエラーを受信したときにレコードを再送信した場合と同じであることに注意してください。**delivery.timeout.ms** によって設定されたタイムアウトが成功確認の前に最初に期限切れになる場合、プロデュース要求は再試行回数が尽きる前に失敗します。通常は、この設定は未設定のままにし、代わりに **delivery.timeout.ms** を使用して再試行動作を制御します。

べき等性を有効にするには、この設定値を 0 より大きくする必要があります。競合する設定が設定されており、べき等が明示的に有効になっていない場合、べき等は無効になります。

enable.idempotence を **false** に設定し、**max.in.flight.requests.per.connection** を 1 に設定して再試行を許可すると、レコードの順序が変更される可能性があります。2 番目のバッチが成功すると、2 番目のバッチのレコードが最初に表示される場合があります。

ssl.key.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルの秘密鍵または 'ssl.keystore.key' で指定した PEM キーのパスワード。これは、双方向認証が設定されている場合にのみ、クライアントに必要です。

ssl.keystore.certificate.chain

タイプ: password

デフォルト: null

重要度: 高

'ssl.keystore.type' で指定された形式の証明書チェーン。デフォルトの SSL エンジンファクトリーは、X.509 証明書の一覧を含む PEM 形式のみをサポートします。

ssl.keystore.key

タイプ: password

デフォルト: null

重要度: 高

ssl.keystore.type で指定された形式の秘密鍵。デフォルトの SSL エンジンファクトリーは、PKCS#8 キーを持つ PEM 形式のみをサポートします。鍵が暗号化されている場合は、'ssl.key.password' を使用して鍵のパスワードを指定する必要があります。

ssl.keystore.location

タイプ: string

デフォルト: null

重要度: 高

キーストアファイルの場所。これはクライアントではオプションで、クライアントの双方向認証に使用できます。

ssl.keystore.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルのストアパスワード。これはクライアントではオプションで、'ssl.keystore.location' が設定されている場合にのみ必要です。キーストアのパスワードは PEM 形式ではサポートされません。

ssl.truststore.certificates

タイプ: password

デフォルト: null

重要度: 高

'ssl.truststore.type' で指定された形式の信頼できる証明書。デフォルトの SSL エンジンファクトリーは、X.509 証明書を使用する PEM 形式のみをサポートします。

ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 高

トラストストアファイルの場所。

ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 高

トラストストアファイルのパスワード。パスワードが設定されていない場合、設定されたトラストストアファイルは引き続き使用されますが、整合性チェックは無効になります。トラストストアのパスワードは PEM 形式ではサポートされません。

batch.size

型: int

デフォルト: 16384

有効な値: [0,...]

重要度: 中

複数のレコードが同じパーティションに送信される場合は常に、プロデューサーはレコードをまとめてより少ない要求にバッチ処理しようとします。これにより、クライアントとサーバーの両方でパフォーマンスが向上します。この設定では、デフォルトのバッチサイズをバイト単位で制御します。

このサイズより大きいレコードをバッチ処理することはありません。

ブローカーに送信されるリクエストには、複数のバッチが含まれます (送信可能なデータがあるパーティションごとに1つ)。

バッチサイズを小さくすると、バッチ処理が少なくなり、スループットが低下する可能性があります (バッチサイズがゼロの場合、バッチ処理は完全に無効になります)。バッチサイズが非常に大きい場合は、追加のレコードを想定して、常に指定のバッチサイズのバッファを割り当てるため、メモリーを多少無駄に使用する可能性があります。

注意: この設定は、送信されるバッチサイズの上限を示します。このパーティションに蓄積されたバイト数がこれより少ない場合は、**linger.ms** の時間、より多くのレコードが表示されるのを待つために待機することになります。この **linger.ms** のデフォルトは 0 であり、累積バッチサイズがこの **batch.size** の設定値以下であっても、すぐにレコードを送信することを意味します。

client.dns.lookup

タイプ: string

デフォルト: use_all_dns_ips

有効な値: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]

重要度: 中

クライアントが DNS ルックアップを使用する方法を制御します。**use_all_dns_ips** に設定すると、正常な接続が確立されるまで、返された各 IP アドレスに順番に接続します。切断後に、次の IP が

使用されます。すべての IP が一度使用されると、クライアントはホスト名から IP を再度解決します (ただし、JVM と OS の両方は DNS 名の検索をキャッシュします)。**resolve_canonical_bootstrap_servers_only** に設定すると、各ブートストラップアドレスを正規名の一覧に解決します。ブートストラップフェーズ後、これは **use_all_dns_ips** と同じように動作します。

client.id

タイプ: string

デフォルト: ""

重要度: 中

要求の実行時にサーバーに渡す ID 文字列。この目的は、サーバー側の要求ロギングに論理アプリケーション名を含めることを許可することにより、ip/port の他にもリクエストのソースを追跡できるようにすることです。

connections.max.idle.ms

型: long

デフォルト: 540000 (9 分)

重要度: 中

この設定で指定された期間 (ミリ秒単位) の後にアイドル状態の接続を閉じます。

delivery.timeout.ms

型: int

デフォルト: 120000 (2 分)

有効な値: [0,...]

重要度: 中

send() の呼び出しが返された後、成功または失敗を報告する時間の上限。これにより、送信前にレコードが遅延する合計時間、ブローカーから確認応答を待つ時間 (予想される場合)、および再試行可能な送信の失敗に許容される時間が制限されます。プロデューサーは、回復不可能なエラーが発生した場合や再試行し尽くした場合、またはレコードが早い配信有効期限に達したバッチに追加された場合に、この設定よりも前にレコードの送信の失敗を報告する可能性があります。この設定の値は、**request.timeout.ms** および **linger.ms** の合計値以上である必要があります。

linger.ms

型: long

デフォルト: 0

有効な値: [0,...]

重要度: 中

プロデューサーは、リクエストの送信の間に到着したレコードを1つのバッチリクエストにグループ化します。通常、これは、レコードが送信できるよりも早く到着した場合に、負荷がかかった状態でのみ発生します。ただし、状況によっては、中程度の負荷がかかっている場合でも、クライアントがリクエストの数を減らしたい場合があります。この設定は、少量の人為的な遅延を追加することでこれを実現します。つまり、レコードをすぐに送信するのではなく、プロデューサーは指定された遅延まで待機して他のレコードを送信できるようにし、送信をまとめてバッチ処理できるようにします。これは、TCP の Nagle アルゴリズムに類似するものと考えられます。この設定は、バッチ処理の遅延の上限を提供します。あるパーティションで **batch.size** 相当のレコードを取得すると、この設定に関係なくすぐに送信されますが、このパーティションで蓄積されたバイト数がこれより少ない場合は、指定された時間の間、さらにレコードが取得されるのを待つこととなります。デフォルトは 0 (つまり遅延なし) に設定されます。たとえば、**linger.ms=5** を設定すると、送信されるリクエストの数が減りますが、負荷がない状態で送信されるレコードに最大 5 ミリ秒のレイテンシーが追加されます。

max.block.ms

型: long

デフォルト: 60000 (1分)

有効な値: [0,...]

重要度: 中

この設定は、**KafkaProducer's**

`send()`、`partitionsFor()`、`initTransactions()`、`sendOffsetsToTransaction()`、`commitTransaction()`、および `abortTransaction()` メソッドがブロックされる期間を制御します。`send()` の場合、このタイムアウトが、メタデータの取得とバッファの割り当ての両方の待ち時間の合計を制限します (ユーザーが提供するシリアライザーやパーティショナーでのブロックは、このタイムアウトにはカウントされません)。`partitionsFor()` の場合、このタイムアウトは、メタデータが利用できない場合の待ち時間を制限します。トランザクション関連のメソッドは常にブロックしますが、トランザクションコーディネーターが検出されなかった場合、またはタイムアウト内に応答しなかった場合は、タイムアウトになる可能性があります。

max.request.size

型: int

デフォルト: 1048576

有効な値: [0,...]

重要度: 中

リクエストの最大サイズ (バイト単位)。この設定により、プロデューサーが1回のリクエストで送信するレコードバッチの数が制限され、大量のリクエストが送信されないようになります。これは事実上、圧縮されていないレコードの最大バッチサイズの上限でもあります。サーバーには、レコードバッチサイズ (圧縮が有効な場合は圧縮後) に独自の上限があり、これとは異なる可能性があることに注意してください。

partitioner.class

タイプ: class

デフォルト: org.apache.kafka.clients.producer.internals.DefaultPartitioner

重要度: 中

レコードを生成するときに送信するパーティションを決定するために使用するクラス。利用可能なオプションは以下の通りです。

- **org.apache.kafka.clients.producer.internals.DefaultPartitioner**: デフォルトのパーティショナー。このストラテジーでは、バッチがいっぱいになるか、**linger.ms** が起動するまで、パーティションに固定しようとします。それはストラテジーで動作します:
- パーティションが指定されていないがキーが存在する場合は、キーのハッシュに基づいてパーティションを選択します
- パーティションまたはキーが存在しない場合は、バッチがいっぱいになるか、**linger.ms** が起動したときに変更される固定パーティションを選択します。
- **org.apache.kafka.clients.producer.RoundRobinPartitioner**: このパーティショニングストラテジーでは、一連の連続するレコードの各レコードは、パーティションが不足して最初からやり直すまで、(キーが提供されているかどうかに関係なく) 異なるパーティションに送信されます。注意: 新しいバッチを作成すると、分布が不均一になる既知の問題があります。詳細については、KAFKA-9965 を確認してください。
- **org.apache.kafka.clients.producer.UniformStickyPartitioner**: このパーティショニングストラテジーは、バッチがいっぱいになるか、**linger.ms** が起動するまで、(キーが提供されているかどうかに関係なく) パーティションに固定しようとします。
org.apache.kafka.clients.producer.Partitioner インターフェイスを実装すると、カスタムパーティショナーをプラグインできます。

receive.buffer.bytes

型: int

デフォルト: 32768 (32 キビバイト)

有効な値: [-1,...]

重要度: 中

データの読み取り時に使用する TCP 受信バッファー (SO_RCVBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

request.timeout.ms

型: int

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 中

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。これは、不必要なプロデューサーの再試行によるメッセージの重複の可能性を減らすために、**replica.lag.time.max.ms** (ブローカーの設定) よりも大きくする必要があります。

sasl.client.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL クライアントコールバックハンドラークラスの完全修飾名。

sasl.jaas.config

タイプ: password

デフォルト: null

重要度: 中

JAAS 設定ファイルで使用される形式の SASL 接続の JAAS ログインコンテキストパラメーター。JAAS 設定ファイルの形式は、[こちら](#) で説明されています。値の形式は **loginModuleClass controlFlag (optionName=optionValue)***; です。ブローカーの場合、設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;` などです。

sasl.kerberos.service.name

タイプ: string

デフォルト: null

重要度: 中

Kafka が実行される Kerberos プリンシパル名。これは、Kafka の JAAS 設定または Kafka の設定で定義できます。

sasl.login.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL ログインコールバックハンドラークラスの完全修飾名。ブローカーの場合、ログインコールバックハンドラー設定の前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler` などです。

sasl.login.class

タイプ: class

デフォルト: null

重要度: 中

Login インターフェイスを実装するクラスの完全修飾名。ブローカーの場合、ログイン設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin` です。

sasl.mechanism

型: string

デフォルト: GSSAPI

重要度: 中

クライアント接続に使用される SASL メカニズム。これは、セキュリティープロバイダーが利用可能な任意のメカニズムである可能性があります。GSSAPI はデフォルトのメカニズムです。

sasl.oauthbearer.jwks.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

プロバイダーの [JWKS \(JSON Web Key Set\)](#) を取得できる OAuth/OIDC プロバイダーの URL。URL は、HTTP(S) ベースまたはファイルベースにすることができます。URL が HTTP(S) ベースの場合、JWKS データは、ブローカーの起動時に設定された URL を介して OAuth/OIDC プロバイダーから取得されます。その時点で最新のすべてのキーは、受信リクエストのためにブローカーにキャッシュされます。まだキャッシュにない kid ヘッダークレーム値を含む JWT の認証リクエストを受信した場合、JWKS エンドポイントはオンデマンドで再度クエリーされます。ただし、ブローカーは、`sasl.oauthbearer.jwks.endpoint.refresh.ms` ミリ秒ごとに URL をポーリングして、それらを含む JWT リクエストが受信される前に、今後のキーでキャッシュを更新します。URL がファイルベースの場合、ブローカーは起動時に設定された場所から JWKS ファイルをロードします。JWT に JWKS ファイルにない kid ヘッダー値が含まれている場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.token.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

OAuth/OIDCID プロバイダーの URL。URL が HTTP(S) ベースの場合、`sasl.jaas.config` の設定に基づいてログインリクエストが行われるのは、`issuer` のトークンエンドポイント URL です。URL がファイルベースの場合、認証に使用するために OAuth/OIDC ID プロバイダーによって発行されたアクセストークン (JWT シリアル化形式) を含むファイルを指定します。

security.protocol

型: string

デフォルト: PLAINTEXT

重要度: 中

ブローカーとの通信に使用されるプロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。

send.buffer.bytes

型: int

デフォルト: 131072 (128 キビバイト)

有効な値: [-1,...]

