**ИНСТРУКЦИЯ АДМИНИСТРАТОРА**

**RT.STREAMING**

2025

# 1 Введение

## 1.1 Наименование и условное обозначение

Наименование системы: **RT.StreamingNiFi**.

## 1.2 Область применения

**RT.StreamingNiFi** – программное обеспечение, предназначенное для организации ETL-процессов в рамках экосистемы RT.DataLake (Hadoop).

# 2 Администрирование RT.StreamingNiFi

## 2.1 Рекомендации по конфигурации

При работе в **Linux** необходимо учесть последующие рекомендации, так как типичные значения по умолчанию для **Linux** могут быть не настроены под нужды такого интенсивного приложения с высоким уровнем ввода-вывода, как **RT.StreamingNiFi**.

### 2.1.1 Максимальное количество дескрипторов файлов

**RT.StreamingNiFi**в любой момент потенциально может иметь очень большое количество открытых дескрипторов файлов. Увеличьте ограничения, отредактировав **/etc/security/limits.conf**, добавив что-то вроде:

\* hard nofile 50000

\* soft nofile 50000

Copy

### 2.1.2 Максимальное количество разветвленных процессов

**RT.StreamingNiFi**может быть настроен на создание значительного количества потоков. Чтобы увеличить допустимое число, отредактируйте **/etc/security/limits.conf**.

\* hard nproc 10000

\* soft nproc 10000

Copy

И ваш дистрибутив может потребовать редактирования **/etc/security/limits.d/90-nproc.conf**, добавив:

### 2.1.3 Увеличение количества доступных портов TCP-сокетов

Это особенно важно, если ваш поток будет устанавливать и отключать большое количество сокетов за небольшой промежуток времени.

sudo sysctl -w net.ipv4.ip\_local\_port\_range="10000 65000"

Copy

### 2.1.4 Продолжительность статуса сокетов TIMED\_WAIT при закрытии

Установите, как долго сокеты остаются в статусе **TIMED\_WAIT** при закрытии.

Вы не хотите, чтобы ваши сокеты простаивали слишком долго, поскольку вы хотите иметь возможность быстро настраивать и отключать новые сокеты. Установите что-то вроде этого:

**для ядра 2.6**

sudo sysctl -w net.ipv4.netfilter.ip\_conntrack\_tcp\_timeout\_time\_wait="1"

Copy

**для ядра 3.0**

sudo sysctl -w net.netfilter.nf\_conntrack\_tcp\_timeout\_time\_wait="1"

Copy

### 2.1.5 Отключение обмена в Linux

Для некоторых приложений свопинг является нормой. Для **RT.StreamingNiFi**это не так, так как он всегда должен быть запущенным. Чтобы сообщить **Linux**, что вы хотите отключить свопинг, вы можете отредактировать**/etc/sysctl.conf**, добавив следующую строку:

vm.swappiness = 0

Copy

Для партиций, обрабатывающих различные репозитории **RT.StreamingNiFi**, отключите такие функции, как **atime**. Это может привести к неожиданному увеличению производительности. Отредактируйте файл **/etc/fstab**и для необходимых партиций добавьте параметр **noatime**.

## 2.2 Рекомендуемые исключения антивируса

Антивирусному ПО может потребоваться много времени для сканирования больших каталогов и многочисленных файлов в них. Кроме того, если антивирусное ПО блокирует файлы или каталоги во время сканирования, эти ресурсы становятся недоступными для процессов **RT.StreamingNiFi**, что приводит к задержке или недоступности этих ресурсов в инстансе/кластере **RT.StreamingNiFi**. Чтобы предотвратить возникновение этих проблем с производительностью и надёжностью, настоятельно рекомендуется настроить антивирусное ПО на пропуск сканирования следующих каталогов **RT.StreamingNiFi**:

content\_repository

flowfile\_repository

logs

provenance\_repository

state

Copy

## 2.3 Конфигурация логирования

**RT.StreamingNiFi**использует [**logback**](https://logback.qos.ch/)в качестве имплементации ведения лога во время выполнения. Каталог **conf**содержит стандартную конфигурацию **logback.xml** с настройками приложения и уровня по умолчанию. [Руководство **logback**](https://logback.qos.ch/manual/index.html) содержит полный справочник доступных опций.

### 2.3.1 Стандартные файлы логов

Стандартная конфигурация лога включает в себя следующие определения приложений и связанные файлы логов:

| **Файл** | **Описание** |
| --- | --- |
| **nifi-app.log** | Лог приложения, содержащий сообщения фреймворка и компонентов. |
| **nifi-bootstrap.log** | Лог начальной загрузки, содержащий сообщения о запуске и завершении работы. |
| **nifi-deprecation.log** | Лог прекращения поддержки, содержащий предупреждения о компонентах и функциях, которые перестают поддерживаться вендором. |
| **nifi-request.log** | Лог HTTP-запросов, содержащий сообщения доступа к пользовательскому интерфейсу и REST API. |
| **nifi-user.log** | Лог пользователя, содержащий сообщения аутентификации и авторизации. |

### 2.3.2 Логирование прекращения поддержки

Файл **nifi-deprecation.log** содержит предупреждающие сообщения, описывающие компоненты и функции, которые будут удалены в последующих версиях. Предупреждения о прекращении поддержки следует оценивать и устранять, чтобы избежать серьёзных изменений при обновлении до новой мажорной версии. Устранение предупреждений о прекращении поддержки включает обновление до новых компонентов, изменение настроек свойств компонентов или рефакторинг классов пользовательских компонентов.

Логирование прекращения поддержки предоставляет метод проверки совместимости перед обновлением одной мажорной версии выпуска на другую. Обновление до последней минорной версии обеспечит наиболее точный набор предупреждений о прекращении поддержки.

Важно отметить, что логирование прекращения поддержки применимо как к компонентам, так и к функциям. Для логирования выводимых из поддержки функций требуется ссылка на затрагиваемое свойство или метод во время их выполнения. Отключённые компоненты с устаревшими свойствами или методами не будут создавать логи прекращения поддержки. По этой причине важно тестировать все настроенные компоненты достаточно долго, чтобы проверить стандартное поведение потока.

Логирование прекращения поддержки может генерировать повторяющиеся сообщения в зависимости от конфигурации компонента и шаблонов использования. Отключение логирования прекращения поддержки для определённого класса компонентов можно настроить, добавив элемент **logger**в **logback.xml**. Атрибут **name**должен начинаться с **deprecation**, за которым следует класс компонента. Установка атрибута **level**в значение **OFF**отключает логирование прекращения поддержки для указанного компонента.

<logger name="deprecation.org.apache.nifi.processors.ListenLegacyProtocol" level="OFF" />

Copy

## 2.4 Конфигурирование безопасности

**RT.StreamingNiFi** предоставляет несколько различных вариантов конфигурации для целей безопасности. Наиболее важные свойства находятся в разделе **security properties** в файле **nifi.properties**. Для безопасной работы необходимо установить следующие свойства:

| **Свойство** | **Описание** |
| --- | --- |
| **nifi.security.keystore** | Имя файла Keystore, содержащего закрытый ключ сервера. |
| **nifi.security.keystoreType** | Тип Keystore. Должен быть **PKCS12**, **JKS**или **BCFKS**. **JKS**является предпочтительным типом, файлы **BCFKS**и **PKCS12**загружаются поставщиком **BouncyCastle**. |
| **nifi.security.keystorePasswd** | Пароль Keystore. |
| **nifi.security.keyPasswd** | Пароль для сертификата в Keystore. Если значение не установлено, будет использоваться значение **nifi.security.keystorePasswd*.*** |
| **nifi.security.truststore** | Имя файла Truststore, которое будет использоваться для авторизации пользователей, подключающихся к **RT.StreamingNiFi**. Защищённый инстанс без Truststore отклоняет все входящие подключения. |
| **nifi.security.truststoreType** | Тип Truststore. Должен быть **PKCS12**, **JKS**или **BCFKS**. **JKS**является предпочтительным типом, файлы **BCFKS**и **PKCS12**загружаются поставщиком **BouncyCastle**. |
| **nifi.security.truststorePasswd** | Пароль Truststore. |

После настройки вышеуказанных свойств мы можем включить доступ к пользовательскому интерфейсу через HTTPS вместо HTTP. Это достигается путём установки свойств **nifi.web.https.host** и **nifi.web.https.port**. Свойство **nifi.web.https.host**указывает, на каком имени хоста должен работать сервер. Если требуется, чтобы интерфейс HTTPS был доступен со всех сетевых интерфейсов, следует использовать значение **0.0.0.0**. Чтобы администраторы могли настроить запуск приложения только на определённых сетевых интерфейсах, можно указать свойства **nifi.web.http.network.interface\*** или **nifi.web.https.network.interface\***.

**Внимание**. При включении HTTPS важно, чтобы свойство **nifi.web.http.port** было отключено. **RT.StreamingNiFi** поддерживает работу только по HTTP или HTTPS, а не по обоим одновременно.

Веб-сервер **RT.StreamingNiFi** требует аутентификацию клиента на основе сертификата для пользователей, получающих доступ к пользовательскому интерфейсу, если не настроен альтернативный механизм аутентификации, для которого потребуется односторонний SSL (например, LDAP, OpenID Connect и т.д.). Включение альтернативного механизма аутентификации настроит веб-сервер на аутентификацию клиента на базе сертификатов **WANT**. Это позволит поддерживать пользователей с сертификатами, а те, у кого их нет, смогут входить в систему с учётными данными.

Теперь, когда пользовательский интерфейс защищён, мы можем легко защитить соединения Site-to-Site, а также коммуникации внутри кластера. Это достигается путём установки свойств **nifi.remote.input.secure** и **nifi.cluster.protocol.is.secure** соответственно в значение **true**. Для этих коммуникаций всегда требуется двусторонний SSL, поскольку ноды будут использовать настроенные keystore\truststore для аутентификации.

Автоматическое обновление SSL context factory **RT.StreamingNiFi** можно включить, используя следующие свойства:

| **Свойство** | **Описание** |
| --- | --- |
| **nifi.security.autoreload.enabled** | Указывает, следует ли автоматически перезагружать SSL context factory при обнаружении обновлений keystore и truststore. По умолчанию для него установлено значение **false**. |
| **nifi.security.autoreload.interval** | Указывает интервал, с которым keystore и truststore проверяются на наличие обновлений. Применяется только в том случае, если для параметра **nifi.security.autoreload.enabled** установлено значение **true**. Значение по умолчанию – **10 secs**. |

Как только для свойства **nifi.security.autoreload.enabled** установлено значение **true**, любые действительные изменения в настроенном keystore и truststore приведут к перезагрузке  SSL context factory **RT.StreamingNiFi**, что позволит клиентам принять изменения. Это предназначено для того, чтобы позволить обновлять сертификаты с истёкшим сроком действия в keystore и добавлять новые доверенные сертификаты в truststore, и все это без необходимости перезапуска сервера **RT.StreamingNiFi**.

**Внимание**. Изменения любого из свойств **nifi.security.keystore\*** или **nifi.security.truststore\*** не будут учитываться логикой автоматического обновления, которая предполагает, что пароли и пути к хранилищу останутся прежними.

### 2.4.1 TLS Toolkit

Для упрощения установки **RT.StreamingNiFi**и автоматического создания необходимых keystore, truststore и соответствующих файлов конфигурации можно использовать CLI**tls-toolkit**, что так же обеспечит безопасность многочисленных нод **RT.StreamingNiFi**.

#### 2.4.1.1 Wildcard-сертификаты

Wildcard-сертификаты (т.е. две ноды **node1.nifi.apache.org** и **node2.nifi.apache.org**, которым назначается тот же сертификат с записью CN или SAN **\*.nifi.apache.org**) официально не поддерживаются и не рекомендуются. Использование wildcard-сертификатов имеет множество недостатков. Записи SAN с wildcard допустимы, если каждый сертификат содержит дополнительную уникальную запись SAN и запись CN.

Потенциальные проблемы использования wildcard-сертификатов:

* Кластерные связи многократно используют идентификаторы сертификатов для определения ноды, а если сертификат представляет собой подстановочное DN, то он не даст ответа;
* Администраторам может потребоваться предоставить кастомный идентификатор ноды в **authorizers.xml** для **\*.nifi.apache.org**, поскольку все действия прокси-сервера разрешаются только в сертификате DN;
* Администраторы не имеют возможности отслеживать, в какой ноде выполняется действие, так как все они направляются в один и тот же DN;
* Администраторы, запускающие несколько инстансов на одном компьютере и используя разные порты для их идентификации, могут случайно поместить ноду **node1**с портом **node2**, и адрес будет в итоге удалён, потому что она использует тот же сертификат, а обработчик ноды блокирует его, так как имя ноды **node1**не указано в качестве допустимого хоста для инстанса **node2**;
* Если wildcard-сертификат скомпрометирован, все ноды оказываются под угрозой.