重要度: 中

データの送信時に使用する TCP 送信バッファ (SO_SNDBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

socket.connection.setup.timeout.max.ms

型: long

デフォルト: 30000 (30 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する最大時間。接続設定のタイムアウトは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。接続ストリームを回避するために、タイムアウトに 0.2 のランダム化ファクターが適用され、計算された値よりも 20% 未満と 20% 超の間にランダムな範囲が発生します。

socket.connection.setup.timeout.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する時間。タイムアウトが経過する前に接続が構築されない場合、クライアントはソケットチャネルを閉じます。

ssl.enabled.protocols

型: list

デフォルト: TLSv1.2, TLSv1.3

重要度: 中

SSL 接続で有効なプロトコルの一覧。Java 11 以降で実行する場合、デフォルトは 'TLSv1.2,TLSv1.3' で、それ以外の場合は 'TLSv1.2' になります。Java 11 のデフォルト値では、クライアントとサーバーの両方が TLSv1.3 をサポートする場合は TLSv1.3 を優先し、そうでない場合は TLSv1.2 にフォールバックします (両方が少なくとも TLSv1.2 をサポートすることを前提とします)。ほとんどの場合、このデフォルトは問題ありません。**ssl.protocol** の設定ドキュメントも参照してください。

ssl.keystore.type

型: string

デフォルト: JKS

重要度: 中

キーストアファイルのファイル形式。これはクライアントにとってオプションになります。

ssl.protocol

型: string

デフォルト: TLSv1.3

重要度: 中

SSLContext の生成に使用される SSL プロトコル。Java 11 以降で実行する場合、デフォルトは 'TLSv1.3' になります。それ以外の場合は 'TLSv1.2' になります。この値は、ほとんどのユースケースで問題ありません。最近の JVM で許可される値は 'TLSv1.2' および 'TLSv1.3' です。'TLS'、'TLSv1.1'、'SSL'、'SSLv2'、および 'SSLv3' は古い JVM でサポートされる可能性がありますが、既知のセキュリティ脆弱性のために使用は推奨されません。この設定のデフォルト値および 'ssl.enabled.protocols' では、サーバーが 'TLSv1.3' に対応していない場合は、クライアントは 'TLSv1.2' にダウングレードされます。この設定が 'TLSv1.2' に設定されている場合、'TLSv1.3' が ssl.enabled.protocols の値の 1 つで、サーバーが 'TLSv1.3' のみをサポートする場合でも、クライアントは 'TLSv1.3' を使用しません。

ssl.provider

タイプ: string

デフォルト: null

重要度: 中

SSL 接続に使用されるセキュリティープロバイダーの名前。デフォルト値は JVM のデフォルトのセキュリティープロバイダーです。

ssl.truststore.type

型: string

デフォルト: JKS

重要度: 中

トラストストアファイルのファイル形式。

acks

タイプ: string

デフォルト: all

有効な値: [all, -1, 0, 1]

重要度: 低

リクエストが完了したと見なす前に、プロデューサーがリーダーに受け取ったことを要求する確認の数。これは、送信されるレコードの耐久性を制御します。以下の設定が許可されます。

- **acks=0** ゼロに設定すると、プロデューサーはサーバーからの確認応答を一切待ちません。レコードは直ちにソケットバッファーに追加され、送信済みと見なされます。この場合、サーバーがレコードを受信したかどうかは保証されず、**retries** の設定は有効になりません (クライアントは通常、失敗を知ることができないからです)。各レコードで返されるオフセットは、常に **-1** に設定されます。
- **acks=1** これは、リーダーはその記録をローカルログに書き込みますが、全フォロワーからの完全な承認を待たずに応答します。この場合、リーダーがレコードを確認した直後に失敗し、これがフォロワーがレコードを複製する前であった場合は、レコードは失われます。
- **acks=all** リーダーは同期しているレプリカの完全セットがレコードを確認するのを待ちます。これにより、少なくとも1つの In-Sync レプリカが動作している限り、レコードが失われないことが保証されます。これは利用可能な最強の保証になります。これは **acks=-1** 設定に相当します。

冪等性を有効にするには、この設定値を **all** にする必要があることに注意してください。競合する設定が設定されており、べき等が明示的に有効になっていない場合、べき等は無効になります。

enable.idempotence

タイプ: boolean

デフォルト: true

重要度: 低

'true' に設定すると、プロデューサーは、プロデューサーは各メッセージのコピーが1つだけストリームに書き込まれるようにします。'false' の場合、ブローカーの失敗などによるプロデューサーの再試行により、再試行されたメッセージの重複がストリームに書き込まれる可能性があります。べき等を有効にするには、**max.in.flight.requests.per.connection** は5以下 (メッセージの順序は任意の許容値で保持される) で、**retries** は0よりも大きくなければならず、**acks** は 'all' である必要がある点に注意してください。

競合する設定が設定されていない場合、べき等性はデフォルトで有効になっています。競合する設定が設定されており、べき等が明示的に有効になっていない場合、べき等は無効になります。べき等が明示的に有効化されていて、競合する設定が設定されている場合、**ConfigException** が出力されます。

interceptor.classes

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 低

インターセプターとして使用するクラスの一

覧。**org.apache.kafka.clients.producer.ProducerInterceptor** インターフェイスを実装すると、プロデューサーが受信したレコードが Kafka クラスターに公開される前に、そのレコードをインターセプト (場合によってはミューテーションも可能) できます。デフォルトでは、インターセプターはありません。

max.in.flight.requests.per.connection

型: int

デフォルト: 5

有効な値: [1,...]

重要度: 低

ブロックする前にクライアントが1つの接続で送信する確認されていないリクエストの最大数。この設定が1よりも大きい値に設定され、**enable.idempotence** が false に設定されている場合、再試行 (再試行が有効な場合) が原因で送信が失敗した後にメッセージが並べ替えられるリスクがあることに注意してください。さらに、冪等性を有効にするには、この設定値を5以下にする必要があります。競合する設定が設定されており、べき等が明示的に有効になっていない場合、べき等は無効になります。

metadata.max.age.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [0,...]

重要度: low

新しいブローカーまたはパーティションをプロアクティブに検出するためのパーティションリーダーシップの変更がない場合でも、メタデータの更新を強制するまでの期間 (ミリ秒単位)。

metadata.max.idle.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [5000,...]

重要度: low

アイドル状態のトピックのメタデータをプロデューサーがキャッシュする時間を制御します。トピックが最後に作成されてからの経過時間がメタデータのアイドル期間を超えた場合、トピックのメタデータは忘れられ、次にアクセスするとメタデータフェッチ要求が強制されます。

metric.reporters

タイプ: list

デフォルト: ""

有効な値: non-null string

重要度: 低

メトリクスレポーターとして使用するクラスの一

覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 低

メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string

デフォルト: INFO

有効な値: [INFO, DEBUG, TRACE]

重要度: 低

メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

メトリクスサンプルが計算される時間枠。

reconnect.backoff.max.ms

型: long

デフォルト: 1000 (1 秒)

有効な値: [0,...]

重要度: 低

接続に繰り返し失敗したブローカーへの再接続時に待機する最大時間 (ミリ秒単位)。これが指定されている場合、ホストごとのバックオフは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。バックオフの増加を計算した後、コネクションストームを回避するために 20% のランダムなジッターが追加されます。

reconnect.backoff.ms

型: long

デフォルト: 50

有効な値: [0,...]

重要度: 低

特定のホストへの再接続を試みる前に待機するベース時間。これにより、タイトなループでホストに繰り返し接続することを回避します。このバックオフは、クライアントによるブローカーへのすべての接続試行に適用されます。

retry.backoff.ms

型: long

デフォルト: 100

有効な値: [0,...]

重要度: 低

特定のトピックパーティションに対して失敗したリクエストを再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

sasl.kerberos.kinit.cmd

タイプ: string

デフォルト: /usr/bin/kinit

重要度: 低

Kerberos kinit コマンドパス。

sasl.kerberos.min.time.before.relogin

型: long

デフォルト: 60000

重要度: 低

更新試行間のログインスレッドのスリープ時間。

sasl.kerberos.ticket.renew.jitter

型: double

デフォルト: 0.05

重要度: 低

更新時間に追加されたランダムなジッターの割合。

sasl.kerberos.ticket.renew.window.factor

型: double

デフォルト: 0.8

重要度: 低

ログインスレッドは、最後の更新からチケットの有効期限までの指定された時間のウィンドウファクターに達するまでスリープし、その時点でチケットの更新を試みます。

sasl.login.connect.timeout.ms

タイプ: int

デフォルト: null

重要度: 低

外部認証プロバイダーの接続タイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.read.timeout.ms

タイプ: int

デフォルト: null

重要度: 低

外部認証プロバイダーの読み取りタイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.buffer.seconds

型: short

デフォルト: 300

有効な値: [0,...,3600]

重要度: 低

クレデンシャルを更新するときに維持するクレデンシャルの有効期限が切れるまでのバッファ時間 (秒単位)。更新が、バッファ秒数よりも有効期限に近いときに発生する場合、更新は、可能な限り多くのバッファ時間を維持するために上に移動されます。設定可能な値は 0 から 3600 (1 時間) です。値を指定しない場合は、デフォルト値の 300 (5 分) が使用されます。この値および `sasl.login.refresh.min.period.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合にどちらも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.min.period.seconds

型: short

デフォルト: 60

有効な値: [0,...,900]

重要度: 低

ログイン更新スレッドがクレデンシャルを更新する前に待機する最小時間 (秒単位)。設定可能な値

は 0 から 900 (15 分) です。値を指定しない場合は、デフォルト値の 60 (1 分) が使用されます。この値および `sasl.login.refresh.buffer.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合に両方とも無視されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.login.refresh.window.factor`

型: double

デフォルト: 0.8

有効な値: [0.5,...,1.0]

重要度: 低

ログイン更新スレッドは、クレデンシャルの有効期間との関連で指定の期間ファクターに達するまでスリープ状態になり、達成した時点でクレデンシャルの更新を試みます。設定可能な値は 0.5(50%) から 1.0(100%) までです。値が指定されていない場合、デフォルト値の 0.8(80%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.login.refresh.window.jitter`

型: double

デフォルト: 0.05

有効な値: [0.0,...,0.25]

重要度: 低

ログイン更新スレッドのスリープ時間に追加されるクレデンシャルの存続期間に関連するランダムジッターの最大量。設定可能な値は 0 から 0.25(25%) です。値が指定されていない場合は、デフォルト値の 0.05(5%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.login.retry.backoff.max.ms`

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーへのログイン試行間の最大待機時間のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms` 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.login.retry.backoff.ms`

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーへのログイン試行間の最初の待機のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms` 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.oauthbearer.clock.skew.seconds`

型: int

デフォルト: 30

重要度: 低

OAuth/OIDC ID プロバイダーとブローカーの時間の差を考慮した秒単位の (オプションの) 値。

`sasl.oauthbearer.expected.audience`

タイプ: list

デフォルト: null

重要度: 低

JWT が想定される対象者の 1 つに対して発行されたことを確認するためにブローカーが使用する (オ

プシヨンの) コンマ区切りの設定です。JWT は標準的な OAuth の aud クレームについて検査され、この値が設定されている場合、ブローカーは JWT の aud クレームから値が完全に一致するかどうかを確認するために照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.expected.issuer

タイプ: string

デフォルト: null

重要度: 低

ブローカーが JWT が想定される issuer によって作成されたことを確認するために使用する (オプションの) 設定。JWT は標準の OAuth iss クレームについて検査され、この値が設定されている場合、ブローカーは JWT の iss クレームにあるものと正確に照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.jwks.endpoint.refresh.ms

型: long

デフォルト: 3600000 (1 時間)

重要度: 低

ブローカーが JWT の署名を検証するためのキーを含む JWKS (JSON Web Key Set) キャッシュを更新するまで待機するミリ秒単位の (オプションの) 値。

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

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーから JWKS (JSON Web Key Set) を取得する試行間の最大待機時間のミリ秒単位の (オプションの) 値。JWKS の取得では、sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

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

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーからの JWKS (JSON Web Key Set) 取得する試行間の最初の待機のミリ秒単位の (オプションの) 値。JWKS の取得では、sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

sasl.oauthbearer.scope.claim.name

Type: 文字列

Type: スコープ

Importance: 低

スコープの OAuth クレームはスコープと呼ばれることがよくありますが、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、この (オプションの) 設定は JWT ペイロードのクレームに含まれるスコープに使用する別の名前を提供できます。

sasl.oauthbearer.sub.claim.name

Type: 文字列

Default: sub

Importance: 低

サブジェクトの OAuth クレームは sub と呼ばれることがよくありますが、この (オプションの) 設定は、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、JWT ペイロードのクレームに含まれるサブジェクトに使用する別の名前を提供できます。

security.providers

タイプ: string

デフォルト: null

重要度: 低

設定可能なクリエータークラスのリストで、それぞれがセキュリティーアルゴリズムを実装するプロバイダーを返します。これらのクラスは

org.apache.kafka.common.security.auth.SecurityProviderCreator インターフェイスを実装する必要があります。

ssl.cipher.suites

タイプ: list

デフォルト: null

重要度: 低

暗号化スイートの一覧。これは、TLS または SSL ネットワークプロトコルを使用してネットワーク接続のセキュリティー設定をネゴシエートするために使用される認証、暗号化、MAC、および鍵交換アルゴリズムの名前付きの組み合わせです。デフォルトでは、利用可能なすべての暗号スイートがサポートされます。

ssl.endpoint.identification.algorithm

タイプ: string

デフォルト: https

重要度: 低

サーバー証明書を使用してサーバーのホスト名を検証するエンドポイント識別アルゴリズム。

ssl.engine.factory.class

タイプ: class

デフォルト: null

重要度: 低

SSL Engine オブジェクトを提供する org.apache.kafka.common.security.auth.SslEngineFactory タイプのクラス。デフォルト値は org.apache.kafka.common.security.ssl.DefaultSslEngineFactory です。

ssl.keymanager.algorithm

型: string

デフォルト: SunX509

重要度: 低

SSL 接続のキーマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたキーマネージャーファクトリーアルゴリズムです。

ssl.secure.random.implementation

タイプ: string

デフォルト: null

重要度: 低

SSL 暗号操作に使用する SecureRandom PRNG 実装。

ssl.trustmanager.algorithm

型: string

デフォルト: PKIX

重要度: 低

SSL 接続のトラストマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたトラストマネージャーファクトリーアルゴリズムです。

transaction.timeout.ms

型: int

デフォルト: 60000 (1分)

重要度: 低

トランザクションコーディネーターが、進行中のトランザクションを積極的に中止する前に、プロデューサーからのトランザクション状態の更新を待つ最大時間 (ミリ秒)。この値がブローカーの `transaction.max.timeout.ms` 設定より大きい場合、リクエストは `InvalidTxnTimeoutException` エラーで失敗します。

transactional.id

タイプ: string

デフォルト: null

有効な値: non-empty string

重要度: 低

トランザクション配信に使用する `TransactionalId`。これにより、クライアントは、新しいトランザクションを開始する前に同じ `TransactionalId` を使用するトランザクションが完了したことを保証できるため、複数のプロデューサーセッションにまたがる信頼性セマンティクスが可能になります。`TransactionalId` が指定されていない場合、プロデューサーはべき等配信に制限されます。`TransactionalId` が設定されている場合は、暗黙的に `enable.idempotence` になります。デフォルトでは、`TransactionalId` は設定されていません。つまり、トランザクションは使用できません。なお、デフォルトでは、トランザクションには少なくとも3つのブローカーのクラスターが必要で、これは本番環境では推奨される設定です。開発環境では `transaction.state.log.replication.factor` 設定を調整して変更できます。

第5章 管理クライアントの設定プロパティー

bootstrap.servers

タイプ: list

重要度: 高

Kafka クラスターへの最初の接続を確立するために使用されるホストとポートのペアの一覧。クライアントは、ブートストラップ用にここで指定されたサーバーに関係なく、すべてのサーバーを利用します。この一覧は、サーバーのフルセットを検出するために使用される最初のホストにのみ影響します。この一覧は、**host1:port1,host2:port2,...** の形式にする必要があります。これらのサーバーは、(動的に変更される可能性がある) 完全なクラスターメンバーシップを検出するための最初の接続にだけ使用されるため、このリストにはサーバーの完全なセットを含める必要はありません (ただし、サーバーがダウンした場合に備えて、複数のサーバーが必要になる場合があります)。

ssl.key.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルの秘密鍵または 'ssl.keystore.key' で指定した PEM キーのパスワード。これは、双方向認証が設定されている場合にのみ、クライアントに必要です。

ssl.keystore.certificate.chain

タイプ: password

デフォルト: null

重要度: 高

'ssl.keystore.type' で指定された形式の証明書チェーン。デフォルトの SSL エンジンファクトリーは、X.509 証明書の一覧を含む PEM 形式のみをサポートします。

ssl.keystore.key

タイプ: password

デフォルト: null

重要度: 高

ssl.keystore.type で指定された形式の秘密鍵。デフォルトの SSL エンジンファクトリーは、PKCS#8 キーを持つ PEM 形式のみをサポートします。鍵が暗号化されている場合は、'ssl.key.password' を使用して鍵のパスワードを指定する必要があります。

ssl.keystore.location

タイプ: string

デフォルト: null

重要度: 高

キーストアファイルの場所。これはクライアントではオプションで、クライアントの双方向認証に使用できます。

ssl.keystore.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルのストアパスワード。これはクライアントではオプションで、'ssl.keystore.location' が設定されている場合にのみ必要です。キーストアのパスワードは PEM 形式ではサポートされません。

ssl.truststore.certificates

タイプ: password

デフォルト: null

重要度: 高

'ssl.truststore.type' で指定された形式の信頼できる証明書。デフォルトの SSL エンジンファクターは、X.509 証明書を使用する PEM 形式のみをサポートします。

ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 高

トラストストアファイルの場所。

ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 高

トラストストアファイルのパスワード。パスワードが設定されていない場合、設定されたトラストストアファイルは引き続き使用されますが、整合性チェックは無効になります。トラストストアのパスワードは PEM 形式ではサポートされません。

client.dns.lookup

タイプ: string

デフォルト: use_all_dns_ips

有効な値: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]

重要度: 中

クライアントが DNS ルックアップを使用する方法を制御します。**use_all_dns_ips** に設定すると、正常な接続が確立されるまで、返された各 IP アドレスに順番に接続します。切断後に、次の IP が使用されます。すべての IP が一度使用されると、クライアントはホスト名から IP を再度解決します (ただし、JVM と OS の両方は DNS 名の検索をキャッシュします)。**resolve_canonical_bootstrap_servers_only** に設定すると、各ブートストラップアドレスを正規名の一覧に解決します。ブートストラップフェーズ後、これは **use_all_dns_ips** と同じように動作します。

client.id

タイプ: string

デフォルト: ""

重要度: 中

要求の実行時にサーバーに渡す ID 文字列。この目的は、サーバー側の要求ロギングに論理アプリケーション名を含めることを許可することにより、ip/port の他にもリクエストのソースを追跡できるようにすることです。

connections.max.idle.ms

型: long

デフォルト: 300000 (5 分)

重要度: 中

この設定で指定された期間 (ミリ秒単位) の後にアイドル状態の接続を閉じます。

default.api.timeout.ms

型: int

デフォルト: 60000 (1 分)

有効な値: [0,...]

重要度: 中

クライアント API のタイムアウト (ミリ秒単位) を指定します。この設定は、**timeout** パラメーターを指定しないすべてのクライアント操作のデフォルトタイムアウトとして使用されます。

receive.buffer.bytes

型: int

デフォルト: 65536 (64 キビバイト)

有効な値: [-1,...]

重要度: 中

データの読み取り時に使用する TCP 受信バッファ (SO_RCVBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

request.timeout.ms

型: int

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 中

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

sasl.client.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL クライアントコールバックハンドラークラスの完全修飾名。

sasl.jaas.config

タイプ: password

デフォルト: null

重要度: 中

JAAS 設定ファイルで使用される形式の SASL 接続の JAAS ログインコンテキストパラメーター。JAAS 設定ファイルの形式は、[こちら](#)で説明されています。値の形式は **loginModuleClass controlFlag (optionName=optionValue)***; です。ブローカーの場合、設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;` などです。

sasl.kerberos.service.name

タイプ: string

デフォルト: null

重要度: 中

Kafka が実行される Kerberos プリンシパル名。これは、Kafka の JAAS 設定または Kafka の設定で定義できます。

sasl.login.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL ログインコールバックハンドラークラスの完全修飾名。ブローカーの場合、ログインコールバックハンドラー設定の前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、

listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler などです。

sasl.login.class

タイプ: class
デフォルト: null
重要度: 中

Login インターフェイスを実装するクラスの完全修飾名。ブローカーの場合、ログイン設定の前にリッスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin です。

sasl.mechanism

型: string
デフォルト: GSSAPI
重要度: 中

クライアント接続に使用される SASL メカニズム。これは、セキュリティープロバイダーが利用可能な任意のメカニズムである可能性があります。GSSAPI はデフォルトのメカニズムです。

sasl.oauthbearer.jwks.endpoint.url

タイプ: string
デフォルト: null
重要度: 中

プロバイダーの [JWKS \(JSON Web Key Set\)](#) を取得できる OAuth/OIDC プロバイダーの URL。URL は、HTTP(S) ベースまたはファイルベースにすることができます。URL が HTTP(S) ベースの場合、JWKS データは、ブローカーの起動時に設定された URL を介して OAuth/OIDC プロバイダーから取得されます。その時点で最新のすべてのキーは、受信リクエストのためにブローカーにキャッシュされます。まだキャッシュにない kid ヘッダークレーム値を含む JWT の認証リクエストを受信した場合、JWKS エンドポイントはオンデマンドで再度クエリーされます。ただし、ブローカーは、sasl.oauthbearer.jwks.endpoint.refresh.ms ミリ秒ごとに URL をポーリングして、それらを含む JWT リクエストが受信される前に、今後のキーでキャッシュを更新します。URL がファイルベースの場合、ブローカーは起動時に設定された場所から JWKS ファイルをロードします。JWT に JWKS ファイルにない kid ヘッダー値が含まれている場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.token.endpoint.url

タイプ: string
デフォルト: null
重要度: 中

OAuth/OIDCID プロバイダーの URL。URL が HTTP(S) ベースの場合、sasl.jaas.config の設定に基づいてログインリクエストが行われるのは、issuer のトークンエンドポイント URL です。URL がファイルベースの場合、認証に使用するために OAuth/OIDC ID プロバイダーによって発行されたアクセストークン (JWT シリアル化形式) を含むファイルを指定します。

security.protocol

型: string
デフォルト: PLAINTEXT
重要度: 中

ブローカーとの通信に使用されるプロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。

send.buffer.bytes

型: int

デフォルト: 131072 (128 キビバイト)

有効な値: [-1,...]