**Внимание.***Для****keystores****и****truststores****в****RT.StreamingNiFi****рекомендуются JKS. Этот инструмент позволяет задавать другие типы*keystore*в командной строке и игнорировать тип PKCS12 для использования в качестве*truststore*, потому что данный формат имеет проблемы совместимости между имплементациями BouncyCastle и Oracle.*

#### 2.4.1.2 Режимы работы

Инструмент командной строки **tls-toolkit** имеет два основных режима работы:

* Standalone (автономный) – создает организацию сертификатов, keystore, truststore и файлы **nifi.properties** в одной команде;
* Client/Server (Клиент/Сервер) – использует Certificate Authority Server, который принимает запросы на подписание сертификатов от клиентов, подписывает и отправляет обратно. И клиент, и сервер проверяют идентификацию друг друга через общий секрет.

#### 2.4.1.2.1 Standalone

Автономный режим вызывается запуском **./bin/tls-toolkit.sh standalone -h** и отображает информацию об использовании с описаниями опций, которые могут быть указаны.

В автономном режиме с **tls-toolkit** можно использовать следующие параметры командной строки:

* **-a, --keyAlgorithm <arg>** – алгоритм использования сгенерированных ключей (по умолчанию: RSA);
* **-B, --clientCertPassword <arg>** – пароль сертификата клиента. Должно быть либо одно значение, либо одно для каждого DN клиента (если не задано, генерируется автоматически);
* **-c, --certificateAuthorityHostname <arg>** – имя хоста **RT.StreamingNiFi**Certificate Authority (по умолчанию: **localhost**);
* **-C, --clientCertDn <arg>** – создание сертификата клиента, подходящего для использования в браузере, с указанным DN (может быть указан несколько раз);
* **-d, --days <arg>** – количество дней, в течение которых выданный сертификат является действительным (по умолчанию: **1095**);
* **-f, --nifiPropertiesFile <arg>** – базовый файл **nifi.properties** для обновления (если не указан, используется встроенный файл, идентичный файлу по умолчанию при установке **NiFi**);
* **-g, --differentKeyAndKeystorePasswords** – использование другого сгенерированного пароля для ключа и keystore;
* **-G, --globalPortSequence <arg>** – использование последовательных портов, которые вычисляются для всех хостов в соответствии с предоставленными выражениями имен хостов (могут быть указаны несколько раз, но должны быть одинаковыми от запуска до запуска);
* **-h, --help** – печать справки и выход;
* **-k, --keySize <arg>** – количество бит для генерации ключей (по умолчанию: **2048**);
* **-K, --keyPassword <arg>** – пароль ключа. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
* **-n, --hostnames <arg>** – список имен хостов через запятую;
* **--nifiDnPrefix <arg>** – строка для добавления имени хоста (в начало) при определении DN (по умолчанию: **CN=**);
* **--nifiDnSuffix <arg>** – строка для добавления имени хоста (в конец) при определении DN (по умолчанию: **OU=NIFI**);
* **-o, --outputDirectory <arg>** – каталог для вывода keystore, truststore и config-файлов (по умолчанию: **../bin**);
* **-O, --isOverwrite** – перезапись существующего вывода хоста;
* **-P, --trustStorePassword <arg>** – пароль truststore. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
* **-s, --signingAlgorithm <arg>** – алгоритм подписи сертификатов (по умолчанию: **SHA256WITHRSA**);
* **-S, --keyStorePassword <arg>** – пароль keytstore. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
* **--subjectAlternativeNames <arg>** – разделенный запятыми список доменов для использования в качестве альтернативных имен в сертификате;
* **-T, --keyStoreType <arg>** – тип создаваемого keystore (по умолчанию: **jks**).

Шаблоны имен хостов:

* Для указания диапазона имен хостов используются квадратные скобки, например: [01-20];
* Круглые скобки используются для определения, что на хосте (хостах) работает больше, чем один инстанс **RT.StreamingNiFi**, например: (5).

Примеры:

* Создать 4 набора keystore , **truststore**, **nifi.properties** для **localhost**вместе с сертификатом клиента с предоставленным DN:

bin/tls-toolkit.sh standalone -n 'localhost(4)' -C 'CN=username,OU=NIFI'

Copy

Создать keystore, **truststore**, **nifi.properties** для 10 имен хостов **RT.StreamingNiFi**в каждом из 4 поддоменов:

bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain'

Copy

Создать 2 набора keystore, **truststore**, **nifi.properties** для 10 имен хостов **RT.StreamingNiFi**в каждом из 4 поддоменов вместе с сертификатом клиента с предоставленным DN:

bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain(2)' -C 'CN=username,OU=NIFI'

Copy

##### 2.4.1.2.2 Client/Server

Режим Клиент/Сервер опирается на Центр сертификации (Certificate Authority, CA) для выдачи сертификатов. Центр можно остановить, если узлы не подключены к сети.

###### 2.4.1.2.2.1 Server

Сервер CA вызывается запуском **./bin/tls-toolkit.sh -h**, который печатает информацию об использовании с описаниями опций, которые могут быть заданы.

В режиме сервера с **tls-toolkit** можно использовать следующие параметры командной строки:

* **-a, --keyAlgorithm <arg>** – алгоритм использования сгенерированных ключей (по умолчанию: **RSA**);
* **--configJsonIn <arg>** – место для чтения информации о конфигурации, подразумевает **useConfigJson**, если установлено (по умолчанию: значение **configJson**);
* **-d, --days <arg>** – количество дней, в течение которых выданный сертификат является действительным (по умолчанию: **1095**);
* **-D, --dn <arg>** – DN для сертификата CA (по умолчанию: **CN=YOUR\_CA\_HOSTNAME,OU=NIFI**);
* **-f, --configJson <arg>** – место записи информации о конфигурации (по умолчанию: **config.json**);
* **-F, --useConfigJson** – флаг, указывающий, что вся конфигурация считывается из **configJson**(для облегчения автоматического использования, иначе в **configJson**производится только запись);
* **-g, --differentKeyAndKeystorePasswords** – использование другого сгенерированного пароля для ключа и keystore;
* **-h, --help** – печать справки и выход;
* **-k, --keySize <arg>** – количество бит для генерации ключей (по умолчанию: **2048**);
* **-p, --PORT <arg>** – порт для прослушивания центром сертификации (по умолчанию: **8443**);
* **-s, --signingAlgorithm <arg>** – алгоритм подписи сертификатов (по умолчанию: **SHA256WITHRSA**);
* **-T, --keyStoreType <arg>** – тип создаваемого keystore (по умолчанию: **jks**);
* **-t, --token <arg>** – маркер для предотвращения MITM (должен быть таким же, как тот, что используется клиентами).

###### 2.4.1.2.2.1 Client

Клиент может использоваться для запроса новых сертификатов из центра сертификации. Утилита клиента генерирует пару ключей и запрос подписи сертификата (CSR, Certificate Signing Request), после чего отправляет CSR в центр сертификации. Клиент вызывается запуском **./bin/tls-toolkit.sh client -h**, который печатает информацию об использовании с описаниями опций, которые могут быть заданы.

В режиме клиента с **tls-toolkit** можно использовать следующие параметры командной строки:

* **-a, --keyAlgorithm <arg>** – алгоритм использования сгенерированных ключей (по умолчанию: **RSA**);
* **-c, --certificateAuthorityHostname <arg>** – имя хоста **RT.StreamingNiFi**Certificate Authority (по умолчанию: **localhost**);
* **-C, --certificateDirectory <arg>** – каталог записи сертификата CA (по умолчанию: **.**);
* **--configJsonIn <arg>** – место для чтения информации о конфигурации, подразумевает **useConfigJson**, если установлено (по умолчанию: значение **configJson**);
* **-D, --dn <arg>** – DN для сертификата клиента (по умолчанию: **CN=<localhost name>,OU=NIFI**, заполняется автоматически инструментом);
* **-f, --configJson <arg>** – место записи информации о конфигурации (по умолчанию: **config.json**);
* **-F, --useConfigJson** – флаг, указывающий, что вся конфигурация считывается из **configJson**(для облегчения автоматического использования, иначе в **configJson**производится только запись);
* **-g, --differentKeyAndKeystorePasswords** – использование другого сгенерированного пароля для ключа и keystore;
* **-h, --help** – печать справки и выход;
* **-k, --keySize <arg>** – количество бит для генерации ключей (по умолчанию: **2048**);
* **-p, --PORT <arg>** – порт для прослушивания центром сертификации (по умолчанию: **8443**);
* **--subjectAlternativeNames <arg>** – разделенный запятыми список доменов для использования в качестве альтернативных имен в сертификате;
* **-T, --keyStoreType <arg>** – тип создаваемого keystore (по умолчанию: **jks**);
* **-t, --token <arg>** – маркер для предотвращения MITM (должен быть таким же, как тот, что используется клиентами).

В результате запуска клиента предоставляется сертификат CA, **keystore**, **truststore**и **config.json** с информацией о них, а также их пароли.

Сертификат клиента можно легко импортировать в браузер, указав: **-T PKCS12**.

## 2.5 Аутентификация пользователя

**RT.StreamingNiFi** поддерживает аутентификацию пользователей через сертификаты клиента, через имя пользователя и пароль, через **Knox** или через [OpenId Connect](http://openid.net/connect).

Проверка подлинности имени пользователя и пароля выполняется с помощью “Идентификатора входа в систему” (“Login Identity Provider”) – это подключаемый механизм для аутентификации пользователей через их имя и пароль, настройка которого осуществляется в файле **nifi.properties**. В настоящее время **RT.StreamingNiFi**предлагает проверку имени пользователя и пароля с параметрами Provider Identity Provider для **LDAP** и **Kerberos**.

Свойство **nifi.login.identity.provider.configuration.file** указывает файл конфигурации для Идентификатора входа в систему. Свойство **nifi.security.user.login.identity.provider** указывает, какой из настроенных Login Identity Provider должен использоваться. По умолчанию свойство не настроено, что означает, что **username/password** должно быть явно включено.

При аутентификации через **OpenId Connect** сервер **RT.StreamingNiFi**перенаправляет пользователей для проверки подлинности в Провайдер, а затем **RT.StreamingNiFi**вызывает Провайдер для получения идентификации пользователя.

При аутентификации через **Knox** сервер **RT.StreamingNiFi**перенаправляет пользователей для проверки подлинности в **Knox**, а затем **RT.StreamingNiFi**во время аутентификации пользователя проверяет токен **Knox**.

**Внимание.**Аутентификация пользователя в **RT.StreamingNiFi**может быть настроена только по **username/password**, OpenId Connect или Knox. Сервер не поддерживает одновременный запуск каждого из них. При этом **RT.StreamingNiFi**потребуются сертификаты клиентов для аутентификации пользователей через HTTPS, если ничто иное не настроено.

К защищенному инстансу **RT.StreamingNiFi**нельзя получить доступ анонимно, если в **LDAP** или **Kerberos** не настроен Login Identity Provider, который, в свою очередь, должен быть настроен на явное разрешение анонимного доступа. При этом анонимный доступ в настоящее время невозможен по умолчанию в **FileAuthorizer**.

**Внимание. RT.StreamingNiFi**не выполняет аутентификацию пользователя через HTTP (используя HTTP,  всем пользователям предоставляются все роли).

### 2.5.1 Lightweight Directory Access Protocol (LDAP)

Далее приведен пример с описанием настроек Login Identity Provider, который интегрируется с Directory Server для аутентификации пользователей.

<provider>

<identifier>ldap-provider</identifier>

<class>org.apache.nifi.ldap.LdapProvider</class>

<property name="Authentication Strategy">START\_TLS</property>

<property name="Manager DN"></property>

<property name="Manager Password"></property>

<property name="TLS - Keystore"></property>

<property name="TLS - Keystore Password"></property>

<property name="TLS - Keystore Type"></property>

<property name="TLS - Truststore"></property>

<property name="TLS - Truststore Password"></property>

<property name="TLS - Truststore Type"></property>

<property name="TLS - Client Auth"></property>

<property name="TLS - Protocol"></property>

<property name="TLS - Shutdown Gracefully"></property>

<property name="Referral Strategy">FOLLOW</property>

<property name="Connect Timeout">10 secs</property>

<property name="Read Timeout">10 secs</property>

<property name="Url"></property>

<property name="User Search Base"></property>

<property name="User Search Filter"></property>

<property name="Identity Strategy">USE\_DN</property>

<property name="Authentication Expiration">12 hours</property>

</provider>

Copy

С помощью данной конфигурации аутентификация имени пользователя и пароля может быть активирована путем ссылки на провайдер в **nifi.properties**:

nifi.security.user.login.identity.provider=ldap-provider

Copy

| **Свойство** | **Описание** |
| --- | --- |
| Authentication Strategy | Аутентификация подключения к LDAP-серверу. Возможные значения: ANONYMOUS, SIMPLE, LDAPS или START\_TLS. |
| Manager DN | DN менеджера, который используется для привязки к LDAP-серверу для поиска пользователей. |
| Manager Password | Пароль менеджера, который используется для привязки к LDAP-серверу для поиска пользователей. |
| TLS - Keystore | Путь к Keystore при подключении к LDAP с использованием LDAPS или START\_TLS. |
| TLS - Keystore Password | Пароль для Keystore при подключении к LDAP с использованием LDAPS или START\_TLS. |
| TLS - Keystore Type | Тип Keystore при подключении к LDAP с использованием LDAPS или START\_TLS (то есть JKS или PKCS12). |
| TLS - Truststore | Путь к Truststore при подключении к LDAP с использованием LDAPS или START\_TLS. |
| TLS - Truststore Password | Пароль для Truststore при подключении к LDAP с использованием LDAPS или START\_TLS. |
| TLS - Truststore Type | Тип Truststore при подключении к LDAP с использованием LDAPS или START\_TLS (то есть JKS или PKCS12). |
| TLS - Client Auth | Политика аутентификации клиента при подключении к LDAP с использованием LDAPS или START\_TLS. Возможные значения: **REQUIRED**, **WANT**, **NONE**. |
| TLS - Protocol | Протокол при подключении к LDAP с использованием LDAPS или START\_TLS (TLS, TLSv1.1, TLSv1.2 и т.д.). |
| TLS - Shutdown Gracefully | Указывает, следует ли корректно завершать работу TLS перед закрытием целевого контекста. По умолчанию: **false**. |
| Referral Strategy | Стратегия обработки рефералов. Возможные значения: **FOLLOW**, **IGNORE**, **THROW**. |
| Connect Timeout | Время ожидания соединения (10 секунд). |
| Read Timeout | Время ожидания чтения (10 секунд). |
| Url | Разделенный пробелами список URL-адресов серверов LDAP (например, **ldap://<hostname>:<port>**). |
| User Search Base | Базовый DN для поиска пользователей (например, CN=Users,DC=example,DC=com). |
| User Search Filter | Фильтр для поиска пользователей в User Search Base (sAMAccountName={0}). Указанное пользователем имя вставляется в {0}. |
| Identity Strategy | Стратегия идентификации пользователей. Возможные значения: USE\_DN и USE\_USERNAME. По умолчанию: - USE\_DN (для сохранения обратной совместимости). USE\_DN использует полный DN пользовательской записи (рекомендуется). USE\_USERNAME использует имя пользователя, под которым он вошел в систему. |
| Authentication Expiration | Продолжительность действия проверки подлинности пользователя. Если пользователь никогда не выходит из системы, он должен будет снова войти в систему в течение указанного времени. |

### 2.5.2 Kerberos

Далее приведен пример с описанием настроек Login Identity Provider, который интегрируется с Kerberos Key Distribution Center (KDC) для аутентификации пользователей.

provider>

<identifier>kerberos-provider</identifier>

<class>org.apache.nifi.kerberos.KerberosProvider</class>

<property name="Default Realm">NIFI.APACHE.ORG</property>

<property name="Kerberos Config File">/etc/krb5.conf</property>

<property name="Authentication Expiration">12 hours</property>

</provider>

Copy

С помощью данной конфигурации аутентификация имени пользователя и пароля может быть активирована путем ссылки на провайдер в **nifi.properties**:

nifi.security.user.login.identity.provider=kerberos-provider

Copy

| **Свойство** | **Описание** |
| --- | --- |
| Default Realm | Область по умолчанию для предоставления пользователю в случае, если пользователь вводит неполный пользовательский принципал (например, **NIFI.APACHE.ORG**). |
| Kerberos Config File | Абсолютный путь к файлу конфигурации клиента Kerberos. |
| Authentication Expiration | Продолжительность действия проверки подлинности пользователя. Если пользователь никогда не выходит из системы, он должен будет снова войти в систему в течение указанного времени. |

### 2.5.3 OpenId Connect

Для включения аутентификации через OpenId Connect необходимо настроить свойства в **nifi.properties**, представленные далее в таблице.

| **Свойство** | **Описание** |
| --- | --- |
| **nifi.security.user.oidc.discovery.url** | URL-адрес обнаружения необходимого [OpenId Connect Provider](http://openid.net/specs/openid-connect-discovery-1_0.html). |
| **nifi.security.user.oidc.connect.timeout** | Время ожидания соединения при обмене данными с OpenId Connect Provider. |
| **nifi.security.user.oidc.read.timeout** | Время ожидания чтения при обмене данными с OpenId Connect Provider. |
| **nifi.security.user.oidc.client.id** | Идентификатор клиента для **RT.StreamingNiFi** после регистрации в OpenId Connect Provider. |
| **nifi.security.user.oidc.client.secret** | Секрет клиента для **RT.StreamingNiFi**после регистрации в OpenId Connect Provider. |
| **nifi.security.user.oidc.preferred.jwsalgorithm** | Предпочтительный алгоритм проверки токенов идентификации. Если значение свойства пустое, по умолчанию используется RS 256, поддерживаемое OpenId Connect Provider в соответствии со спецификацией. Если значение равно HS256, HS384 или HS512, **RT.StreamingNiFi**пытается проверить защищенные токены HMAC, используя указанный секретный ключ. Если значение свойства равно none, **RT.StreamingNiFi**пытается проверить незащищенные/простые токены. Иные значения для алгоритма анализируются как алгоритм RSA или EC, который используется совместно с JSON Web Key (JWK), предоставленным через jwks\_uri в метаданных URL-обнаружения. |

### 2.5.4 Knox

Для включения аутентификации через **Knox** необходимо настроить свойства в **nifi.properties**, представленные далее в таблице.

| **Свойство** | **Описание** |
| --- | --- |
| **nifi.security.user.knox.url** | URL-адрес страницы входа в Knox. |
| **nifi.security.user.knox.publicKey** | Путь к открытому ключу Knox для проверки подписей токенов аутентификации в HTTP Cookie. |
| **nifi.security.user.knox.cookieName** | Имя файла HTTP Cookie, которое Knox создает после успешного входа в систему. |
| **nifi.security.user.knox.audiences** | (Опционально) Разделенный запятыми список разрешенных подключений. Если значение задано, подключение должно присутствовать в списке. Разрешенные подключения, заполненные токеном, могут быть настроены в Knox. |

### 2.5.5 Настройка пользователей и политик доступа

В зависимости от характеристик настроенных параметров **UserGroupProvider** и **AccessPolicyProvider** пользователи, группы и политики могут конфигурироваться в пользовательском интерфейсе. Если расширения не настроены, то в пользовательском интерфейсе пользователи, группы и политики доступны только для чтения. Если сконфигурированный авторизатор не использует **UserGroupProvider** и **AccessPolicyProvider**, пользователи и политики могут быть или не быть видимыми и настраиваемыми в пользовательском интерфейсе на основе базовой реализации.

Далее в главе предполагается, что пользователи, группы и политики настраиваются в пользовательском интерфейсе, и описывается:

* Создание пользователей и групп;
* Политики доступа;
* Настройка политик доступа на основе конкретных примеров.

### 2.5.6 Создание пользователей и групп

В пользовательском интерфейсе необходимо выбрать **Users**в глобальном меню, при этом открывается диалоговое окно для создания пользователей и групп и управления ими. Для создания пользователей и групп используется кнопка **Add User**.

Для создания пользователя в новом открывшемся диалоговом окне необходимо выбрать **Individual**и ввести информацию **Identity**, соответствующую методу аутентификации защиты инстанса **RT.StreamingNiFi**. После чего нажать **ОК**.

Для создания группы в диалоговом окне следует выбрать **Group**, ввести имя группы в поле **Identity**и отметить пользователей в **Members**, которые необходимо включить в группу. После чего нажать **ОК**.

### 2.5.7 Политики доступа

Управление возможностями пользователей и групп **RT.StreamingNiFi**осуществляется с помощью политик доступа. Существует два типа политик доступа, которые могут быть применены к ресурсу:

* View (просмотр) – если ресурсу назначается политика просмотра, то добавленные в эту политику пользователи и группы могут только видеть детали данного ресурса;
* Modify (изменение) – если ресурсу назначается политика изменения, то добавленные в эту политику пользователи и группы могут изменить конфигурацию данного ресурса.

Создавать и применять политики доступа можно как на глобальном уровне, так и на уровне компонентов.

#### 2.5.7.1 Global Access Policies

Политики глобального доступа управляют следующими полномочиями на уровне системы:

| **Policy** | **Privilege** | **Global Menu Selection** | **Resource Descriptor** |
| --- | --- | --- | --- |
| view the UI | Разрешение пользователям просматривать UI. | N/A | /flow |
| access the controller | Позволяет пользователям просматривать/изменять контроллер, включая задачи отчетности, службы контроллеров и узлы в кластере. | Controller Settings | /controller |
| query provenance | Позволяет пользователям отправлять Provenance Search и запрашивать Event Lineage. | Data Provenance | /provenance |
| access restricted components | Позволяет пользователям создавать/изменять ограниченные компоненты при условии наличия других разрешений. Ограниченные компоненты могут указывать, какие конкретные разрешения требуются. Разрешения могут предоставляться для определенных ограничений или независимо от них. Если разрешение предоставляется независимо от ограничений, пользователь может создавать/изменять все ограниченные компоненты. | N/A | /restricted-components |
| access all policies | Позволяет пользователям просматривать/изменять политики для всех компонентов. | Policies | /policies |
| access users/user groups | Позволяет пользователям просматривать/изменять пользователей и группы пользователей. | Users | /tenants |
| retrieve site-to-site details | Позволяет другим инстансам **RT.StreamingNiFi**извлекать информацию site-to-site. | N/A | /site-to-site |
| view system diagnostics | Позволяет пользователям просматривать системную диагностику. | Summary | /system |
| proxy user requests | Позволяет прокси отправлять запросы от имени других пользователей. | N/A | /proxy |
| access counters | Позволяет пользователям просматривать/изменять счетчики доступа. | Counters | /counters |

#### 2.5.7.2 Component Level Access Policies

Политики доступа на уровне компонентов управляют следующими полномочиями на уровне компонентов:

| **Policy** | **Privilege** | **Resource Descriptor & Action** |
| --- | --- | --- |
| view the component | Позволяет пользователям просматривать детали конфигурации компонентов. | resource=”/<component-type>/<component-UUID>” action=”R” |
| modify the component | Позволяет пользователям изменять детали конфигурации компонентов. | resource=”/<component-type>/<component-UUID>” action=”W” |
| view provenance | Позволяет пользователям просматривать события происхождения, созданные компонентом. | resource=”/provenance-data/<component-type>/<component-UUID>” action=”R” |
| view the data | Позволяет пользователям просматривать метаданные и содержимое компонента в очередях потока в исходящих соединениях и через события происхождения. | resource=”/data/<component-type>/<component-UUID>” action=”R” |
| modify the data | Позволяет пользователям очищать очереди потоков в исходящих соединениях и повторно отправлять через события происхождения. | resource=”/data/<component-type>/<component-UUID>” action=”W” |
| view the policies | Позволяет пользователям просматривать список пользователей, которые могут просматривать/изменять компонент. | resource=”/policies/<component-type>/<component-UUID>” action=”R” |
| modify the policies | Позволяет пользователям изменять список пользователей, которые могут просматривать/изменять компонент. | resource=”/policies/<component-type>/<component-UUID>” action=”W” |
| receive data via site-to-site | Позволяет порту получать данные из инстансов **RT.StreamingNiFi**. | resource=”/data-transfer/input-ports/<port-UUID>” action=”W” |
| send data via site-to-site | Позволяет порту отправлять данные из инстансов **RT.StreamingNiFi**. | resource=”/data-transfer/output-ports/<port-UUID>” action=”W” |

**Внимание.**Политики доступа можно применять ко всем типам компонентов, кроме соединений. Разрешения на соединения определяются по индивидуальным политикам доступа к исходному и целевому компонентам соединения, а также по политике доступа группы процессов, содержащей компоненты. Более подробно это описано далее в примерах.

**Внимание.**Для доступа к **List Queue** и **Delete Queue** для соединения пользователю требуются политики “view the data” и “modify the data” на компоненте. Так же все узлы в кластерной среде должны быть добавлены к этим политикам, так как запрос пользователя может быть реплицирован через любой узел в кластере.

### 2.5.8 Настройка политик доступа на основе конкретных примеров

Самый эффективный способ понять, как создавать и применять политики доступа, – это пройтись по некоторым распространенным примерам. В приведенных далее сценариях **User1**является администратором, а **User2**– недавно добавленным пользователем, которому предоставлен доступ только к пользовательскому интерфейсу. На рисунке в качестве отправных точек показаны два процессора в рабочей области: **GenerateFlowFile**и **LogAttribute**.

**User1**может добавлять компоненты в поток данных, а также перемещать, редактировать и подключать все процессоры. Детали и свойства процессоров и групп процессов root видны для **User1**.

**User2**не может добавлять компоненты в поток данных, а также перемещать, редактировать и подключать компоненты. Детали и свойства процессоров и групп процессов root скрыты от **User2**.

#### 2.5.8.1 Перемещение процессора

**User1** необходимо выполнить следующие шаги для выдачи разрешения пользователю **User2**на перемещение процессора GenerateFlowFile в потоке данных с сохранением привилегий у **User1**:

1. Выбрать процессор GenerateFlowFile.

2. Нажать значок **Access Policies** на панели управления **Operate**. При этом открывается диалоговое окно **Access Policies**.

3. Выбрать **modify the component** в раскрывающемся списке политики. Политика **modify the component**, которая в настоящее время существует на процессоре (дочернем), является унаследованной от группы процессов root (родительской), на которой **User1** имеет привилегии.

4. Нажать ссылку **Override**. При замещении политики необходимо выбрать ее переопределение либо на копию унаследованной политики, либо на пустую политику. Для создания копии следует в диалоговом окне **Override Policy** выбрать **Copy**и нажать кнопку **Override**.