重要度: 中

データの送信時に使用する TCP 送信バッファ (SO_SNDBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

socket.connection.setup.timeout.max.ms

型: long

デフォルト: 30000 (30 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する最大時間。接続設定のタイムアウトは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。接続ストームを回避するために、タイムアウトに 0.2 のランダム化ファクターが適用され、計算された値よりも 20% 未満と 20% 超の間にランダムな範囲が発生します。

socket.connection.setup.timeout.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 中

ソケット接続が確立されるまでクライアントが待機する時間。タイムアウトが経過する前に接続が構築されない場合、クライアントはソケットチャンネルを閉じます。

ssl.enabled.protocols

型: list

デフォルト: TLSv1.2、TLSv1.3

重要度: 中

SSL 接続で有効なプロトコルの一覧。Java 11 以降で実行する場合、デフォルトは 'TLSv1.2,TLSv1.3' で、それ以外の場合は 'TLSv1.2' になります。Java 11 のデフォルト値では、クライアントとサーバーの両方が TLSv1.3 をサポートする場合は TLSv1.3 を優先し、そうでない場合は TLSv1.2 にフォールバックします (両方が少なくとも TLSv1.2 をサポートすることを前提とします)。ほとんどの場合、このデフォルトは問題ありません。ssl.protocol の設定ドキュメントも参照してください。

ssl.keystore.type

型: string

デフォルト: JKS

重要度: 中

キーストアファイルのファイル形式。これはクライアントにとってオプションになります。

ssl.protocol

型: string

デフォルト: TLSv1.3

重要度: 中

SSLContext の生成に使用される SSL プロトコル。Java 11 以降で実行する場合、デフォルトは 'TLSv1.3' になります。それ以外の場合は 'TLSv1.2' になります。この値は、ほとんどのユースケースで問題ありません。最近の JVM で許可される値は 'TLSv1.2' および 'TLSv1.3' です。'TLS'、'TLSv1.1'、'SSL'、'SSLv2'、および 'SSLv3' は古い JVM でサポートされる可能性がありますが、既知のセキュリティ脆弱性のために使用は推奨されません。この設定のデフォルト値および 'ssl.enabled.protocols' では、サーバーが 'TLSv1.3' に対応していない場合は、クライアントは 'TLSv1.2' にダウングレードされます。この設定が 'TLSv1.2' に設定されている場合、'TLSv1.3' が ssl.enabled.protocols の値の 1 つで、サーバーが 'TLSv1.3' のみをサポートする場合でも、クライアントは 'TLSv1.3' を使用しません。

ssl.provider

タイプ: string

デフォルト: null

重要度: 中

SSL 接続に使用されるセキュリティープロバイダーの名前。デフォルト値は JVM のデフォルトのセキュリティープロバイダーです。

ssl.truststore.type

型: string

デフォルト: JKS

重要度: 中

トラストストアファイルのファイル形式。

metadata.max.age.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [0,...]

重要度: low

新しいブローカーまたはパーティションをプロアクティブに検出するためのパーティションリーダーシップの変更がない場合でも、メタデータの更新を強制するまでの期間 (ミリ秒単位)。

metric.reporters

タイプ: list

デフォルト: ""

重要度: 低

メトリクスレポーターとして使用するクラスの一

覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 低

メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string

デフォルト: INFO

有効な値: [INFO, DEBUG, TRACE]

重要度: 低

メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

メトリクスサンプルが計算される時間枠。

reconnect.backoff.max.ms

型: long

デフォルト: 1000 (1 秒)

有効な値: [0,...]

重要度: 低

接続に繰り返し失敗したブローカーへの再接続時に待機する最大時間 (ミリ秒単位)。これが指定されている場合、ホストごとのバックオフは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。バックオフの増加を計算した後、コネクションストームを回避するために 20% のランダムなジッターが追加されます。

reconnect.backoff.ms

型: long

デフォルト: 50

有効な値: [0,...]

重要度: 低

特定のホストへの再接続を試みる前に待機するベース時間。これにより、タイトなループでホストに繰り返し接続することを回避します。このバックオフは、クライアントによるブローカーへのすべての接続試行に適用されます。

retries

型: int

デフォルト: 2147483647

有効な値: [0,...,2147483647]

重要度: 低

ゼロより大きい値を設定すると、クライアントは、一時的なエラーの可能性のある失敗した要求を再送信します。値をゼロまたは **MAX_VALUE** のいずれかに設定し、対応するタイムアウトパラメータを使用して、クライアントがリクエストを再試行する期間を制御することが推奨されます。

retry.backoff.ms

型: long

デフォルト: 100

有効な値: [0,...]

重要度: 低

失敗した要求を再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

sasl.kerberos.kinit.cmd

タイプ: string

デフォルト: /usr/bin/kinit

重要度: 低

Kerberos kinit コマンドパス。

sasl.kerberos.min.time.before.relogin

型: long

デフォルト: 60000

重要度: 低

更新試行間のログインスレッドのスリープ時間。

sasl.kerberos.ticket.renew.jitter

型: double
デフォルト: 0.05
重要度: 低
更新時間に追加されたランダムなジッターの割合。

sasl.kerberos.ticket.renew.window.factor

型: double
デフォルト: 0.8
重要度: 低
ログインスレッドは、最後の更新からチケットの有効期限までの指定された時間のウィンドウファクターに達するまでスリープし、その時点でチケットの更新を試みます。

sasl.login.connect.timeout.ms

タイプ: int
デフォルト: null
重要度: 低
外部認証プロバイダーの接続タイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.read.timeout.ms

タイプ: int
デフォルト: null
重要度: 低
外部認証プロバイダーの読み取りタイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.buffer.seconds

型: short
デフォルト: 300
有効な値: [0,...,3600]
重要度: 低
クレデンシャルを更新するときに維持するクレデンシャルの有効期限が切れるまでのバッファ時間 (秒単位)。更新が、バッファ秒数よりも有効期限に近いときに発生する場合、更新は、可能な限り多くのバッファ時間を維持するために上に移動されます。設定可能な値は 0 から 3600 (1 時間) です。値を指定しない場合は、デフォルト値の 300 (5 分) が使用されます。この値および `sasl.login.refresh.min.period.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合にどちらも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.min.period.seconds

型: short
デフォルト: 60
有効な値: [0,...,900]
重要度: 低
ログイン更新スレッドがクレデンシャルを更新する前に待機する最小時間 (秒単位)。設定可能な値は 0 から 900 (15 分) です。値を指定しない場合は、デフォルト値の 60 (1 分) が使用されます。この値および `sasl.login.refresh.buffer.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合に両方とも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.factor

型: double
デフォルト: 0.8
有効な値: [0.5,...,1.0]

重要度: 低

ログイン更新スレッドは、クレデンシャルの有効期間との関連で指定の期間ファクターに達するまでスリープ状態になり、達成した時点でクレデンシャルの更新を試みます。設定可能な値は 0.5(50%) から 1.0(100%) までです。値が指定されていない場合、デフォルト値の 0.8(80%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.jitter

型: double

デフォルト: 0.05

有効な値: [0.0,...,0.25]

重要度: 低

ログイン更新スレッドのスリープ時間に追加されるクレデンシャルの存続期間に関連するランダムジッターの最大量。設定可能な値は 0 から 0.25(25%) です。値が指定されていない場合は、デフォルト値の 0.05(5%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.max.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーへのログイン試行間の最大待機時間のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は sasl.login.retry.backoff.ms 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は sasl.login.retry.backoff.max.ms 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.ms

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーへのログイン試行間の最初の待機のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は sasl.login.retry.backoff.ms 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は sasl.login.retry.backoff.max.ms 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

sasl.oauthbearer.clock.skew.seconds

型: int

デフォルト: 30

重要度: 低

OAuth/OIDC ID プロバイダーとブローカーの時間の差を考慮した秒単位の (オプションの) 値。

sasl.oauthbearer.expected.audience

タイプ: list

デフォルト: null

重要度: 低

JWT が想定される対象者の 1 つに対して発行されたことを確認するためにブローカーが使用する (オプションの) コンマ区切りの設定です。JWT は標準的な OAuth の aud クレームについて検査され、この値が設定されている場合、ブローカーは JWT の aud クレームから値が完全に一致するかどうかを確認するために照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.expected.issuer

タイプ: string
デフォルト: null
重要度: 低

ブローカーが JWT が想定される issuer によって作成されたことを確認するために使用する (オプションの) 設定。JWT は標準の OAuth iss クレームについて検査され、この値が設定されている場合、ブローカーは JWT の iss クレームにあるものと正確に照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.jwks.endpoint.refresh.ms

型: long
デフォルト: 3600000 (1 時間)
重要度: 低

ブローカーが JWT の署名を検証するためのキーを含む JWKS (JSON Web Key Set) キャッシュを更新するまで待機するミリ秒単位の (オプションの) 値。

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

型: long
デフォルト: 10000 (10 秒)
重要度: 低

外部認証プロバイダーから JWKS (JSON Web Key Set) を取得する試行間の最大待機時間のミリ秒単位の (オプションの) 値。JWKS の取得では、sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

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

型: long
デフォルト: 100
重要度: 低

外部認証プロバイダーからの JWKS (JSON Web Key Set) 取得する試行間の最初の待機のミリ秒単位の (オプションの) 値。JWKS の取得では、sasl.oauthbearer.jwks.endpoint.retry.backoff.ms 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

sasl.oauthbearer.scope.claim.name

Type: 文字列
Type: スコープ
Importance: 低

スコープの OAuth クレームはスコープと呼ばれることがよくありますが、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、この (オプションの) 設定は JWT ペイロードのクレームに含まれるスコープに使用する別の名前を提供できます。

sasl.oauthbearer.sub.claim.name

Type: 文字列
Default: sub
Importance: 低

サブジェクトの OAuth クレームは sub と呼ばれることがよくありますが、この (オプションの) 設定は、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、JWT ペイロードのクレームに含まれるサブジェクトに使用する別の名前を提供できます。

security.providers

タイプ: string

デフォルト: null

重要度: 低

設定可能なクリエイタークラスのリストで、それぞれがセキュリティーアルゴリズムを実装するプロバイダーを返します。これらのクラスは

org.apache.kafka.common.security.auth.SecurityProviderCreator インターフェイスを実装する必要があります。

ssl.cipher.suites

タイプ: list

デフォルト: null

重要度: 低

暗号化スイートの一覧。これは、TLS または SSL ネットワークプロトコルを使用してネットワーク接続のセキュリティー設定をネゴシエートするために使用される認証、暗号化、MAC、および鍵交換アルゴリズムの名前付きの組み合わせです。デフォルトでは、利用可能なすべての暗号スイートがサポートされます。

ssl.endpoint.identification.algorithm

タイプ: string

デフォルト: https

重要度: 低

サーバー証明書を使用してサーバーのホスト名を検証するエンドポイント識別アルゴリズム。

ssl.engine.factory.class

タイプ: class

デフォルト: null

重要度: 低

SSL Engine オブジェクトを提供する org.apache.kafka.common.security.auth.SslEngineFactory タイプのクラス。デフォルト値は org.apache.kafka.common.security.ssl.DefaultSslEngineFactory です。

ssl.keymanager.algorithm

型: string

デフォルト: SunX509

重要度: 低

SSL 接続のキーマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたキーマネージャーファクトリーアルゴリズムです。

ssl.secure.random.implementation

タイプ: string

デフォルト: null

重要度: 低

SSL 暗号操作に使用する SecureRandom PRNG 実装。

ssl.trustmanager.algorithm

型: string

デフォルト: PKIX

重要度: 低

SSL 接続のトラストマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたトラストマネージャーファクトリーアルゴリズムです。

第6章 KAFKA CONNECT 設定プロパティ

config.storage.topic

タイプ: string

重要度: 高

コネクタ設定が保存される Kafka トピックの名前。

group.id

タイプ: string

重要度: 高

このワーカーが属する Connect クラスターグループを識別する一意の文字列。

key.converter

タイプ: class

重要度: 高

Kafka Connect 形式と Kafka に書き込まれるシリアル化された形式の間の変換に使用されるコンバータークラス。これは、Kafka に対して書き込みまたは読み取りされたメッセージのキーの形式を制御します。これはコネクタとは独立しているため、任意のコネクタが任意のシリアル化形式で動作することができます。一般的なフォーマットの例には、JSON や Avro があります。

offset.storage.topic

タイプ: string

重要度: 高

コネクタオフセットが保存される Kafka トピックの名前。

status.storage.topic

タイプ: string

重要度: 高

コネクタおよびタスクのステータスが保存される Kafka トピックの名前。

value.converter

タイプ: class

重要度: 高

Kafka Connect 形式と Kafka に書き込まれるシリアル化された形式の間の変換に使用されるコンバータークラス。これは、Kafka に対して書き込みまたは読み取りされたメッセージの値の形式を制御します。これはコネクタとは独立しているため、任意のコネクタが任意のシリアル化形式で動作することができます。一般的なフォーマットの例には、JSON や Avro があります。

bootstrap.servers

タイプ: list

デフォルト: localhost:9092

重要度: 高

Kafka クラスターへの最初の接続を確立するために使用されるホストとポートのペアの一覧。クライアントは、ブートストラップ用にここで指定されたサーバーに関係なく、すべてのサーバーを利用します。この一覧は、サーバーのフルセットを検出するために使用される最初のホストにのみ影響します。この一覧は、**host1:port1,host2:port2,...** の形式にする必要があります。これらのサーバーは、(動的に変更される可能性がある) 完全なクラスターメンバーシップを検出するための最初の接続にだけ使用されるため、このリストにはサーバーの完全なセットを含める必要はありません (ただし、サーバーがダウンした場合に備えて、複数のサーバーが必要になる場合があります)。

heartbeat.interval.ms

型: int

デフォルト: 3000 (3 秒)

重要度: 高

Kafka のグループ管理機能を使用する場合の、グループコーディネーターへのハートビート間の予想時間。ハートビートは、ワーカーのセッションがアクティブな状態を維持し、新しいメンバーがグループに参加したり離脱したりする際のリバランスを促進するために使用されます。この値は **session.timeout.ms** よりも低く設定する必要がありますが、通常はその値の 1/3 以下に設定する必要があります。さらに低く調整することで、通常のリバランスの予想時間を制御することもできます。

rebalance.timeout.ms

型: int

デフォルト: 60000 (1分)

重要度: 高

リバランスが開始された後、各ワーカーがグループに参加するための最大許容時間。これは基本的に、すべてのタスクが保留中のデータをフラッシュしてオフセットをコミットするために必要な時間の制限です。タイムアウトを超えると、ワーカーはグループから削除され、オフセットコミットの失敗が発生します。

session.timeout.ms

型: int

デフォルト: 10000 (10 秒)

重要度: 高

ワーカーの障害検出に使用されるタイムアウト。ワーカーは定期的にハートビートを送信し、liveness をブローカーに示します。このセッションタイムアウトの期限切れ前にブローカーによってハートビートが受信されない場合、ブローカーはグループからワーカーを削除し、リバランスを開始します。この値は、**group.min.session.timeout.ms** および **group.max.session.timeout.ms** によってブローカー設定で設定される許容範囲内である必要があることに注意してください。

ssl.key.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルの秘密鍵または 'ssl.keystore.key' で指定した PEM キーのパスワード。これは、双方向認証が設定されている場合にのみ、クライアントに必要です。

ssl.keystore.certificate.chain

タイプ: password

デフォルト: null

重要度: 高

'ssl.keystore.type' で指定された形式の証明書チェーン。デフォルトの SSL エンジンファクトリーは、X.509 証明書の一覧を含む PEM 形式のみをサポートします。

ssl.keystore.key

タイプ: password

デフォルト: null

重要度: 高

ssl.keystore.type で指定された形式の秘密鍵。デフォルトの SSL エンジンファクトリーは、PKCS#8 キーを持つ PEM 形式のみをサポートします。鍵が暗号化されている場合は、'ssl.key.password' を使用して鍵のパスワードを指定する必要があります。

ssl.keystore.location

タイプ: string

デフォルト: null

重要度: 高

キーストアファイルの場所。これはクライアントではオプションで、クライアントの双方向認証に使用できます。

ssl.keystore.password

タイプ: password

デフォルト: null

重要度: 高

キーストアファイルのストアパスワード。これはクライアントではオプションで、'ssl.keystore.location' が設定されている場合にのみ必要です。キーストアのパスワードは PEM 形式ではサポートされません。

ssl.truststore.certificates

タイプ: password

デフォルト: null

重要度: 高

'ssl.truststore.type' で指定された形式の信頼できる証明書。デフォルトの SSL エンジンファクトリーは、X.509 証明書を使用する PEM 形式のみをサポートします。

ssl.truststore.location

タイプ: string

デフォルト: null

重要度: 高

トラストストアファイルの場所。

ssl.truststore.password

タイプ: password

デフォルト: null

重要度: 高

トラストストアファイルのパスワード。パスワードが設定されていない場合、設定されたトラストストアファイルは引き続き使用されますが、整合性チェックは無効になります。トラストストアのパスワードは PEM 形式ではサポートされません。

client.dns.lookup

タイプ: string

デフォルト: use_all_dns_ips

有効な値: [use_all_dns_ips, resolve_canonical_bootstrap_servers_only]

重要度: 中

クライアントが DNS ルックアップを使用する方法を制御します。**use_all_dns_ips** に設定すると、正常な接続が確立されるまで、返された各 IP アドレスに順番に接続します。切断後に、次の IP が使用されます。すべての IP が一度使用されると、クライアントはホスト名から IP を再度解決します (ただし、JVM と OS の両方は DNS 名の検索をキャッシュします)。**resolve_canonical_bootstrap_servers_only** に設定すると、各ブートストラップアドレスを正規名の一覧に解決します。ブートストラップフェーズ後、これは **use_all_dns_ips** と同じように動作します。

connections.max.idle.ms

型: long

デフォルト: 540000 (9 分)

重要度: 中

この設定で指定された期間 (ミリ秒単位) の後にアイドル状態の接続を閉じます。

connector.client.config.override.policy

型: string

デフォルト: All

重要度: 中

ConnectorClientConfigOverridePolicy の実装のクラス名またはエイリアス。コネクタによってオーバーライドできるクライアント設定を定義します。デフォルトの実装は **All** です。つまり、コネクタ設定はすべてのクライアントプロパティをオーバーライドできます。フレームワークで可能な他のポリシーには、コネクタによるクライアントプロパティの上書きを許可しない **None** と、コネクタがクライアントプリンシパルのみを上書きできるようにする **Principal** が含まれません。

receive.buffer.bytes

型: int

デフォルト: 32768 (32 キビバイト)

有効な値: [0,...]

重要度: 中

データの読み取り時に使用する TCP 受信バッファ (SO_RCVBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

request.timeout.ms

型: int

デフォルト: 40000 (40 秒)

有効な値: [0,...]

重要度: 中

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

sasl.client.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL クライアントコールバックハンドラークラスの完全修飾名。

sasl.jaas.config

タイプ: password

デフォルト: null

重要度: 中

JAAS 設定ファイルで使用される形式の SASL 接続の JAAS ログインコンテキストパラメーター。JAAS 設定ファイルの形式は、[こちら](#) で説明されています。値の形式は **loginModuleClass controlFlag (optionName=optionValue)***; です。ブローカーの場合、設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;` などです。

sasl.kerberos.service.name

タイプ: string

デフォルト: null

重要度: 中

Kafka が実行される Kerberos プリンシパル名。これは、Kafka の JAAS 設定または Kafka の設定で定義できます。

sasl.login.callback.handler.class

タイプ: class

デフォルト: null

重要度: 中

AuthenticateCallbackHandler インターフェイスを実装する SASL ログインコールバックハンドラークラスの完全修飾名。ブローカーの場合、ログインコールバックハンドラー設定の前には、リスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler` などです。

sasl.login.class

タイプ: class

デフォルト: null

重要度: 中

Login インターフェイスを実装するクラスの完全修飾名。ブローカーの場合、ログイン設定の前にリスナー接頭辞と SASL メカニズム名を小文字で指定する必要があります。たとえば、`listener.name.sasl_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin` です。

sasl.mechanism

型: string

デフォルト: GSSAPI

重要度: 中

クライアント接続に使用される SASL メカニズム。これは、セキュリティープロバイダーが利用可能な任意のメカニズムである可能性があります。GSSAPI はデフォルトのメカニズムです。

sasl.oauthbearer.jwks.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

プロバイダーの [JWKS \(JSON Web Key Set\)](#) を取得できる OAuth/OIDC プロバイダーの URL。URL は、HTTP(S) ベースまたはファイルベースにすることができます。URL が HTTP(S) ベースの場合、JWKS データは、ブローカーの起動時に設定された URL を介して OAuth/OIDC プロバイダーから取得されます。その時点で最新のすべてのキーは、受信リクエストのためにブローカーにキャッシュされます。まだキャッシュにない kid ヘッダークレーム値を含む JWT の認証リクエストを受信した場合、JWKS エンドポイントはオンデマンドで再度クエリーされます。ただし、ブローカーは、`sasl.oauthbearer.jwks.endpoint.refresh.ms` ミリ秒ごとに URL をポーリングして、それらを含む JWT リクエストが受信される前に、今後のキーでキャッシュを更新します。URL がファイルベースの場合、ブローカーは起動時に設定された場所から JWKS ファイルをロードします。JWT に JWKS ファイルにない kid ヘッダー値が含まれている場合、ブローカーは JWT を拒否し、認証は失敗します。

sasl.oauthbearer.token.endpoint.url

タイプ: string

デフォルト: null

重要度: 中

OAuth/OIDCID プロバイダーの URL。URL が HTTP(S) ベースの場合、`sasl.jaas.config` の設定に基づいてログインリクエストが行われるのは、issuer のトークンエンドポイント URL です。URL がファイルベースの場合、認証に使用するために OAuth/OIDC ID プロバイダーによって発行されたアクセストークン (JWT シリアル化形式) を含むファイルを指定します。

security.protocol

型: string

デフォルト: PLAINTEXT

重要度: 中

ブローカーとの通信に使用されるプロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。

send.buffer.bytes

型: int

デフォルト: 131072 (128 キビバイト)

有効な値: [0,...]

重要度: 中

データの送信時に使用する TCP 送信バッファ (SO_SNDBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

ssl.enabled.protocols

型: list

デフォルト: TLSv1.2、TLSv1.3

重要度: 中

SSL 接続で有効なプロトコルの一覧。Java 11 以降で実行する場合、デフォルトは 'TLSv1.2,TLSv1.3' で、それ以外の場合は 'TLSv1.2' になります。Java 11 のデフォルト値では、クライアントとサーバーの両方が TLSv1.3 をサポートする場合は TLSv1.3 を優先し、そうでない場合は TLSv1.2 にフォールバックします (両方が少なくとも TLSv1.2 をサポートすることを前提とします)。ほとんどの場合、このデフォルトは問題ありません。**ssl.protocol** の設定ドキュメントも参照してください。

ssl.keystore.type

型: string

デフォルト: JKS

重要度: 中

キーストアファイルのファイル形式。これはクライアントにとってオプションになります。

ssl.protocol

型: string

デフォルト: TLSv1.3

重要度: 中

SSLContext の生成に使用される SSL プロトコル。Java 11 以降で実行する場合、デフォルトは 'TLSv1.3' になります。それ以外の場合は 'TLSv1.2' になります。この値は、ほとんどのユースケースで問題ありません。最近の JVM で許可される値は 'TLSv1.2' および 'TLSv1.3' です。'TLS'、'TLSv1.1'、'SSL'、'SSLv2'、および 'SSLv3' は古い JVM でサポートされる可能性がありますが、既知のセキュリティ脆弱性のために使用は推奨されません。この設定のデフォルト値および 'ssl.enabled.protocols' では、サーバーが 'TLSv1.3' に対応していない場合は、クライアントは 'TLSv1.2' にダウングレードされます。この設定が 'TLSv1.2' に設定されている場合、'TLSv1.3' が ssl.enabled.protocols の値の1つで、サーバーが 'TLSv1.3' のみをサポートする場合でも、クライアントは 'TLSv1.3' を使用しません。

ssl.provider

タイプ: string

デフォルト: null

重要度: 中

SSL 接続に使用されるセキュリティプロバイダーの名前。デフォルト値は JVM のデフォルトのセキュリティプロバイダーです。

ssl.truststore.type

型: string

デフォルト: JKS

重要度: 中

トラストストアファイルのファイル形式。

worker.sync.timeout.ms

型: int

デフォルト: 3000 (3 秒)

重要度: 中

ワーカーが他のワーカーと同期されず、設定を再同期する必要がある場合は、この時間まで待機してから、ギブアップし、グループから離脱し、バックオフ期間を待ってから再度参加します。

worker.unicync.backoff.ms

型: int

デフォルト: 300000 (5 分)

重要度: 中

ワーカーが他のワーカーと同期されず、worker.sync.timeout.ms 内で追いつくことができない場合は、再度参加する前に、この期間は Connect クラスタを離れます。

access.control.allow.methods

タイプ: string

デフォルト: ""

重要度: 低

Access-Control-Allow-Methods ヘッダーを設定することにより、クロスオリジン要求でサポートされるメソッドを設定します。Access-Control-Allow-Methods ヘッダーのデフォルト値は、GET、POST、および HEAD のクロスオリジン要求を許可します。

access.control.allow.origin

タイプ: string

デフォルト: ""

重要度: 低

REST API リクエスト用に Access-Control-Allow-Origin ヘッダーを設定する値。クロスオリジンアクセスを有効にするには、これを API へのアクセスを許可する必要があるアプリケーションのドメインに設定するか、'*' を設定して任意のドメインからのアクセスを許可します。デフォルト値は、REST API のドメインからのみアクセスを許可します。

admin.listeners

型: list

デフォルト: null

有効な値: コンマ区切りの URL (例: <http://localhost:8080>,<https://localhost:8443>)