5. В созданной политике выбрать значок **Add User**. В поле **User Identity** ввести вручную или найти в списке **User2**и нажать **OK**.

6. С такими изменениями **User1**сохраняет возможность перемещения обоих процессоров в рабочей области. А **User2**теперь может перемещать процессор GenerateFlowFile.

#### 2.5.8.2 Изменение процессора

В приведенном примере “Перемещение процессора” **User2**добавлен в политику **modify the component** для процессора GenerateFlowFile. Но без возможности просмотра свойств процессора **User2**не может изменять его конфигурацию – чтобы отредактировать компонент, пользователь должен быть также включен в политику **view the component**.

**User1**необходимо выполнить следующие шаги для реализации возможности изменения конфигурации процессора пользователю **User2**:

1. Выбрать процессор GenerateFlowFile.

2. Нажать значок **Access Policies** на панели управления **Operate**. При этом открывается диалоговое окно **Access Policies**.

3. Выбрать **view the component** в раскрывающемся списке политики. Политика **view the component**, которая в настоящее время существует на процессоре (дочернем), является унаследованной от группы процессов root (родительской), на которой **User1**имеет привилегии.

4. Нажать ссылку **Override**и в открывшемся диалоговом окне, сохранив политику копирования по умолчанию, нажать кнопку **Override**.

5. В созданной политике выбрать значок **Add User**. В поле **User Identity** ввести вручную или найти в списке **User2**и нажать **OK**.

С такими изменениями **User1**сохраняет возможность просмотра и редактирования процессоров в рабочей области. А **User2**теперь может просматривать и редактировать процессор GenerateFlowFile.

#### 2.5.8.3 Создание подключения

При настройке политик так, как описано в предыдущих двух примерах, **User1**может подключить GenerateFlowFile к LogAttribute.

При этом **User2**не имеет права доступа на установку соединения процессоров.

Это объясняется тем, что:

* **User2**не имеет доступа к изменениям в группе процессов;
* Несмотря на то, что **User2**имеет право на просмотр и изменение исходного компонента (GenerateFlowFile), **User2**не имеет политики доступа к целевому компоненту (LogAttribute).

**User1**необходимо выполнить следующие шаги для реализации возможности подключения GenerateFlowFile к LogAttribute пользователю **User2**:

1. Выбрать группу процессов root, при этом панель управления **Operate**обновляется с подробными сведениями.

2. Выбрать значок **Access Policies** на панели управления **Operate**. При этом открывается диалоговое окно **Access Policies**.

3. В диалоговом окне в раскрывающемся списке политики выбрать **modify the component**.

4. Выбрать значок **Add User**. В поле **User Identity** ввести вручную или найти в списке **User2**и нажать **OK**.

Добавляя **User2**в политику **modify the component** группы процессов, **User2**так же добавляется к политике **modify the component** в процессоре LogAttribute путем наследования. Чтобы проверить это, необходимо в рабочей области выделить процессор LogAttribute и выбрать значок **Access Policies** на панели управления **Operate**. При этом открывается диалоговое окно политик доступа процессора LogAttribute с наличием пользователя **User2**в политике **modify the component**.

С такими изменениями **User2**теперь может подключать процессор GenerateFlowFile к процессору LogAttribute.

#### 2.5.8.4 Изменение соединения

В следующем сценарии **User1**и **User2**добавляют процессор ReplaceText в группу процессов root.

**User1**может выбрать и изменить существующее соединение между GenerateFlowFile и LogAttribute, чтобы подключить GenerateFlowFile к ReplaceText.

При этом **User2**не имеет возможности выполнить такое действие.

**User1**необходимо выполнить следующие шаги для реализации возможности подключения GenerateFlowFile к ReplaceText пользователю **User2**:

1. Выбрать группу процессов root, при этом панель управления **Operate**обновляется с подробными сведениями.

2. Выбрать значок **Access Policies** на панели управления **Operate**. При этом открывается диалоговое окно **Access Policies**.

3. В диалоговом окне в раскрывающемся списке политики выбрать **view the component**.

4. Выбрать значок **Add User**. В поле **User Identity** ввести вручную или найти в списке **User2** и нажать **OK**.

Будучи добавленным к политикам просмотра и изменения для группы процессов **User2**теперь может подключать процессор GenerateFlowFile к процессору ReplaceText.

## 2.6 Kerberos Service

**RT.StreamingNiFi**может быть настроен для использования Kerberos SPNEGO (или “Kerberos Service”) для аутентификации. В таком случае пользователи попадают в конечную точку REST **/access/kerberos**, и сервер отвечает кодом состояния 401 с заголовком задачи **WWW-Authenticate: Negotiate**. Далее сервер связывается с браузером для использования GSS-API и загрузки тикета пользователя Kerberos с указанием его в качестве заголовка Base64 в последующем запросе. Он принимает форму **Authorization: Negotiate YII…**, и **RT.StreamingNiFi**пробует подтвердить этот билет с помощью KDC. В случае успеха принципал пользователя возвращается как подлинный, и поток следует аутентификации login/credential, в результате которой в ответ выдается JWT для предотвращения ненужных издержек аутентификации Kerberos при каждом последующем запросе. В случае если билет пользователя не подтверждается, то он возвращается с соответствующим кодом ошибки. После чего пользователь может предоставить свои учетные данные для формы регистрации Kerberos при условии настроенного **KerberosLoginIdentityProvider**.

**RT.StreamingNiFi**отвечает на запросы Kerberos SPNEGO только по соединению HTTPS, поскольку незащищенные запросы никогда не проходят проверку подлинности.

Для включения аутентификации службы Kerberos в **nifi.properties** должны быть настроены следующие свойства:

| **Свойство** | **Значение** | **Описание** |
| --- | --- | --- |
| Service Principal | true | Принципал сервиса, используемый **RT.StreamingNiFi**для связи с KDC. |
| Keytab Location | true | Путь к файлу keytab, содержащему принципал сервиса. |

**Примечания:**

* Kerberos чувствителен к регистру во многих местах, и сообщения об ошибках (или их отсутствие) могут быть недостаточно понятны. Рекомендуется проверить у службы чувствительность к регистру в конфигурационных файлах. Конвенция – **HTTP/fully.qualified.domain@REALM**;
* Браузеры имеют разные уровни ограничений при работе с SPNEGO. Некоторые из них предоставляют локальный тикет Kerberos в любой запрашиваемый домен, в то время как другие выдают пустой список доверенных доменов. Справочная информация для общих браузеров приведена в Приложении 1;
* Некоторые браузеры (например, устаревший IE) не поддерживают последние алгоритмы шифрования, такие как AES, и ограничены устаревшими алгоритмами (например, DES). Это следует учитывать при создании keytabs;
* Должен быть настроен KDC, определен принципал сервиса для **RT.StreamingNiFi**и экспортирован keytab. Подробные инструкции по настройке и администрированию Kerberos Service выходят за рамки данного документа ([MIT Kerberos Admin Guide](http://web.mit.edu/kerberos/krb5-current/doc/admin/index.html)), но далее приведен пример.

Пример добавления принципала для сервера на **nifi.nifi.apache.org** и экспорта ключа из KDC:

root@kdc:/etc/krb5kdc# kadmin.local

Authenticating as principal admin/admin@NIFI.APACHE.ORG with password.

kadmin.local: listprincs

K/M@NIFI.APACHE.ORG

admin/admin@NIFI.APACHE.ORG

...

kadmin.local: addprinc -randkey HTTP/nifi.nifi.apache.org

WARNING: no policy specified for HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG; defaulting to no policy

Principal "HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG" created.

Principal "HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG" created.

kadmin.local: ktadd -k /http-nifi.keytab HTTP/nifi.nifi.apache.org

Entry for principal HTTP/nifi.nifi.apache.org with kvno 2, encryption type des3-cbc-sha1 added to keytab WRFILE:/http-nifi.keytab.

Entry for principal HTTP/nifi.nifi.apache.org with kvno 2, encryption type des-cbc-crc added to keytab WRFILE:/http-nifi.keytab.

kadmin.local: listprincs

HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG

K/M@NIFI.APACHE.ORG

admin/admin@NIFI.APACHE.ORG

...

kadmin.local: q

root@kdc:~# ll /http\*

-rw------- 1 root root 162 Mar 14 21:43 /http-nifi.keytab

root@kdc:~#

Copy

## 2.7 Удаление/Добавление компонентов сервиcа ****RT.StreamingNiFi****

Доступно с версии 1.4.11.

**Внимание.**Описанные ниже операции не удаляют/добавляют хост из кластера – они лишь управляют компонентом NiFi Server и NiFi-Registry на хостах. Удаление хоста из кластера возможно в разделе “Hosts” кластера при условии, что к хосту не привязан ни один компонент.

Для добавления или удаления **RT.StreamingNiFi**с хостов необходимо воспользоваться соответствующими кнопками выпадающего меню, доступного по нажатию на иконку в поле “Actions” сервиса **RT.StreamingNiFi**.

### 2.7.1 Добавление компонентов Nifi Server и Nifi-Registry

Когда хосты становятся доступными для подключения по ssh для менеджера кластеров, необходимо выбрать действие **Expand**cервиса **RT.StreamingNiFi**из списка возможных операций. В появившемся диалоговом окне предоставляется выбор опций:

* Disable SELinux before cluster installation – отключение SELinux на добавляемых хостах. Для того, чтобы данная настройка применилась, после завершения операции Expand необходимо перезагрузить хосты вручную;
* Disable Firewalld before cluster installation – выключение firewalld на добавляемых хостах;
* Install OpenJDK before cluster installation – установка пакета **java-1.8.0-openjdk** на добавляемых хостах;
* Set vm.swappiness to 0 for all hosts – отключение swapping на добавляемых хостах;
* Append hosts into /etc/hosts file before cluster installation – запись добавляемых нод в **/etc/hosts** на всех хостах кластера. Данную опцию рекомендуется отключить, если настроен DNS.

После выбора опций для перехода к следующей странице конфигурации следует нажать кнопку “Next”, и в открывшейся форме необходимо распределить компонент Nifi Server по добавляемым хостам. Также есть возможность установить Nifi-Registry, если ранее он не был установлен. В случае если используется сервис Monitoring Clinets, его компоненты также необходимо разместить на добавляемых хостах.

Расширение сервиса запускается кнопкой **Run**. На добавленные хосты устанавливаются необходимые пакеты и производится их настройка. Текущая конфигурация Flow, представленная в **flow.xml.gz**, копируется на новый хост.

### 2.7.2 Удаление Nifi Server

Для удаления одного или нескольких **Nifi Server** с хостов кластера необходимо:

1. Выбрать действие **Shrink**cервиса **RT.StreamingNiFi**из списка возможных операций, что приводит к появлению окна распределения компонента по хостам;

2. Любым из двух способов удалить привязку компонента к хосту (компонент **Nifi Server** выделяется белым цветом, как возможный к удалению с хостов):

* Выбрать компонент в колонке **Components**и убрать выделение с хостов в колонке **Hosts**, рамки которых выделены зеленым цветом;
* Выбрать хост в колонке **Hosts**и убрать выделение с компонента **Nifi Server** в колонке **Components**, если рамка компонента **Nifi Server** выделяется зеленым цветом.

3. Нажать кнопку **Run**в нижней части окна.

**Внимание.**Описанная процедура не удаляет данные и пакет **RT.StreamingNiFi**c хоста – онa лишь выводит ноду из кластера **RT.StreamingNiFi**.

# 3 Интеграция сервисов MiNiFi и RT.StreamingNiFi

**RT.StreamingNiFi** предоставляет, помимо **RT.StreamingNiFi**и **MiNiFi**, поддержку централизованного управления **MiNiFi Agent** с помощью **MiNiFi C2 Server**. Данный сервис обеспечивает автоматическое обновление конфигураций **MiNiFi Agent** без сторонних вспомогательных средств. В данном разделе приведены основные шаги для настройки взаимодействия между **MiNiFi**и **RT.StreamingNiFi**сервисами:

* создание шаблона;
* проверка конфигурации.

## 3.1 Создание шаблона

Для выполнения какой-либо задачи **MiNiFi Agent** необходимо создать шаблон в интерфейсе **RT.StreamingNiFi**. В данном разделе представлен элементарный шаблон для сбора содержимого файла с машин **MiNiFi Agent**.

После установки **RT.StreamingNiFi**и **MiNiFi**, в разделе **Template**появляется шаблон с название **simple-minifi-listener**, который состоит из следующих элементов.

Чтобы создать шаблон для Агентов, необходимо перейти в **MiNiFi Process Group** и создать **Flow**, который будет выполнятся непосредственно **MiNiFi Agent**.

В нашем случае созданный **Flow**содержит процессор **TailFile**, который считывает содержимое файла и передает инстансу **RT.StreamingNiFi**.

Для успешной загрузки **Flow**на **MiNiFi Agent**, необходимо сохранить шаблон с названием указанным в **nifi.minifi.notifier.ingestors.pull.http.query** с добавлением версии (например, **minifi.v1**). Если вы изменили **Flow**, то для актуализации его на агентах необходимо увеличить версию шаблона (например, **minifi.v2**).