重要度: 低

管理 REST API がリッスンするコンマ区切りの URI の一覧。サポートされるプロトコルは HTTP および HTTPS です。空または空白の文字列は、この機能を無効にします。デフォルトの動作では、通常のリッスナーを使用します ('listeners' プロパティで指定)。

client.id

タイプ: string

デフォルト: ""

重要度: 低

要求の実行時にサーバーに渡す ID 文字列。この目的は、サーバー側の要求ロギングに論理アプリケーション名を含めることを許可することにより、ip/port の他にもリクエストのソースを追跡できるようにすることです。

config.providers

タイプ: list

デフォルト: ""

重要度: 低

指定された順序でロードされ、使用される **ConfigProvider** クラスのコンマ区切りリスト。インターフェイス **ConfigProvider** を実装すると、外部化されたシークレットなどのコネクタ設定の変数参照を置き換えることができます。

config.storage.replication.factor

型: short

デフォルト: 3

有効な値: Kafka クラスター内のブローカーの数以下の正の数。ブローカーのデフォルトを使用する場合は -1

重要度: 低

設定ストレージピックの作成時に使用されるレプリケーション係数。

connect.protocol

タイプ: string

デフォルト: sessioned

有効な値: [eager, compatible, sessioned]

重要度: 低

Kafka Connect プロトコルの互換性モード。

header.converter

タイプ: class

デフォルト: org.apache.kafka.connect.storage.SimpleHeaderConverter

重要度: 低

Kafka Connect 形式と Kafka に書き込まれるシリアライズされた形式の間の変換に使用される HeaderConverter クラス。これは、Kafka に対して書き込みまたは読み取りされたメッセージのヘッダー値の形式を制御します。これはコネクタから独立しているため、任意のコネクタが任意のシリアル化形式で動作することができます。一般的なフォーマットの例には、JSON や Avro などがあります。デフォルトでは、SimpleHeaderConverter はヘッダー値を文字列にシリアライズし、スキーマを推測してデシリアライズするために使用されます。

inter.worker.key.generation.algorithm

型: string

デフォルト: HmacSHA256

有効な値: ワーカー JVM によってサポートされる KeyGenerator アルゴリズム

重要度: 低

内部要求キーの生成に使用するアルゴリズム。

inter.worker.key.size

タイプ: int

デフォルト: null

重要度: 低

内部要求の署名に使用するキーのサイズ (ビット単位)。null の場合、キー生成アルゴリズムのデフォルトキーサイズが使用されます。

inter.worker.key.ttl.ms

型: int

デフォルト: 3600000 (1 時間)

有効な値: [0,...,2147483647]

重要度: 低

内部リクエスト検証に使用される生成されたセッションキーの TTL (ミリ秒単位)。

inter.worker.signature.algorithm

型: string

デフォルト: HmacSHA256

有効な値: ワーカー JVM によってサポートされる MAC アルゴリズム

重要度: 低

内部リクエストの署名に使用されるアルゴリズム。

inter.worker.verification.algorithms

型: list

デフォルト: HmacSHA256

有効な値: 1つ以上の MAC アルゴリズムの一覧 (各アルゴリズムはワーカー JVM でサポートされる)

重要度: 低

内部要求を検証するために許可されたアルゴリズムのリスト。

listeners

型: list

デフォルト: <http://:8083>

有効な値: コマ区切りの URL (例: <http://localhost:8080>,<https://localhost:8443>)

重要度: 低

REST API がリスンするコマ区切りの URI の一覧。サポートされるプロトコルは HTTP および HTTPS です。ホスト名を 0.0.0.0 として指定し、すべてのインターフェイスにバインドします。ホスト名を空白のままにして、デフォルトのインターフェイスにバインドします。正当なリスナーリストの例は、[HTTP://myhost:8083](http://myhost:8083)、[HTTPS://myhost:8084](https://myhost:8084) です。

metadata.max.age.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [0,...]

重要度: low

新しいブローカーまたはパーティションをプロアクティブに検出するためのパーティションリーダーシップの変更がない場合でも、メタデータの更新を強制するまでの期間 (ミリ秒単位)。

metric.reporters

タイプ: list

デフォルト: ""

重要度: 低

メトリクスレポーターとして使用するクラスの一

覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 低

メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string

デフォルト: INFO

有効な値: [INFO, DEBUG]

重要度: 低

メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

メトリクスサンプルが計算される時間枠。

offset.flush.interval.ms

型: long

デフォルト: 60000 (1分)

重要度: 低

タスクのオフセットのコミットを試行する間隔。

offset.flush.timeout.ms

型: long

デフォルト: 5000 (5 秒)

重要度: 低

プロセスをキャンセルし、今後の試行でコミットするオフセットデータを復元する前に、レコードをフラッシュして、パーティションオフセットデータがオフセットストレージにコミットするまでの最大待機時間(ミリ秒単位)。

offset.storage.partitions

型: int

デフォルト: 25

有効な値: 正の数。ブローカーのデフォルトを使用する場合は -1

重要度: 低

オフセットストレージトピックの作成時に使用されるパーティションの数。

offset.storage.replication.factor

型: short

デフォルト: 3

有効な値: Kafka クラスター内のブローカーの数以下の正の数。ブローカーのデフォルトを使用する場合は -1

重要度: 低

オフセットストレージトピックの作成時に使用されるレプリケーション係数。

plugin.path

タイプ: list

デフォルト: null

重要度: 低

プラグイン (コネクタ、コンバーター、変換) が含まれるコンマ (,) で区切られたパスのリスト。リストは、次の任意の組み合わせを含むトップレベルのディレクトリーで設定される必要があります。a) プラグインとその依存関係を含む jars を直接含むディレクトリー b) プラグインとその依存関係を含む uber-jars c) プラグインのクラスとその依存関係のパッケージディレクトリー構造を直接含むディレクトリー。注記: 依存関係またはプラグインを検出するために、シンボリックリンクが続きます。例: `plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors`。設定プロバイダーが初期化され、変数の置き換えとして使用される前にワーカーのスキャナーによって raw パスが使用されるため、このプロパティーで設定プロバイダーの変数を使用しないでください。

reconnect.backoff.max.ms

型: long

デフォルト: 1000 (1 秒)

有効な値: [0,...]

重要度: 低

接続に繰り返し失敗したブローカーへの再接続時に待機する最大時間 (ミリ秒単位)。これが指定されている場合、ホストごとのバックオフは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。バックオフの増加を計算した後、コネクションストームを回避するために 20% のランダムなジッターが追加されます。

reconnect.backoff.ms

型: long

デフォルト: 50

有効な値: [0,...]

重要度: 低

特定のホストへの再接続を試みる前に待機するベース時間。これにより、タイトなループでホストに繰り返し接続することを回避します。このバックオフは、クライアントによるブローカーへのすべての接続試行に適用されます。

response.http.headers.config

型: string

デフォルト: ""

有効な値: コンマ区切りのヘッダルール。各ヘッダルールの形式は `'[action] [header name]: [header value]'` で、ヘッダルールの一部にコンマが含まれている場合は二重引用符で囲むことができます。

重要度: 低

REST API HTTP レスポンスヘッダーのルール。

rest.advertised.host.name

タイプ: string

デフォルト: null

重要度: 低

これが設定されている場合、これは、他のワーカーの接続用に提供されるホスト名です。

rest.advertised.listener

タイプ: string

デフォルト: null

重要度: 低

他のワーカーが使用するために提供されるアドバタイズされたリスナー (HTTP または HTTPS) を設定します。

rest.advertised.port

タイプ: int
デフォルト: null
重要度: 低

これが設定されている場合、これは、他のワーカーの接続用に提供されるポートになります。

rest.extension.classes

タイプ: list
デフォルト: ""
重要度: 低

指定された順序でロードされ、呼び出される **ConnectRestExtension** クラスのコンマ区切りリスト。インターフェイス **ConnectRestExtension** を実装すると、フィルターなどの Connect の REST API ユーザー定義リソースを注入できます。通常、ログイン、セキュリティーなどのカスタム機能の追加に使用されます。

retry.backoff.ms

型: long
デフォルト: 100
有効な値: [0,...]
重要度: 低

特定のトピックパーティションに対して失敗したリクエストを再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

sasl.kerberos.kinit.cmd

タイプ: string
デフォルト: /usr/bin/kinit
重要度: 低
Kerberos kinit コマンドパス。

sasl.kerberos.min.time.before.relogin

型: long
デフォルト: 60000
重要度: 低
更新試行間のログインスレッドのスリープ時間。

sasl.kerberos.ticket.renew.jitter

型: double
デフォルト: 0.05
重要度: 低
更新時間に追加されたランダムなジッターの割合。

sasl.kerberos.ticket.renew.window.factor

型: double
デフォルト: 0.8
重要度: 低
ログインスレッドは、最後の更新からチケットの有効期限までの指定された時間のウィンドウファクターに達するまでスリープし、その時点でチケットの更新を試みます。

sasl.login.connect.timeout.ms

タイプ: int
デフォルト: null
重要度: 低

外部認証プロバイダーの接続タイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.read.timeout.ms

型: int

デフォルト: null

重要度: 低

外部認証プロバイダーの読み取りタイムアウトの (オプションの) ミリ秒単位の値。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.buffer.seconds

型: short

デフォルト: 300

有効な値: [0,...,3600]

重要度: 低

クレデンシャルを更新するときに維持するクレデンシャルの有効期限が切れるまでのバッファ時間 (秒単位)。更新が、バッファ秒数よりも有効期限に近いときに発生する場合、更新は、可能な限り多くのバッファ時間を維持するために上に移動されます。設定可能な値は 0 から 3600 (1 時間) です。値を指定しない場合は、デフォルト値の 300 (5 分) が使用されます。この値および `sasl.login.refresh.min.period.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合にどちらも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.min.period.seconds

型: short

デフォルト: 60

有効な値: [0,...,900]

重要度: 低

ログイン更新スレッドがクレデンシャルを更新する前に待機する最小時間 (秒単位)。設定可能な値は 0 から 900 (15 分) です。値を指定しない場合は、デフォルト値の 60 (1 分) が使用されます。この値および `sasl.login.refresh.buffer.seconds` は、その合計がクレデンシャルの残りの有効期間を超える場合に両方とも無視されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.factor

型: double

デフォルト: 0.8

有効な値: [0.5,...,1.0]

重要度: 低

ログイン更新スレッドは、クレデンシャルの有効期間との関連で指定の期間ファクターに達するまでスリープ状態になり、達成した時点でクレデンシャルの更新を試みます。設定可能な値は 0.5(50%) から 1.0(100%) までです。値が指定されていない場合、デフォルト値の 0.8(80%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.refresh.window.jitter

型: double

デフォルト: 0.05

有効な値: [0.0,...,0.25]

重要度: 低

ログイン更新スレッドのスリープ時間に追加されるクレデンシャルの存続期間に関連するランダムジッターの最大量。設定可能な値は 0 から 0.25(25%) です。値が指定されていない場合は、デフォルト値の 0.05(5%) が使用されます。現在、OAUTHBEARER にのみ適用されます。

sasl.login.retry.backoff.max.ms

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーへのログイン試行間の最大待機時間のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms` 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.login.retry.backoff.ms`

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーへのログイン試行間の最初の待機のミリ秒単位の (オプションの) 値。ログインでは、指数関数バックオフアルゴリズムが使用され、初期待機時間は `sasl.login.retry.backoff.ms` 設定に基づき、試行間の待機時間は 2 倍になり、最大待機時間は `sasl.login.retry.backoff.max.ms` 設定に基づき、指定されます。現在、OAUTHBEARER にのみ適用されます。

`sasl.oauthbearer.clock.skew.seconds`

型: int

デフォルト: 30

重要度: 低

OAuth/OIDC ID プロバイダーとブローカーの時間の差を考慮した秒単位の (オプションの) 値。

`sasl.oauthbearer.expected.audience`

タイプ: list

デフォルト: null

重要度: 低

JWT が想定される対象者の 1 つに対して発行されたことを確認するためにブローカーが使用する (オプションの) コンマ区切りの設定です。JWT は標準的な OAuth の `aud` クレームについて検査され、この値が設定されている場合、ブローカーは JWT の `aud` クレームから値が完全に一致するかどうかを確認するために照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

`sasl.oauthbearer.expected.issuer`

タイプ: string

デフォルト: null

重要度: 低

ブローカーが JWT が想定される issuer によって作成されたことを確認するために使用する (オプションの) 設定。JWT は標準の OAuth `iss` クレームについて検査され、この値が設定されている場合、ブローカーは JWT の `iss` クレームにあるものと正確に照合します。一致するものがない場合、ブローカーは JWT を拒否し、認証は失敗します。