Автоматическое обновление конфигурации **MiNiFi Agent** происходит с периодичностью, заданной **nifi.minifi.notifier.ingestors.pull.http.period.ms**. Если шаблон был неправильно собран, то Агенты продолжат работу на последней работоспособной конфигурации.

## 3.2 Проверка конфигурации

Текущую конфигурацию **Flow**, которую запрашивают **MiNiFi Agent** у **MiNiFi C2 Server**, можно проверить, обратившись к API **MiNiFi C2 Server**.

# 4 Обзор RT.StreamingNiFi Toolkit

Набор инструментов **RT.StreamingNiFi**(**RT.StreamingNiFi Toolkit**) содержит несколько утилит командной строки (CLI) для настройки и поддержки **RT.StreamingNiFi**в автономных и кластеризованных средах. Утилиты включают в себя:

* CLI — Инструмент cli позволяет администраторам взаимодействовать с инстансами **RT.StreamingNiFi**и **NiFi Registry** для автоматизации таких задач, как развертывание версий потоков и управление группами процессов и узлами кластера.
* Encrypt Config — инструмент **encrypt-config** шифрует конфиденциальные ключи в файле**nifi.properties**, чтобы облегчить настройку защищенного инстанса **RT.StreamingNiFi**.
* File Manager — инструмент File Manager позволяет администраторам создавать резервные копии, устанавливать или восстанавливать **RT.StreamingNiFi**из резервной копии.
* Flow Analyzer — инструмент **flow-analyzer** создает отчет, который помогает администраторам понять максимальный объем данных, который может быть сохранен при обратном давлении (backpressure) для данного потока.
* Node Manager — инструмент Node Manager позволяет администраторам выполнять проверку состояния нод, а также возможность подключать, отключать или удалять ноды из кластера.
* Notify — инструмент Notify позволяет администраторам отправлять bulletins в интерфейс **RT.StreamingNiFi**.
* S2S — инструмент **s2s**позволяет администраторам отправлять или получать данные **RT.StreamingNiFi**flows site-to-site.
* TLS Toolkit — утилита **tls-toolkit** генерирует необходимые keystores, truststore и соответствующие конфигурационные файлы для облегчения настройки защищенного инстанса **RT.StreamingNiFi**.
* ZooKeeper Migrator — инструмент **zk-migrator** позволяет администраторам:
  + переместить информацию ZooKeeper из одного кластера ZooKeeper в другой.
  + перенести права ноды ZooKeeper.

# Приложение 1. Настройка браузеров для согласования SPNEGO

**E. 1 Firefox**

Выполните следующие действия, чтобы убедиться, что ваш браузер Firefox включен для выполнения проверки подлинности Spnego:

* Откройте Firefox.
* В поле адрес введите **about:config**.
* В поле фильтр/поиск введите **negotiate**.
* Параметр **network.negotiate-auth.trusted-uris** может быть установлен по умолчанию **https://**, который не работает для вас. Вообще говоря, этот параметр должен быть заменен адресом сервера, если требуется делегирование Kerberos.
* Рекомендуется использовать https для всех коммуникаций.

**Е. 2 Chrome**

С Google Chrome вам, как правило, нужно установить параметры командной строки, чтобы серверы белого списка с Chrome договорились.

* на машинах Windows (клиентах): Chrome разделяет конфигурацию с Internet Explorer, поэтому, если все изменения были применены к IE (как описано в E. 3), ничего не должно передаваться через параметры командной строки.
* на машинах (клиентах) Linux/Mac OS: параметр командной строки **--auth-negotiate-delegate-whitelist** должен использоваться только в том случае, если требуется делегирование Kerberos (в противном случае не устанавливайте этот параметр).
* Рекомендуется использовать https для всех коммуникаций.

--auth-server-whitelist="\*.example.com"

--auth-negotiate-delegate-whitelist="\*.example.com"

Copy

Вы можете увидеть, какие политики включены, набрав **chrome://policy/** в адресной строке Chrome.

С Linux Chrome также будет читать файлы политики из каталога **/etc/opt/chrome/policies/managed**.

**mypolicy.json.**

{

"AuthServerWhitelist" : "\*.example.org",

"AuthNegotiateDelegateWhitelist" : "\*.example.org",

"DisableAuthNegotiateCnemeLookup" : true,

"EnableAuthNegotiatePort" : true

}

Copy

**E. 3 Internet Explorer**

Выполните следующие действия, чтобы убедиться, что браузер Internet Explorer включен для выполнения проверки подлинности Spnego.

* Откройте Internet Explorer.
* Выберите **Инструменты > Параметры Intenet >> Безопасность>>** вкладка.
* В разделе **Локальная интрасеть** убедитесь, что ваш сервер является доверенным, т.е. добавьте его в список.

# 1 Введение

## 1.1 Наименование и условное обозначение

Наименование системы: **RT.StreamingKafka**.

## [¶](https://docs.data.rt.ru/ru/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F/RT_StreamingKafka/rt_streamingkafka_admin_manual#h-12-%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F)1.2 Область применения

**RT.StreamingKafka** – распределённый программный брокер сообщений, который разрабатывается на языках Java и Scala.

# 2 Конфигурирование RT.StreamingKafka

**RT.StreamingKafka** использует пары ключ-значение в формате файла свойств для конфигурации. Эти значения могут быть предоставлены либо из файла, либо программно.

## 2.1 Конфигурации брокера

Основные конфигурации следующие:

broker.id

log.dirs

zookeeper.connect

Copy

Конфигурации и значения по умолчанию уровня топика содержатся в **п. 2.2**.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** | **Режим обновления** |
| --- | --- | --- | --- | --- | --- | --- |
| **advertised.listeners** | Слушатели для публикации в ZooKeeper для использования клиентами, если они отличаются от свойства конфигурации **listeners**. В средах IaaS они могут отличаться от интерфейса, к которому привязывается брокер. Если этот параметр не установлен, будет использоваться значение для **listeners**. В отличие от **listeners**, определять метаадрес 0.0.0.0 недопустимо.  Кроме того, в отличие от **listeners**, в этом свойстве могут дублироваться порты, так что один слушатель можно настроить для объявления адреса другого слушателя. Это может быть полезно в некоторых случаях, когда используются внешние балансировщики нагрузки. | string | null | -- | высокая | для каждого брокера |
| **auto.create.topics.enable** | Включает автосоздание топика на сервере. | boolean | true | -- | высокая | только для чтения |
| **auto.leader.rebalance.enable** | Включает автоматическую балансировку лидеров. Фоновый поток проверяет распределение лидеров разделов через регулярные промежутки времени, настраиваемые с помощью **leader.imbalance.check.interval.seconds**. Если дисбаланс лидеров превышает **leader.imbalance.per.broker.percentage**, запускается перебалансировка лидеров до предпочтительного лидера для партиций. | boolean | true | -- | высокая | только для чтения |
| **background.threads** | Количество потоков, используемых для различных задач фоновой обработки. | int | 10 | [1,...] | высокая | для всего кластера |
| **broker.id** | Идентификатор брокера для этого сервера. Если этот параметр не установлен, будет сгенерирован уникальный идентификатор брокера. Чтобы избежать конфликтов между идентификаторами брокера, сгенерированными Zookeeper, и идентификаторами брокера, настроенными пользователем, генерируемые идентификаторы брокера начинаются с **reserved.broker.max.id + 1**. | int | -1 | -- | высокая | только для чтения |
| **compression.type** | Укажите окончательный тип сжатия для данного топика. Эта конфигурация поддерживает стандартные кодеки сжатия (**gzip**, **snappy**, **lz4**, **zstd**). Кроме того, он принимает **uncompressed**, что эквивалентно отсутствию сжатия; и **producer**, что означает сохранение исходного кодека сжатия, установленного поставщиком. | string | producer | [uncompressed, zstd, lz4, snappy, gzip, producer] | высокая | для всего кластера |
| **control.plane.listener.name** | Имя слушателя, используемого для связи между контроллером и брокерами. Брокер будет использовать **control.plane.listener.name** для поиска эндпоинта в списке слушателей для прослушивания соединений от контроллера. Например, если конфигурация брокера:  **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**  при запуске брокер начнет прослушивать **192.1.1.8:9094** с протоколом безопасности **SSL**.  На стороне контроллера, когда он обнаруживает опубликованные эндпоинты брокера через Zookeeper, он будет использовать **control.plane.listener.name** для поиска эндпоинта, который он будет использовать для установления соединения с брокером.  Например, если опубликованные эндпоинты брокера на Zookeeper:  **"endpoints" : ["INTERNAL://broker1.example.com:9092"," EXTERNAL://broker1.example.com:9093","CONTROLLER: //broker1.example.com:9094"]**  и конфигурация контроллера:  **listener.security.protocol.map = INTERNAL:PLAINTEXT, EXTERNAL:SSL, CONTROLLER:SSL** **control.plane.listener.name = CONTROLLER**  тогда контроллер будет использовать **broker1.example.com:9094** с протоколом безопасности **SSL**для подключения к брокеру.  Если параметр не настроен явно, значение по умолчанию будет **null**, и для подключений контроллера не будет выделенных эндпоинтов.  Если параметр настроен явно, значение не может совпадать со значением **inter.broker.listener.name**. | string | null | -- | высокая | только для чтения |
| **controller.listener.names** | Разделённый запятыми список имён слушателей, используемых контроллером. Параметр необходим при работе в режиме KRaft. При общении с кворумом контроллера брокер всегда будет использовать первого слушателя в этом списке.  **Примечание**. Контроллеру на базе ZK не следует устанавливать эту конфигурацию. | string | null | -- | высокая | только для чтения |
| **controller.quorum.election.backoff.max.ms** | Максимальное время в миллисекундах до начала нового выбора. Параметр используется в механизме бинарной экспоненциальной отсрочки, который помогает предотвратить блокировки выбора. | int | 1000 (1 секунда) | -- | высокая | только для чтения |
| **controller.quorum.election.timeout.ms** | Максимальное время в миллисекундах ожидания без возможности получить данные от лидера, прежде чем инициировать новый выбор. | int | 1000 (1 секунда) | -- | высокая | только для чтения |
| **controller.quorum.fetch.timeout.ms** | Максимальное время без успешного получения информации от текущего лидера, прежде чем он станет кандидатом и будет инициирован выбор избирателей. Максимальное время без получения запроса от большинства кворума, прежде чем спросить, наступит ли новая эпоха для лидера. | int | 2000 (2 секунды) | -- | высокая | только для чтения |
| **controller.quorum.voters** | Маппинг информации об идентификаторе/эндпоинте для набора избирателей в списке записей **`{id}@{host}:{port}`**, разделённых запятыми. Например: **`1@localhost:9092,2@localhost:9093,3@localhost:9094`**. | list | "" | не пустой список | высокая | только для чтения |
| **delete.topic.enable** | Позволяет удалить топик. Удаление топика через инструмент администратора не будет иметь никакого эффекта, если эта конфигурация отключена. | boolean | true | -- | высокая | только для чтения |
| **early.start.listeners** | Список имён слушателей, разделённых запятыми, которые могут быть запущены до того, как авторизатор завершит инициализацию. Это полезно, когда авторизатор зависит от самого кластера при начальной загрузке, как в случае со StandardAuthorizer (который хранит списки ACL в логе метаданных). По умолчанию все слушатели, включённые в **controller.listener.names**, также будут слушателями раннего запуска. Слушатель не должен появляться в этом списке, если он принимает внешний трафик. | string | null | -- | высокая | только для чтения |
| **leader.imbalance.check.interval.seconds** | Частота, с которой контроллер запускает проверку ребалансировки партиций. | long | 300 | [1,...] | высокая | только для чтения |
| **leader.imbalance.per.broker.percentage** | Допускаемый коэффициент дисбаланса лидеров на одного брокера. Контроллер активирует баланс лидеров, если он превысит это значение для каждого брокера. Значение указывается в процентах. | int | 10 | -- | высокая | только для чтения |
| **listeners** | Список слушателей — разделённый запятыми список URI, которые мы будем прослушивать, и имена слушателей. Если имя слушателя не удовлетворяет протоколу безопасности, также необходимо установить **listener.security.protocol.map**.  Имена слушателей и номера портов должны быть уникальными.  Укажите имя хоста как 0.0.0.0 для привязки ко всем интерфейсам.  Оставьте имя хоста пустым, чтобы привязать его к интерфейсу по умолчанию.  Примеры валидных списков слушателей:  **PLAINTEXT://myhost:9092,SSL://:9091** **CLIENT://0.0.0.0:9092,REPLICATION://localhost:9093** | string | PLAINTEXT://:9092 | -- | высокая | для каждого брокера |
| **log.dir** | Каталог, в котором хранятся данные лога (дополнительно к свойству **log.dirs**). | string | /tmp/kafka-logs | -- | высокая | только для чтения |
| **log.dirs** | Разделённый запятыми список каталогов, в которых хранятся данные лога. Если параметр не установлен, используется значение из **log.dir**. | string | null | -- | высокая | только для чтения |
| **log.flush.interval.messages** | Количество сообщений, накопленных в партиции лога, прежде чем сообщения будут сброшены на диск. | long | 9223372036854775807 | [1,...] | высокая | для всего кластера |
| **log.flush.interval.ms** | Максимальное время в мс, в течение которого сообщение в любом топике хранится в памяти перед записью на диск. Если параметр не установлен, используется значение из **log.flush.scheduler.interval.ms**. | long | null | -- | высокая | для всего кластера |
| **log.flush.offset.checkpoint.interval.ms** | Частота, с которой мы обновляем постоянную запись последней очистки, которая действует как точка восстановления лога. | int | 60000 (1 минута) | [0,...] | высокая | только для чтения |
| **log.flush.scheduler.interval.ms** | Частота в мс, с которой средство очистки лога проверяет, нужно ли сбрасывать какой-либо лог на диск. | long | 9223372036854775807 | -- | высокая | только для чтения |
| **log.flush.start.offset.checkpoint.interval.ms** | Частота, с которой мы обновляем постоянную запись смещения начала лога. | int | 60000 (1 минута) | [0,...] | высокая | только для чтения |
| **log.retention.bytes** | Максимальный размер лога перед его удалением. | long | -1 | -- | высокая | для всего кластера |
| **log.retention.hours** | Количество часов хранения файла лога перед его удалением (в часах), третичное значение свойства **log.retention.ms**. | int | 168 | -- | высокая | только для чтения |
| **log.retention.minutes** | Количество минут хранения файла лога перед его удалением (в минутах), вторичное по отношению к свойству **log.retention.ms**. Если параметр не установлен, используется значение в **log.retention.hours**. | int | null | -- | высокая | только для чтения |
| **log.retention.ms** | Количество миллисекунд, в течение которого файл лога будет храниться перед его удалением (в миллисекундах). Если параметр не установлен, используется значение в **log.retention.minutes**. Если установлено значение **-1**, ограничение по времени не применяется. | long | null | -- | высокая | для всего кластера |
| **log.roll.hours** | Максимальное время до развёртывания нового сегмента лога (в часах), вторичное по отношению к свойству **log.roll.ms**. | int | 168 | [1,...] | высокая | только для чтения |
| **log.roll.jitter.hours** | Максимальный джиттер, который нужно вычесть из **logRollTimeMillis**(в часах), вторичный по отношению к свойству **log.roll.jitter.ms**. | int | 0 | [0,...] | высокая | только для чтения |
| **log.roll.jitter.ms** | Максимальный джиттер, который нужно вычесть из **logRollTimeMillis**(в миллисекундах). Если параметр не установлен, используется значение в **log.roll.jitter.hours**. | long | null | -- | высокая | для всего кластера |
| **log.roll.ms** | Максимальное время до развёртывания нового сегмента лога (в миллисекундах). Если не установлено, используется значение в **log.roll.hours**. | long | null | -- | высокая | для всего кластера |
| **log.segment.bytes** | Максимальный размер одного файла лога. | int | 1073741824 (1 гигибайт) | [14,...] | высокая | для всего кластера |
| **log.segment.delete.delay.ms** | Время ожидания перед удалением файла из файловой системы. | long | 60000 (1 минута) | [0,...] | высокая | для всего кластера |
| **message.max.bytes** | Максимальный размер пакета записей, разрешённый RT.StreamingKafka (после сжатия, если сжатие включено). Если это значение увеличено и есть потребители с версией старше 0.10.2, размер выборки потребителей также должен быть увеличен, чтобы они могли получать такие большие пакеты записей. В последней версии формата сообщений записи всегда группируются в пакеты для повышения эффективности. В предыдущих версиях формата сообщений несжатые записи не группируются в пакеты, и в этом случае это ограничение применяется только к одной записи. Параметр можно установить для каждого топика с помощью конфигурации **max.message.bytes** на уровне топика. | int | 1048588 | [0,...] | высокая | для всего кластера |
| **metadata.log.dir** | Эта конфигурация определяет, куда мы поместим лог метаданных для кластеров в режиме KRaft. Если параметр не установлен, лог метаданных размещается в первом каталоге логов из **log.dirs**. | string | null | -- | высокая | только для чтения |
| **metadata.log.max.record.bytes.between.snapshots** | Это максимальное количество байтов в логе между последним снапшотом и high watermark (максимальное доступное смещение для чтения), необходимое для создания нового снапшота. Значение по умолчанию — **20971520**. Чтобы создавать снапшоты на основе затраченного времени, смотри конфигурацию **metadata.log.max.snapshot.interval.ms**. Нода RT.StreamingKafka создаст снапшот при достижении максимального интервала времени или максимального лимита в байтах. | long | 20971520 | [1,...] | высокая | только для чтения |
| **metadata.log.max.snapshot.interval.ms** | Это максимальное количество миллисекунд ожидания для создания снапшота, если в логе есть зафиксированные записи, которые не включены в снапшот. Значение **0** отключает создание снапшотов  на основе времени. Значение по умолчанию — **3600000**. Чтобы создавать снапшоты на основе количества байтов метаданных, см. конфигурацию **metadata.log.max.record.bytes.between.snapshots**. Нода RT.StreamingKafka создаст снапшот при достижении максимального интервала времени или максимального лимита в байтах. | long | 3600000 (1 час) | [0,...] | высокая | только для чтения |
| **metadata.log.segment.bytes** | Максимальный размер одного файла лога метаданных. | int | 1073741824 (1 гигибайт) | [12,...] | высокая | только для чтения |
| **metadata.log.segment.ms** | Максимальное время перед развёртыванием нового файла лога метаданных (в миллисекундах). | long | 604800000 (7 дней) | -- | высокая | только для чтения |
| **metadata.max.retention.bytes** | Максимальный совокупный размер лога метаданных и снапшотов перед удалением старых снапшотов и файлов логов. Поскольку для удаления логов должен существовать хотя бы один снапшот, это "мягкое" ограничение. | long | 104857600 (100 мебибайт) | -- | высокая | только для чтения |
| **metadata.max.retention.ms** | Количество миллисекунд, в течение которых файл лога метаданных или снапшот будут храниться перед его удалением. Поскольку для удаления логов должен существовать хотя бы один снапшот, это “мягкое” ограничение. | long | 604800000 (7 дней) | -- | высокая | только для чтения |
| **min.insync.replicas** | Когда поставщик устанавливает для **acks**значение **all**(или **-1**), **min.insync.replicas** определяет минимальное количество реплик, которые должны подтвердить запись, чтобы запись считалась успешной. Если этот минимум не может быть достигнут, поставщик выдаст исключение (либо **NotEnoughReplicas**, либо **NotEnoughReplicasAfterAppend**).  При совместном использовании **min.insync.replicas** и **acks**вы можете обеспечить более высокие гарантии надёжности. Типичным сценарием будет создание топика с фактором репликации 3, установка **min.insync.replicas** равным **2** и установка для **acks**значения **all**. Это гарантирует, что поставщик вызовет исключение, если большинство реплик не получат запись. | int | 1 | [1,...] | высокая | для всего кластера |
| **node.id** | Идентификатор ноды, связанной с ролями, которые играет этот процесс, если **process.roles** не пуст. Это необходимая настройка при работе в режиме KRaft. | int | -1 | -- | высокая | только для чтения |
| **num.io.threads** | Количество потоков, которые сервер использует для обработки запросов, которые могут включать дисковый ввод-вывод. | int | 8 | [1,...] | высокая | для всего кластера |
| **num.network.threads** | Количество потоков, которые сервер использует для получения запросов из сети и отправки ответов в сеть. | int | 3 | [1,...] | высокая | для всего кластера |
| **num.recovery.threads.per.data.dir** | Количество потоков на каталог данных, которые будут использоваться для восстановления лога при запуске и для его очистки при завершении работы. | int | 1 | [1,...] | высокая | для всего кластера |
| **num.replica.alter.log.dirs.threads** | Количество потоков, которые могут перемещать реплики между каталогами логов, включая дисковый ввод-вывод. | int | null | -- | высокая | только для чтения |
| **num.replica.fetchers** | Количество потоков выборки, используемых для репликации записей от каждого исходного брокера. Общее количество сборщиков на каждом брокере ограничивается числом **num.replica.fetchers**, умноженным на количество брокеров в кластере. Увеличение этого значения может увеличить степень параллелизма ввода-вывода в ведомом и ведущем брокере за счёт более высокой загрузки ЦП и использования памяти. | int | 1 | -- | высокая | для всего кластера |