`sasl.oauthbearer.jwks.endpoint.refresh.ms`

型: long

デフォルト: 3600000 (1 時間)

重要度: 低

ブローカーが JWT の署名を検証するためのキーを含む JWKS (JSON Web Key Set) キャッシュを更新するまで待機するミリ秒単位の (オプションの) 値。

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

型: long

デフォルト: 10000 (10 秒)

重要度: 低

外部認証プロバイダーから JWKS (JSON Web Key Set) を取得する試行間の最大待機時間のミリ秒単位の (オプションの) 値。JWKS の取得では、`sasl.oauthbearer.jwks.endpoint.retry.backoff.ms` 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、`sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms` 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

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

型: long

デフォルト: 100

重要度: 低

外部認証プロバイダーからの JWKS (JSON Web Key Set) 取得する試行間の最初の待機のミリ秒単位の (オプションの) 値。JWKS の取得では、`sasl.oauthbearer.jwks.endpoint.retry.backoff.ms` 設定に基づく初期待機を伴う指数関数バックオフアルゴリズムが使用され、`sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms` 設定で指定された最大待機時間まで試行間の待機時間が 2 倍になります。

sasl.oauthbearer.scope.claim.name

Type: 文字列

Type: スコープ

Importance: 低

スコープの OAuth クレームはスコープと呼ばれることがよくありますが、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、この (オプションの) 設定は JWT ペイロードのクレームに含まれるスコープに使用する別の名前を提供できます。

sasl.oauthbearer.sub.claim.name

Type: 文字列

Default: sub

Importance: 低

サブジェクトの OAuth クレームは sub と呼ばれることがよくありますが、この (オプションの) 設定は、OAuth/OIDC プロバイダーがそのクレームに別の名前を使用する場合、JWT ペイロードのクレームに含まれるサブジェクトに使用する別の名前を提供できます。

scheduled.rebalance.max.delay.ms

型: int

デフォルト: 300000 (5 分)

有効な値: [0,...,2147483647]

重要度: low

1人または複数の離脱したワーカーが戻ってくるのを待ってから、コネクタやタスクをリバランスしてグループに再割り当てるために予定されている最大の遅延時間。この期間は、離脱したワーカーのコネクタやタスクが割り当てられないままです。

socket.connection.setup.timeout.max.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

ソケット接続が確立されるまでクライアントが待機する最大時間。接続設定のタイムアウトは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。接続ストームを回避するために、タイムアウトに 0.2 のランダム化ファクターが適用され、計算された値よりも 20% 未満と 20% 超の間にランダムな範囲が発生します。

socket.connection.setup.timeout.ms

型: long

デフォルト: 10000 (10 秒)

有効な値: [0,...]

重要度: 低

ソケット接続が確立されるまでクライアントが待機する時間。タイムアウトが経過する前に接続が構築されない場合、クライアントはソケットチャネルを閉じます。

ssl.cipher.suites

タイプ: list

デフォルト: null

重要度: 低

暗号化スイートの一覧。これは、TLS または SSL ネットワークプロトコルを使用してネットワーク接続のセキュリティ設定をネゴシエートするために使用される認証、暗号化、MAC、および鍵交換アルゴリズムの名前付きの組み合わせです。デフォルトでは、利用可能なすべての暗号スイートがサポートされます。

ssl.client.auth

タイプ: string

デフォルト: none

重要度: 低

クライアント認証を要求する kafka ブローカーを設定します。以下の設定は一般的な設定です。

- **ssl.client.auth=required**: 必要なクライアント認証が必要な場合
- **ssl.client.auth=requested**: これは、クライアント認証が任意であることを意味します。required とは異なり、このオプションが設定されている場合、クライアントは自身に関する認証情報を提供しないことを選択できます。
- **ssl.client.auth=none**: これは、クライアント認証が不要であることを意味します。

ssl.endpoint.identification.algorithm

タイプ: string

デフォルト: https

重要度: 低

サーバー証明書を使用してサーバーのホスト名を検証するエンドポイント識別アルゴリズム。

ssl.engine.factory.class

タイプ: class

デフォルト: null

重要度: 低

SSL Engine オブジェクトを提供する org.apache.kafka.common.security.auth.SslEngineFactory タイプのクラス。デフォルト値は org.apache.kafka.common.security.ssl.DefaultSslEngineFactory です。

ssl.keymanager.algorithm

型: string

デフォルト: SunX509

重要度: 低

SSL 接続のキーマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたキーマネージャーファクトリーアルゴリズムです。

ssl.secure.random.implementation

タイプ: string
デフォルト: null
重要度: 低
SSL 暗号操作に使用する SecureRandom PRNG 実装。

ssl.trustmanager.algorithm

型: string
デフォルト: PKIX
重要度: 低
SSL 接続のトラストマネージャーファクトリーによって使用されるアルゴリズム。デフォルト値は、Java 仮想マシンに設定されたトラストマネージャーファクトリーアルゴリズムです。

status.storage.partitions

型: int
デフォルト: 5
有効な値: 正の数。ブローカーのデフォルトを使用する場合は -1
重要度: 低
ステータスストレージトピックの作成時に使用されるパーティションの数。

status.storage.replication.factor

型: short
デフォルト: 3
有効な値: Kafka クラスター内のブローカーの数以下の正の数。ブローカーのデフォルトを使用する場合は -1
重要度: 低
ステータスストレージトピックの作成時に使用されるレプリケーション係数。

task.shutdown.graceful.timeout.ms

型: long
デフォルト: 5000 (5 秒)
重要度: 低
タスクを正常にシャットダウンするまで待機する時間。これは、タスクごとにではなく、合計時間です。タスクはすべてシャットダウンされ、順番に待機します。

topic.creation.enable

タイプ: boolean
デフォルト: true
重要度: 低
ソースコネクターが **topic.creation.** プロパティで設定されている場合、ソースコネクターによって使用されるトピックの自動作成を許可するかどうか。各タスクは管理クライアントを使用してトピックを作成し、トピックを自動的に作成する Kafka ブローカーに依存しません。

topic.tracking.allow.reset

タイプ: boolean
デフォルト: true
重要度: 低
true に設定すると、ユーザーリクエストはコネクターごとにアクティブなトピックのセットをリセットできます。

topic.tracking.enable

タイプ: boolean

デフォルト: true

重要度: 低

ランタイム時におけるコネクターごとにアクティブなトピックのセットの追跡を有効にします。

第7章 KAFKA ストリームの設定プロパティ

application.id

タイプ: string

重要度: 高

ストリーム処理アプリケーションの識別子。Kafka クラスター内で一意である必要があります。これは、1) デフォルトのクライアント ID の接頭辞、2) メンバーシップ管理のためのグループ ID、3) Changelog のトピックの接頭辞として使用されます。

bootstrap.servers

タイプ: list

重要度: 高

Kafka クラスターへの最初の接続を確立するために使用されるホストとポートのペアの一覧。クライアントは、ブートストラップ用にここで指定されたサーバーに関係なく、すべてのサーバーを利用します。この一覧は、サーバーのフルセットを検出するために使用される最初のホストにのみ影響します。この一覧は、**host1:port1,host2:port2,...** の形式にする必要があります。これらのサーバーは、(動的に変更される可能性がある) 完全なクラスターメンバーシップを検出するための最初の接続にだけ使用されるため、このリストにはサーバーの完全なセットを含める必要はありません(ただし、サーバーがダウンした場合に備えて、複数のサーバーが必要になる場合があります)。

num.standby.replicas

型: int

デフォルト: 0

重要度: 高

各タスクのスタンバイレプリカ数。

state.dir

タイプ: string

デフォルト: /tmp/kafka-streams

重要度: 高

状態ストアのディレクトリーの場所。このパスは、同じ基礎となるファイルシステムを共有するストリームインスタンスごとに一意である必要があります。

acceptable.recovery.lag

型: long

デフォルト: 10000

有効な値: [0,...]

重要度: 中

クライアントがアクティブなタスク割り当てを受け取るのに十分なほど追いつたと見なされる最大許容ラグ(追いつくまでのオフセット数)。割り当て時に、処理前に残りの変更ログを復元します。リバランス中の処理の一時停止を回避するために、この設定は、特定のワークロードの1分未満の回復時間に対応する必要があります。0以上である必要があります。

cache.max.bytes.buffering

型: long

デフォルト: 10485760

有効な値: [0,...]

重要度: 中

すべてのスレッドでバッファするのに使用されるメモリーバイトの最大数。

client.id

タイプ: string

デフォルト: ""

重要度: 中

内部コンシューマー、プロデューサー、および復元コンシューマーのクライアント ID に使用される ID 接頭辞。パターンは、'`<client.id>-StreamThread-<threadSequenceNumber>-<consumer|producer|restore-consumer>`' です。

default.deserialization.exception.handler

タイプ: class

デフォルト: org.apache.kafka.streams.errors.LogAndFailExceptionHandler

重要度: 中

org.apache.kafka.streams.errors.DeserializationExceptionHandler インターフェイスを実装する例外処理クラス。

default.key.serde

タイプ: class

デフォルト: null

重要度: 中

org.apache.kafka.common.serialization.Serde インターフェイスを実装するキーのデフォルトのシリアライザー/デシリアライザークラス。windowed serde クラスを使用する場合は、**org.apache.kafka.common.serialization.Serde** インターフェイスを実装した inner serde クラスを 'default.windowed.key.serde.inner' または 'default.windowed.value.serde.inner' で設定する必要があります。ご注意ください。

default.list.key.serde.inner

タイプ: class

デフォルト: null

重要度: 中

org.apache.kafka.common.serialization.Serde インターフェイスを実装するキーの list serde のデフォルトの内部クラス。この設定は、**default.key.serde** 設定が **org.apache.kafka.common.serialization.Serdes.ListSerde** に設定されている場合にのみ読み取られます。

default.list.key.serde.type

タイプ: class

デフォルト: null

重要度: 中

java.util.List インターフェイスを実装するキーのデフォルトクラス。この設定は、**default.key.serde** 設定が **org.apache.kafka.common.serialization.Serdes.ListSerde** に設定されている場合にのみ読み取られます。リスト serde クラスが使用されている場合は、**org.apache.kafka.common.serialization.Serde** インターフェイスを 'default.list.key.serde.inner' 経由で実装する内部 serde クラスを設定する必要がある点に注意してください。

default.list.value.serde.inner

タイプ: class

デフォルト: null

重要度: 中

org.apache.kafka.common.serialization.Serde インターフェイスを実装する値の list serde のデフォルトの内部クラス。この設定は、**default.value.serde** 設定が **org.apache.kafka.common.serialization.Serdes.ListSerde** に設定されている場合にのみ読み取られます。

default.list.value.serde.type

タイプ: class

デフォルト: null

重要度: 中

java.util.List インターフェイスを実装する値のデフォルトクラス。この設定

は、**default.value.serde** 設定が **org.apache.kafka.common.serialization.Serdes.ListSerde** に設定されている場合のみ読み取られます。リスト serde クラスが使用されている場合

は、**org.apache.kafka.common.serialization.Serde** インターフェイスを

'default.list.value.serde.inner' 経由で実装する内部 serde クラスを設定する必要がある点に注意してください。

default.production.exception.handler

タイプ: class

デフォルト: org.apache.kafka.streams.errors.DefaultProductionExceptionHandler

重要度: 中

org.apache.kafka.streams.errors.ProductionExceptionHandler インターフェイスを実装する例外処理クラス。

default.timestamp.extractor

タイプ: class

デフォルト: org.apache.kafka.streams.processor.FailOnInvalidTimestamp

重要度: 中

org.apache.kafka.streams.processor.TimestampExtractor インターフェイスを実装した、デフォルトのタイムスタンプ抽出クラスです。

default.value.serde

タイプ: class

デフォルト: null

重要度: 中

org.apache.kafka.common.serialization.Serde インターフェイスを実装する値のデフォルトのシリアライザー/デシリアライザークラス。windowed serde クラスを使用する場合

は、**org.apache.kafka.common.serialization.Serde** インターフェイスを実装した inner serde クラスを 'default.windowed.key.serde.inner' または 'default.windowed.value.serde.inner' で設定する必要があります。ことに注意してください。

max.task.idle.ms

型: long

デフォルト: 0

重要度: 中

この設定は、結合とマージが順不同の結果を生成するかどうかを制御します。この設定値は、ストリームタスクが、一部の(すべてではない)入力パーティションに完全に追いついたときに、プロデューサーが追加のレコードを送信するのを待ち、複数の入力ストリームにまたがるレコード処理の順序がずれる可能性を避けるためにアイドル状態を保つ最大時間(ミリ秒単位)になります。デフォルト(ゼロ)は、プロデューサーがさらにレコードを送信するのを待ちませんが、ブローカーにすでに存在するデータをフェッチするのを待ちます。このデフォルトは、ブローカーにすでに存在するレコードの場合、Streams がそれらをタイムスタンプ順に処理することを意味します。-1 に設定すると、アイドルリングを完全に無効にし、ローカルで利用可能なデータをすべて処理しますが、このように処理する場合でも、順序外の処理が発生する可能性があります。

max.warmup.replicas

型: int

デフォルト: 2

有効な値: [1,...]