| **offset.metadata.max.bytes** | Максимальный размер записи метаданных, связанной с коммитом смещения. | int | 4096 (4 кибибайт) | -- | высокая | только для чтения |
| **offsets.commit.required.acks** | Необходимые **ack**перед коммитом могут быть приняты. В общем, значение по умолчанию (**-1**) не следует переопределять. | short | -1 | -- | высокая | только для чтения |
| **offsets.commit.timeout.ms** | Коммит смещения будет отложен до тех пор, пока все реплики для топика смещений не получат коммит или пока не истечёт этот таймаут. Этот параметр похож на таймаут запроса поставщика. | int | 5000 (5 секунд) | [1,...] | высокая | только для чтения |
| **offsets.load.buffer.size** | Размер пакета для чтения из сегментов смещений при загрузке смещений в кэш ("мягкое" ограничение, переопределяемое, если записи слишком велики). | int | 5242880 | [1,...] | высокая | только для чтения |
| **offsets.retention.check.interval.ms** | Частота проверки устаревших смещений. | long | 600000 (10 минут) | [1,...] | высокая | только для чтения |
| **offsets.retention.minutes** | Для подписанных потребителей срок действия закоммиченного смещения определённой партиции истечёт и будет отброшен, когда:   * этот период хранения истечёт после того, как группа потребителей потеряет всех своих потребителей (т.е. станет пустой); * этот срок хранения истёк с момента последнего коммита смещения для партиции и группа больше не подписана на соответствующий топик.   Для standalone-потребителей (использующих назначение вручную) срок действия смещений истечёт по истечении срока хранения с момента последнего коммита. Обратите внимание, что когда группа удаляется с помощью запроса на удаление группы, её закоммиченные смещения также будут удалены без дополнительного периода хранения. Кроме того, когда топик удаляется с помощью запроса на удаление топика, при обновлении распространяемых метаданных закоммиченные смещения любой группы для этого топика также будут удалены без дополнительного периода хранения. | int | 10080 | [1,...] | высокая | только для чтения |
| **offsets.topic.compression.codec** | Кодек сжатия для топика смещений — сжатие можно использовать для архивирования «атомарных» коммитов. | int | 0 | -- | высокая | только для чтения |
| **offsets.topic.num.partitions** | Количество партиций для топика коммита смещения (не должно меняться после развёртывания). | int | 50 | [1,...] | высокая | только для чтения |
| **offsets.topic.replication.factor** | Фактор репликации для топика смещений (установите более высокое значение, чтобы обеспечить доступность). Создание внутреннего топика не будет выполнено до тех пор, пока размер кластера не будет соответствовать этому требованию фактора репликации. | short | 3 | [1,...] | высокая | только для чтения |
| **offsets.topic.segment.bytes** | Байты сегмента топика смещения должны быть относительно небольшими, чтобы обеспечить более быстрое сжатие лога и загрузку кэша. | int | 104857600 (100 мебибайт) | [1,...] | высокая | только для чтения |
| **process.roles** | Роли, которые использует этот процесс: **broker**, **controller** или **broker,controller**, если обе. Эта конфигурация применима только для кластеров в режиме KRaft (Kafka Raft) (вместо ZooKeeper). Оставьте эту конфигурацию неопределённой или пустой для кластеров Zookeeper. | list | "" | [broker, controller] | высокая | только для чтения |
| **queued.max.requests** | Количество запросов в очереди, разрешённых для плоскости передачи данных, до блокировки сетевых потоков. | int | 500 | [1,...] | высокая | только для чтения |
| **replica.fetch.min.bytes** | Минимальное количество байт, ожидаемое для каждого ответа на выборку. Если байт недостаточно, дожидается **replica.fetch.wait.max.ms**(конфигурация брокера). | int | 1 | -- | высокая | только для чтения |
| **replica.fetch.wait.max.ms** | Максимальное время ожидания для каждого запроса выборки, выданного репликами-подписчиками. Это значение всегда должно быть меньше **replica.lag.time.max.ms**, чтобы предотвратить частое сокращение ISR для топиков с низкой пропускной способностью. | int | 500 | -- | высокая | только для чтения |
| **replica.high.watermark.checkpoint.interval.ms** | Частота, с которой high watermark (максимальное доступное смещение для чтения) сохраняется на диск. | long | 5000 (5 секунд) | -- | высокая | только для чтения |
| **replica.lag.time.max.ms** | Если подписчик не отправлял никаких запросов на выборку или не израсходовал последнее смещение лога лидеров, по крайней мере, в течение этого времени, лидер удалит подписчика из **isr**. | long | 30000 (30 секунд) | -- | высокая | только для чтения |
| **replica.socket.receive.buffer.bytes** | Буфер приёма сокета для сетевых запросов. | int | 65536 (64 кибибайт) | -- | высокая | только для чтения |
| **replica.socket.timeout.ms** | Таймаут сокета для сетевых запросов. Его значение должно быть не ниже **replica.fetch.wait.max.ms**. | int | 30000 (30 секунд) | -- | высокая | только для чтения |
| **request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. | int | 30000 (30 секунд) | -- | высокая | только для чтения |
| **sasl.mechanism.controller.protocol** | Механизм SASL, используемый для связи с контроллерами. По умолчанию — **GSSAPI**. | string | GSSAPI | -- | высокая | только для чтения |
| **socket.receive.buffer.bytes** | Буфер SO\_RCVBUF сокетов сервера сокетов. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 102400 (100 кибибайт) | -- | высокая | только для чтения |
| **socket.request.max.bytes** | Максимальное количество байтов в запросе сокета. | int | 104857600 (100 мебибайт) | [1,...] | высокая | только для чтения |
| **socket.send.buffer.bytes** | Буфер SO\_SNDBUF сокетов сервера сокетов. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 102400 (100 кибибайт) | -- | высокая | только для чтения |
| **transaction.max.timeout.ms** | Максимально допустимый таймаут для транзакций. Если запрошенное клиентом время транзакции превышает это значение, брокер вернёт ошибку в **InitProducerIdRequest**. Это предотвращает слишком большой таймаут клиента, который может остановить чтение потребителями партиций, включённых в транзакцию. | int | 900000 (15 минут) | [1,...] | высокая | только для чтения |
| **transaction.state.log.load.buffer.size** | Размер пакета для чтения из сегментов лога транзакций при загрузке идентификаторов поставщиков и транзакций в кэш ("мягкое" ограничение, переопределяемое, если записи слишком велики). | int | 5242880 | [1,...] | высокая | только для чтения |
| **transaction.state.log.min.isr** | Переопределение конфигурации **min.insync.replicas** для топика транзакции. | int | 2 | [1,...] | высокая | только для чтения |
| **transaction.state.log.num.partitions** | Количество партиций для топика транзакции (не должно меняться после развёртывания). | int | 50 | [1,...] | высокая | только для чтения |
| **transaction.state.log.replication.factor** | Фактор репликации для топика транзакции (установите более высокое значение, чтобы обеспечить доступность). Создание внутреннего топика не будет выполнено до тех пор, пока размер кластера не будет соответствовать этому требованию фактора репликации. | short | 3 | [1,...] | высокая | только для чтения |
| **transaction.state.log.segment.bytes** | Байты сегмента топика транзакции должны быть относительно небольшими, чтобы обеспечить более быстрое сжатие логов и загрузку кэша. | int | 104857600 (100 мебибайт) | [1,...] | высокая | только для чтения |
| **transactional.id.expiration.ms** | Время в мс, в течение которого координатор транзакций будет ждать без получения каких-либо обновлений статуса для текущей транзакции до истечения срока действия идентификатора транзакции. Срок действия идентификаторов транзакций не истекает, пока транзакция ещё продолжается. | int | 604800000 (7 дней) | [1,...] | высокая | только для чтения |
| **unclean.leader.election.enable** | Указывает, следует ли разрешить выбор реплик, не входящих в набор ISR, в качестве лидера в крайнем случае, даже если это может привести к потере данных. | boolean | false | -- | высокая | для всего кластера |
| **zookeeper.connect** | Указывает строку подключения ZooKeeper в форме **hostname:port**, где **host**и **port**— это хост и порт сервера ZooKeeper. Чтобы разрешить подключение через другие ноды ZooKeeper, когда эта машина ZooKeeper не работает, вы также можете указать несколько хостов в форме **hostname1:port1,hostname2:port2,hostname3:port3**.  Сервер также может иметь путь chroot ZooKeeper как часть строки подключения ZooKeeper, которая помещает его данные по некоторому пути в глобальном пространстве имён ZooKeeper. Например, чтобы указать путь chroot **/chroot/path**, вы должны указать строку подключения как **hostname1:port1,hostname2:port2,hostname3:port3/chroot/path**. | string | null | -- | высокая | только для чтения |
| **zookeeper.connection.timeout.ms** | Максимальное время, в течение которого клиент ожидает установления соединения с Zookeeper. Если не установлено, используется значение из **zookeeper.session.timeout.ms**. | int | null | -- | высокая | только для чтения |
| **zookeeper.max.in.flight.requests** | Максимальное количество неподтверждённых запросов, которые клиент отправит Zookeeper перед блокировкой. | int | 10 | [1,...] | высокая | только для чтения |
| **zookeeper.metadata.migration.enable** | Включить миграцию ZK на KRaft. | boolean | false | -- | высокая | только для чтения |
| **zookeeper.session.timeout.ms** | Таймаут сеанса Zookeeper. | int | 18000 (18 секунд) | -- | высокая | только для чтения |
| **zookeeper.set.acl** | Настройка клиента на использование безопасных списков управления доступом (ACL). | boolean | false | -- | высокая | только для чтения |
| **broker.heartbeat.interval.ms** | Промежуток времени в миллисекундах между контрольными (heartbeat-) сообщениями брокера. Используется при работе в режиме KRaft. | int | 2000 (2 секунды) | -- | средняя | только для чтения |
| **broker.id.generation.enable** | Включает автоматическое создание идентификатора брокера на сервере. Если этот параметр включён, необходимо просмотреть значение, настроенное для **reserved.broker.max.id**. | boolean | true | -- | средняя | только для чтения |
| **broker.rack** | Стойка брокера. Параметр будет использоваться при назначении репликации с учётом стойки для обеспечения отказоустойчивости. Примеры: **RACK1**, **us-east-1d**. | string | null | -- | средняя | только для чтения |
| **broker.session.timeout.ms** | Продолжительность времени в миллисекундах, в течение которого длится аренда брокера, если контрольные сигналы не поступают. Используется при работе в режиме KRaft. | int | 9000 (9 секунд) | -- | средняя | только для чтения |
| **connections.max.idle.ms** | Таймаут бездействующих соединений: потоки процессора сокета сервера закрывают соединения, которые простаивают дольше этого времени. | long | 600000 (10 минут) | -- | средняя | только для чтения |
| **connections.max.reauth.ms** | Если явно установлено положительное число (по умолчанию — **0**, а не положительное число), время существования сеанса, которое не превысит настроенное значение, будет сообщено клиентам версии 2.2.0 или более поздних версий при их аутентификации. Брокер отключит любое такое соединение, которое не прошло повторную аутентификацию в течение времени существования сеанса и которое затем впоследствии используется для каких-либо целей, кроме повторной аутентификации. Имена конфигураций могут опционально начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.oauthbearer.connections.max.reauth.ms=3600000**. | long | 0 | -- | средняя | только для чтения |
| **controlled.shutdown.enable** | Включить контролируемое завершение работы сервера. | boolean | true | -- | средняя | только для чтения |
| **controlled.shutdown.max.retries** | Управляемое отключение может завершиться неудачей по нескольким причинам. Этот параметр определяет количество повторных попыток при возникновении такого сбоя. | int | 3 | -- | средняя | только для чтения |
| **controlled.shutdown.retry.backoff.ms** | Перед каждой повторной попыткой системе требуется время для восстановления из состояния, вызвавшего предыдущий сбой (переход контроллера при сбое, задержка реплики и т.д.). Эта конфигурация определяет время ожидания перед повторной попыткой. | long | 5000 (5 секунд) | -- | средняя | только для чтения |
| **controller.quorum.append.linger.ms** | Продолжительность в миллисекундах, в течение которой лидер будет ждать накопления записей перед их сбросом на диск. | int | 25 | -- | средняя | только для чтения |
| **controller.quorum.request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. | int | 2000 (2 секунды) | -- | средняя | только для чтения |
| **controller.socket.timeout.ms** | Таймаут сокета для каналов контроллер-брокер. | int | 30000 (30 секунд) | -- | средняя | только для чтения |
| **default.replication.factor** | Факторы репликации по умолчанию для автоматически создаваемых топиков. | int | 1 | -- | средняя | только для чтения |
| **delegation.token.expiry.time.ms** | Время действия токена в миллисекундах, прежде чем его необходимо будет обновить. Значение по умолчанию 1 день. | long | 86400000 (1 день) | [1,...] | средняя | только для чтения |
| **delegation.token.master.key** | **[Устарело]**: алиас для **delegation.token.secret.key**, который следует использовать вместо этой конфигурации. | password | null | -- | средняя | только для чтения |
| **delegation.token.max.lifetime.ms** | Токен имеет максимальный срок действия, по истечении которого его нельзя будет продлить. Значение по умолчанию 7 дней. | long | 604800000 (7 дней) | [1,...] | средняя | только для чтения |
| **delegation.token.secret.key** | Секретный ключ для создания и проверки токенов делегирования. Один и тот же ключ должен быть настроен для всех брокеров. Если ключ не установлен или имеет пустую строку, брокеры отключат поддержку токена делегирования. | password | null | -- | средняя | только для чтения |
| **delete.records.purgatory.purge.interval.requests** | Интервал очистки (в количестве запросов) запроса на удаление записей. | int | 1 | -- | средняя | только для чтения |
| **fetch.max.bytes** | Максимальное количество байт, которые мы вернём для запроса на выборку. Должно быть не менее 1024. | int | 57671680 (55 мебибайт) | [1024,...] | средняя | только для чтения |
| **fetch.purgatory.purge.interval.requests** | Интервал очистки (в количестве запросов) чистилища запросов на выборку. | int | 1000 | -- | средняя | только для чтения |
| **group.initial.rebalance.delay.ms** | Время, в течение которого координатор группы будет ждать, пока новые потребители присоединятся к новой группе, прежде чем выполнить первую ребалансировку. Более длительная задержка означает потенциально меньшее количество ребалансировок, но увеличивает время до начала обработки. | int | 3000 (3 секунды) | -- | средняя | только для чтения |
| **group.max.session.timeout.ms** | Максимально допустимое время ожидания сеанса для зарегистрированных потребителей. Более длительные таймауты дают потребителям больше времени для обработки сообщений между тактами за счёт увеличения времени на обнаружение сбоев. | int | 1800000 (30 минут) | -- | средняя | только для чтения |
| **group.max.size** | Максимальное количество потребителей, которое может разместить одна группа потребителей. | int | 2147483647 | [1,...] | средняя | только для чтения |
| **group.min.session.timeout.ms** | Минимально допустимое время ожидания сеанса для зарегистрированных потребителей. Более короткие таймауты приводят к более быстрому обнаружению сбоев за счёт более частого тактового сигнала потребителя, что может привести к перегрузке ресурсов брокера. | int | 6000 (6 секунд) | -- | средняя | только для чтения |
| **initial.broker.registration.timeout.ms** | При первоначальной регистрации в кворуме контроллера — количество миллисекунд ожидания перед объявлением об ошибке и выходом из процесса брокера. | int | 60000 (1 минута) | -- | средняя | только для чтения |
| **inter.broker.listener.name** | Имя слушателя, используемого для связи между брокерами. Если этот параметр не установлен, имя слушателя определяется параметром **security.inter.broker.protocol**. Одновременная установка этого свойства и свойства **security.inter.broker.protocol** является ошибкой. | string | null | -- | средняя | только для чтения |