重要度: 中

タスクが再割り当てされたあるインスタンスでウォームアップしている間に、別のインスタンスでタスクを利用できるようにするために、一度に割り当てることができるウォームアップレプリカ (設定された `num.standbys` を超える追加のスタンバイ) の最大数です。高可用性を確保するために追加のブローカートラフィックとクラスターの状態を調整するのに使用されます。1以上でなければなりません。

num.stream.threads

型: int

デフォルト: 1

重要度: 中

ストリーム処理を実行するスレッドの数。

processing.guarantee

タイプ: string

デフォルト: `at_least_once`

有効な値: [`at_least_once`, `exactly_once`, `exactly_once_beta`, `exactly_once_v2`]

重要度: 中

使用されるべき処理保証です。使用できる値は、**at_least_once** (デフォルト) および **exactly_once_v2** (ブローカーバージョン 2.5 以降が必要) です。非推奨のオプションは **exactly_once** (ブローカーバージョン 0.11.0 以降が必要) および **exactly_once_beta** (ブローカーバージョン 2.5 以降が必要) です。1回だけの処理には、デフォルトで少なくとも3つのブローカーのクラスターが必要であることを注意してください。これは、実稼働環境に推奨される設定です。開発の場合、ブローカー設定 **transaction.state.log.replication.factor** および **transaction.state.log.min.isr** を調整することにより、これを変更できます。

rack.aware.assignment.tags

タイプ: list

デフォルト: ""

有効な値: 最大5つの要素を含むリスト

重要度: 中

Kafka Streams インスタンス間でスタンバイレプリカを配布するために使用されるクライアントタグキーのリスト。設定すると、Kafka ストリームは、各クライアントタグディメンションにスタンバイタスクを分散するために最善を尽くします。

replication.factor

タイプ: int

デフォルト: -1

重要度: 中

ストリーム処理アプリケーションによって作成されたログトピックおよびパーティショントピックを変更するためのレプリケーション係数。デフォルトの **-1** (つまり、ブローカーのデフォルトレプリケーションファクターを使用) にはブローカーバージョン 2.4 以降が必要です。

security.protocol

型: string

デフォルト: PLAINTEXT

重要度: 中

ブローカーとの通信に使用されるプロトコル。有効な値は、PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL です。

task.timeout.ms

型: long

デフォルト: 300000 (5 分)

有効な値: [0,...]

重要度: 中

内部エラーが原因でタスクが停止し、エラーが発生するまで再試行する最大時間 (ミリ秒単位)。タイムアウトが 0ms の場合、タスクは最初の内部エラーに対してエラーを発生させます。タイムアウトが 0ms を超える場合、タスクはエラーが発生する前に少なくとも 1 回再試行します。

topology.optimization

タイプ: string

デフォルト: none

有効な値: [none, all]

重要度: 中

デフォルトでは、トポロジーを最適化する必要がある場合に Kafka Streams に指示する設定。

application.server

タイプ: string

デフォルト: ""

重要度: 低

この KafkaStreams インスタンスでの状態ストア検出とインタラクティブなクエリーに使用できるユーザー定義のエンドポイントを参照する host:port ペア。

buffered.records.per.partition

型: int

デフォルト: 1000

重要度: 低

パーティションごとにバッファーを行う最大レコード数。

built.in.metrics.version

タイプ: string

デフォルト: latest

有効な値: [latest]

重要度: 低

使用する組み込みメトリクスのバージョン。

commit.interval.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

プロセッサの位置を保存する頻度 (ミリ秒単位)。(processing.guarantee が **exactly_once_v2**、**exactly_once** に設定されている場合、デフォルト値は **100** になり、これ以外の場合のデフォルト値は **30000** になる点に注意してください。

connections.max.idle.ms

型: long

デフォルト: 540000 (9 分)

重要度: low

この設定で指定された期間 (ミリ秒単位) の後にアイドル状態の接続を閉じます。

default.dsl.store

タイプ: string
デフォルト: rocksDB
有効な値: [rocksDB, in_memory]
重要度: 低
DSL Operator が使用するデフォルトの状態ストアタイプ。

metadata.max.age.ms

型: long
デフォルト: 300000 (5 分)
有効な値: [0,...]
重要度: low
新しいブローカーまたはパーティションをプロアクティブに検出するためのパーティションリーダーシップの変更がない場合でも、メタデータの更新を強制するまでの期間 (ミリ秒単位)。

metric.reporters

タイプ: list
デフォルト: ""
重要度: 低
メトリクスレポーターとして使用するクラスの一覧。**org.apache.kafka.common.metrics.MetricsReporter** インターフェイスを実装すると、新しいメトリクスの作成が通知されるクラスのプラグが可能になります。JmxReporter は、JMX 統計を登録するために常に含まれます。

metrics.num.samples

型: int
デフォルト: 2
有効な値: [1,...]
重要度: 低
メトリクスを計算するために保持されるサンプルの数。

metrics.recording.level

型: string
デフォルト: INFO
有効な値: [INFO, DEBUG, TRACE]
重要度: 低
メトリクスの最も高い記録レベル。

metrics.sample.window.ms

型: long
デフォルト: 30000 (30 秒)
有効な値: [0,...]
重要度: 低
メトリクスサンプルが計算される時間枠。

poll.ms

型: long
デフォルト: 100
重要度: 低
入力を待つためにブロックする時間をミリ秒単位で指定します。

probing.rebalance.interval.ms

型: long

デフォルト: 600000 (10 分)

有効な値: [60000,...]

重要度: low

ウォーミングアップが終了し、アクティブになる準備ができていないウォームアップレプリカをプローブするためにリバランスをトリガーするまで待機する最大時間 (ミリ秒単位)。プローブリバランスは、割り当てが分散されるまで引き続きトリガーされます。1分以上でなければなりません。

receive.buffer.bytes

型: int

デフォルト: 32768 (32 キビバイト)

有効な値: [-1,...]

重要度: low

データの読み取り時に使用する TCP 受信バッファ (SO_RCVBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

reconnect.backoff.max.ms

型: long

デフォルト: 1000 (1 秒)

有効な値: [0,...]

重要度: 低

接続に繰り返し失敗したブローカーへの再接続時に待機する最大時間 (ミリ秒単位)。これが指定されている場合、ホストごとのバックオフは、連続して接続に失敗するたびに、この最大値まで指数関数的に増加します。バックオフの増加を計算した後、接続ストリームを回避するために 20% のランダムなジッターが追加されます。

reconnect.backoff.ms

型: long

デフォルト: 50

有効な値: [0,...]

重要度: 低

特定のホストへの再接続を試みる前に待機するベース時間。これにより、タイトなループでホストに繰り返し接続することを回避します。このバックオフは、クライアントによるブローカーへのすべての接続試行に適用されます。

repartition.purge.interval.ms

型: long

デフォルト: 30000 (30 秒)

有効な値: [0,...]

重要度: 低

再分割トピックから完全に消費されたレコードを削除する頻度 (ミリ秒単位)。ページは、最後のページから少なくともこの値の後に行われますが、それ以降まで遅れる場合があります。

(**commit.interval.ms** とは異なり、**processing.guarantee** が **exact_once_v2** に設定されている場合、この値のデフォルトは変更されないことに注意してください)。

request.timeout.ms

型: int

デフォルト: 40000 (40 秒)

有効な値: [0,...]

重要度: 低

この設定は、クライアントの要求の応答を待つ最大時間を制御します。タイムアウトが経過する前に応答が受信されない場合、クライアントは必要に応じてリクエストを再送信します。または、再試行が使い切られるとリクエストが失敗します。

retries

型: int

デフォルト: 0

有効な値: [0,...,2147483647]

重要度: 低

ゼロより大きい値を設定すると、クライアントは、一時的なエラーの可能性がある失敗した要求を再送信します。値をゼロまたは **MAX_VALUE** のいずれかに設定し、対応するタイムアウトパラメータを使用して、クライアントがリクエストを再試行する期間を制御することが推奨されます。

retry.backoff.ms

型: long

デフォルト: 100

有効な値: [0,...]

重要度: 低

特定のトピックパーティションに対して失敗したリクエストを再試行するまでの待機時間。これにより、一部の障害シナリオでタイトループでリクエストを繰り返し送信することを回避できます。

rocksdb.config.setter

タイプ: class

デフォルト: null

重要度: 低

org.apache.kafka.streams.state.RocksDBConfigSetter インターフェイスを実装する Rocks DB 設定セッタークラスまたはクラス名。

send.buffer.bytes

型: int

デフォルト: 131072 (128 キビバイト)

有効な値: [-1,...]

重要度: low

データの送信時に使用する TCP 送信バッファ (SO_SNDBUF) のサイズ。値が -1 の場合、OS のデフォルトが使用されます。

state.cleanup.delay.ms

型: long

デフォルト: 600000 (10 分)

重要度: low

パーティションが移行されたときに状態を削除するまで待機する時間 (ミリ秒単位)。少なくとも **state.cleanup.delay.ms** の間変更されていない state ディレクトリのみが削除されます。

upgrade.from

タイプ: string

デフォルト: null

有効な値: [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]

重要度: 低

後方互換性のある方法でのアップグレードを許可します。これは、[0.10.0, 1.1] から 2.0+ にアップグレード、または [2.0, 2.3] から 2.4+ にアップグレードする際に必要です。2.4 から新しいバージョンにアップグレードする場合は、この設定を指定する必要はありません。デフォルトは **null** です。許

可される値は 0.10.0、0.10.1、0.10.2、0.11.0、1.0、1.1、2.0、2.1、2.2、2.3 (対応する旧バージョンからのアップグレード用) です。

window.size.ms

タイプ: long

デフォルト: null

重要度: 低

ウィンドウの終了時間を計算するためにデシリアライザーのウィンドウサイズを設定します。

windowed.inner.class.serde

タイプ: string

デフォルト: null

重要度: 低

ウィンドウ表示されたレコードの内部クラスのデフォルトのシリアライザー/デシリアライザー。"`org.apache.kafka.common.serialization.Serde`` インターフェイスを実装する必要があります。KafkaStreams アプリケーションでこの設定を設定すると、Plain コンシューマクライアントからのみ使用されることが意図されているため、エラーが発生することに注意してください。

windowstore.changelog.additional.retention.ms

型: long

デフォルト: 86400000 (1日)

重要度: 低

windows maintainMs に追加され、データがログから早期に削除されないようにします。クロックドリフトを許可します。デフォルトは1日です。

付録A サブスクリプションの使用

AMQ Streams は、ソフトウェアサブスクリプションから提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

アカウントへのアクセス

1. access.redhat.com に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

サブスクリプションのアクティベート

1. access.redhat.com に移動します。
2. **サブスクリプション** に移動します。
3. **Activate a subscription** に移動し、16 桁のアクティベーション番号を入力します。

Zip および Tar ファイルのダウンロード

zip または tar ファイルにアクセスするには、カスタマーポータルを使用して、ダウンロードする関連ファイルを検索します。RPM パッケージを使用している場合は、この手順は必要ありません。

1. ブラウザーを開き、access.redhat.com/downloads で Red Hat カスタマーポータルの **製品のダウンロード** ページにログインします。
2. **インテグレーションおよび自動化** カテゴリで、**AMQ Streams for Apache Kafka** エントリーを見つけます。
3. 必要な AMQ Streams 製品を選択します。**Software Downloads** ページが開きます。
4. コンポーネントの **ダウンロード** リンクをクリックします。

DNF を使用したパッケージのインストール

パッケージとすべてのパッケージ依存関係をインストールするには、以下を使用します。

```
dnf install <package_name>
```

ローカルディレクトリーからダウンロード済みのパッケージをインストールするには、以下を使用します。

```
dnf install <path_to_download_package>
```

改訂日時: 2022-10-08 21:35:09 +1000