| **inter.broker.protocol.version** | Укажите, какая версия межброкерского протокола будет использоваться.  Обычно параметр устанавливается после обновления всех брокеров до новой версии.  Примеры некоторых допустимых значений: 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. Полный список можно найти в **MetadataVersion**. | string | 3.4-IV0 | [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.4-IV0] | средняя | только для чтения |
| **log.cleaner.backoff.ms** | Время сна, когда нет логов, которые нужно очистить. | long | 15000 (15 секунд) | [0,...] | средняя | для всего кластера |
| **log.cleaner.dedupe.buffer.size** | Общий объём памяти, используемый для дедупликации логов во всех потоках очистки. | long | 134217728 | -- | средняя | для всего кластера |
| **log.cleaner.delete.retention.ms** | Время хранения маркеров удаления для топиков, сжатых логом. Этот параметр также даёт ограничение на время, в течение которого потребитель должен завершить чтение, если он начинает со смещения **0**, чтобы гарантировать получение валидного снапшота конечного этапа (в противном случае маркеры удаления могут быть собраны до того, как они завершат сканирование). | long | 86400000 (1 день) | [0,...] | средняя | для всего кластера |
| **log.cleaner.enable** | Включает запуск процесса очистки логов на сервере. Должен быть включен при использовании любых топиков с **cleanup.policy=compact**, включая топик внутренних смещений. Если этот параметр отключен, эти топики не будут сжиматься и постоянно увеличиваться в размерах. | boolean | true | -- | средняя | только для чтения |
| **log.cleaner.io.buffer.load.factor** | Коэффициент загрузки буфера дедупликации для очистки логово. Процент заполнения буфера дедупликации. Более высокое значение позволит очистить больше логов одновременно, но приведёт к большему количеству коллизий хэшей. | double | 0.9 | -- | средняя | для всего кластера |
| **log.cleaner.io.buffer.size** | Общий объём памяти, используемый для буферов ввода-вывода очистки лога во всех потоках очистки. | int | 524288 | [0,...] | средняя | для всего кластера |
| **log.cleaner.io.max.bytes.per.second** | Очиститель логов будет регулироваться так, чтобы сумма операций чтения и записи в среднем была меньше этого значения. | double | 1.7976931348623157E308 | -- | средняя | для всего кластера |
| **log.cleaner.max.compaction.lag.ms** | Максимальное время, в течение которого сообщение не подлежит сжатию в логе. Применимо только для сжатых логов. | long | 9223372036854775807 | [1,...] | средняя | для всего кластера |
| **log.cleaner.min.cleanable.ratio** | Минимальное соотношение “грязного” лога к общему количеству логов, пригодных для очистки. Если также указаны конфигурации **log.cleaner.max.compaction.lag.ms** или **log.cleaner.min.compaction.lag.ms**, то механизм сжатия логов считает лог пригодным для сжатия, как только:   * пороговое значение коэффициента загрязнённости достигнуто, и в логе были “грязные” (несжатые) записи в течение как минимум продолжительности **log.cleaner.min.compaction.lag.ms**, или * если в логе были “грязные” (несжатые) записи не более периода **log.cleaner.max.compaction.lag.ms**. | double | 0.5 | [0,...,1] | средняя | для всего кластера |
| **log.cleaner.min.compaction.lag.ms** | Минимальное время, в течение которого сообщение будет оставаться в логе несжатым. Применимо только для сжимаемых логов. | long | 0 | [0,...] | средняя | для всего кластера |
| **log.cleaner.threads** | Количество фоновых потоков, используемых для очистки лога. | int | 1 | [0,...] | средняя | для всего кластера |
| **log.cleanup.policy** | Политика очистки по умолчанию для сегментов за пределами окна хранения. Список допустимых политик, разделённый запятыми. Допустимые политики: **delete**и **compact**. | list | delete | [compact, delete] | средняя | для всего кластера |
| **log.index.interval.bytes** | Интервал, с которым мы добавляем запись в индекс смещения. | int | 4096 (4 кибибайта) | [0,...] | средняя | для всего кластера |
| **log.index.size.max.bytes** | Максимальный размер в байтах индекса смещения. | int | 10485760 (10 мебибайт) | [4,...] | средняя | для всего кластера |
| **log.message.format.version** | Укажите версию формата сообщения, которую брокер будет использовать для добавления сообщений в логи. Значение должно быть допустимой версией **MetadataVersion**. Некоторые примеры: 0.8.2, 0.9.0.0, 0.10.0. Для получения более подробной информации проверьте **MetadataVersion**. Устанавливая конкретную версию формата сообщения, пользователь подтверждает, что все существующие сообщения на диске меньше или равны указанной версии. Неправильная установка этого значения приведёт к поломке потребителей более старых версий, поскольку они будут получать сообщения в формате, который они не понимают. | string | 3.0-IV1 | [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.4-IV0] | средняя | только для чтения |
| **log.message.timestamp.difference.max.ms** | Максимальная допустимая разница между отметкой времени, когда брокер получает сообщение, и отметкой времени, указанной в сообщении. Если **log.message.timestamp.type=CreateTime**, сообщение будет отклонено, если разница во временных метках превысит этот порог. Эта конфигурация игнорируется, если **log.message.timestamp.type=LogAppendTime**. Максимально допустимая разница временных меток не должна превышать **log.retention.ms**, чтобы избежать ненужной частой смены логов. | long | 9223372036854775807 | [0,...] | средняя | для всего кластера |
| **log.message.timestamp.type** | Определите, является ли временная метка в сообщении временем создания сообщения или временем добавления лога. Значение должно быть либо **CreateTime**, либо **LogAppendTime**. | string | CreateTime | [CreateTime, LogAppendTime] | средняя | для всего кластера |
| **log.preallocate** | Следует ли предварительно выделять файл при создании нового сегмента? Если вы используете RT.StreamingKafka в Windows, вам, вероятно, нужно установить для него значение **true**. | boolean | false | -- | средняя | для всего кластера |
| **log.retention.check.interval.ms** | Частота в миллисекундах, с которой средство очистки логов проверяет, можно ли удалить какой-либо лог. | long | 300000 (5 минут) | [1,...] | средняя | только для чтения |
| **max.connection.creation.rate** | Максимальная скорость создания соединений, которую мы разрешаем в брокере в любое время. Ограничения на уровне слушателя также можно настроить, добавив к имени конфигурации префикс слушателя, например, **listener.name.internal.max.connection.creation.rate**. Ограничение скорости соединения на уровне брокера должно быть настроено на основе мощности брокера, в то время как ограничения слушателя должны быть настроены на основе требований приложения. Новые соединения будут регулироваться, если будет достигнут предел слушателя или брокера, за исключением межброкерского слушателя. Соединения на межброкерском слушателе будут регулироваться только при достижении ограничения скорости на уровне слушателя. | int | 2147483647 | [0,...] | средняя | для всего кластера |
| **max.connections** | Максимальное количество подключений, которое мы разрешаем в брокере в любое время. Это ограничение применяется в дополнение к любым ограничениям для каждого IP-адреса, настроенным с помощью **max.connections.per.ip**. Ограничения на уровне слушателя также можно настроить, добавив к имени конфигурации префикс слушателя, например, **listener.name.internal.max.connections**. Ограничение на уровне брокера должно быть настроено на основе мощности брокера, а ограничения на слушателя должны быть настроены на основе требований приложения. Новые соединения блокируются, если достигнут лимит слушателя или брокера. Соединения на межброкерском слушателе разрешены, даже если достигнут лимита для всех брокеров. В этом случае будет закрыто последнее использованное соединение на другом слушателе. | int | 2147483647 | [0,...] | средняя | для всего кластера |
| **max.connections.per.ip** | Максимальное количество подключений, которые мы разрешаем с каждого IP-адреса. Для этого параметра можно установить значение **0**, если с использованием свойства **max.connections.per.ip.overrides** настроены переопределения. Новые соединения с IP-адреса сбрасываются, если достигнут предела. | int | 2147483647 | [0,...] | средняя | для всего кластера |
| **max.connections.per.ip.overrides** | Разделённый запятыми список IP-адресов или имён хостов переопределяет максимальное количество подключений по умолчанию. Пример значения: **"hostName:100,127.0.0.1:200"**. | string | "" | -- | средняя | для всего кластера |
| **max.incremental.fetch.session.cache.slots** | Максимальное количество сеансов инкрементальной выборки, которое мы будем поддерживать. | int | 1000 | [0,...] | средняя | только для чтения |
| **num.partitions** | Количество партиций лога по умолчанию для каждого топика. | int | 1 | [1,...] | средняя | только для чтения |
| **password.encoder.old.secret** | Старый секрет, который использовался для шифрования динамически настраиваемых паролей. Параметр требуется только при обновлении секрета. Если указано, все динамически закодированные пароли декодируются с использованием этого старого секрета и перекодируются с использованием **password.encoder.secret** при запуске брокера. | password | null | -- | средняя | только для чтения |
| **password.encoder.secret** | Секрет, используемый для шифрования динамически настраиваемых паролей для этого брокера. | password | null | -- | средняя | только для чтения |
| **principal.builder.class** | Полное имя класса, имплементирующего интерфейс **KafkaPrincipalBuilder**, который используется для создания объекта **KafkaPrincipal**, используемого во время авторизации. Если основной конструктор не определён, поведение по умолчанию зависит от используемого протокола безопасности. Для аутентификации SSL принципал будет получен с использованием правил, определенных в **ssl.principal.mapping.rules**, применяемых к отличительному имени из сертификата клиента, если таковой имеется; в противном случае, если аутентификация клиента не требуется, имя участника будет **ANONYMOUS**. Для аутентификации SASL принципал будет получен с использованием правил, определённых в **sasl.kerberos.principal.to.local.rules**, если используется GSSAPI, и идентификатора аутентификации SASL для других механизмов. Для **PLAINTEXT**принципал будет **ANONYMOUS**. | class | org.apache.kafka.common. security.authenticator. DefaultKafkaPrincipalBuilder | -- | средняя | для каждого брокера |
| **producer.purgatory.purge.interval.requests** | Интервал очистки (в количестве запросов) чистилища запроса поставщика. | int | 1000 | -- | средняя | только для чтения |
| **queued.max.request.bytes** | Допустимое количество байт в очереди, прежде чем запросы больше не будут прочитаны. | long | -1 | -- | средняя | только для чтения |
| **replica.fetch.backoff.ms** | Время ожидания при возникновении ошибки выборки партиции. | int | 1000 (1 секунда) | [0,...] | средняя | только для чтения |
| **replica.fetch.max.bytes** | Число байт сообщений, которые необходимо попытаться получить для каждой партиции. Это не абсолютный максимум: если первый пакет записей в первой непустой партиции выборки больше этого значения, пакет записей всё равно будет возвращён, чтобы гарантировать возможность достижения прогресса. Максимальный размер пакета записей, принимаемый брокером, определяется через **message.max.bytes** (конфигурация брокера) или **max.message.bytes** (конфигурация топика). | int | 1048576 (1 мебибайт) | [0,...] | средняя | только для чтения |
| **replica.fetch.response.max.bytes** | Ожидаемое максимальное количество байт для всего ответа на выборку. Записи извлекаются пакетами, и если первый пакет записей в первой непустой партиции выборки больше этого значения, пакет записей всё равно будет возвращён, чтобы гарантировать возможность достижения прогресса. Таким образом, это не абсолютный максимум. Максимальный размер пакета записей, принимаемый брокером, определяется через **message.max.bytes** (конфигурация брокера) или **max.message.bytes** (конфигурация топика). | int | 1048576 (1 мебибайт) | [0,...] | средняя | только для чтения |
| **replica.selector.class** | Полное имя класса, имплементирующего **ReplicaSelector**. Параметр используется брокером для поиска предпочтительной реплики чтения. По умолчанию мы используем имплементацию, которая возвращает лидера. | string | null | -- | средняя | только для чтения |
| **reserved.broker.max.id** | Максимальное число, которое можно использовать для **broker.id**. | int | 1000 | [0,...] | средняя | только для чтения |
| **sasl.client.callback.handler.class** | Полное имя класса механизма обратного вызова клиента SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. | class | null | -- | средняя | только для чтения |
| **sasl.enabled.mechanisms** | Список механизмов SASL, включённых на сервере RT.StreamingKafka. Список может содержать любой механизм, для которого доступен поставщик безопасности. По умолчанию включён только GSSAPI. | list | GSSAPI | -- | средняя | для каждого брокера |
| **sasl.jaas.config** | Параметры контекста входа в систему JAAS для соединений SASL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан [здесь](http://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html). Формат значения: **loginModuleClass controlFlag (optionName=optionValue)\*;**. Для брокеров конфигурация должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;** | password | null | -- | средняя | для каждого брокера |
| **sasl.kerberos.kinit.cmd** | Путь к команде Kerberos **kinit**. | string | /usr/bin/kinit | -- | средняя | для каждого брокера |
| **sasl.kerberos.min.time.before.relogin** | Время ожидания потока входа между попытками обновления. | long | 60000 | -- | средняя | для каждого брокера |
| **sasl.kerberos.principal.to.local.rules** | Список правил маппинга имён принципалов с короткими именами (обычно именами пользователей операционной системы). Правила оцениваются по порядку, и первое правило, соответствующее основному имени, используется для маппинга его с коротким именем. Любые последующие правила в списке игнорируются. По умолчанию имена участников в форме **{username}/{hostname}@{REALM}**сопоставляются с **{username}**. Обратите внимание, что эта конфигурация игнорируется, если расширение **KafkaPrincipalBuilder**предоставляется конфигурацией **principal.builder.class**. | list | DEFAULT | -- | средняя | для каждого брокера |
| **sasl.kerberos.service.name** | Основное имя Kerberos, под которым работает RT.StreamingKafka. Это можно определить либо в конфигурации JAAS RT.StreamingKafka, либо в конфигурации RT.StreamingKafka. | string | null | -- | средняя | для каждого брокера |
| **sasl.kerberos.ticket.renew.jitter** | Процент случайного колебания, добавленного ко времени обновления. | double | 0.05 | -- | средняя | для каждого брокера |
| **sasl.kerberos.ticket.renew.window.factor** | Поток входа будет находиться в спящем режиме до тех пор, пока не будет достигнут указанный временной интервал от последнего обновления до истечения срока действия билета, после чего он попытается продлить билет. | double | 0.8 | -- | средняя | для каждого брокера |
| **sasl.login.callback.handler.class** | Полное имя класса механизма обратного вызова для входа в систему SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. Для брокеров конфигурация механизма обратного вызова при входе должна иметь префикс слушателя и имя механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler**. | class | null | -- | средняя | только для чтения |
| **sasl.login.class** | Полное имя класса, имплементирующего интерфейс входа в систему. Для брокеров конфигурация входа должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin**. | class | null | -- | средняя | только для чтения |
| **sasl.login.refresh.buffer.seconds** | Время буферизации до истечения срока действия учётных данных, которое необходимо поддерживать при обновлении учётных данных, в секундах. Если в противном случае обновление произойдёт ближе к истечению срока действия, чем количество буферных секунд, тогда обновление будет перенесено вверх, чтобы сохранить как можно большую часть буферного времени. Допустимые значения: от **0** до **3600**(1 час); если значение не указано, используется значение по умолчанию **300**(5 минут). Это значение и **sasl.login.refresh.min.period.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к **OAUTHBEARER**. | short | 300 | -- | средняя | для каждого брокера |
| **sasl.login.refresh.min.period.seconds** | Желаемое минимальное время ожидания потока обновления входа перед обновлением учётных данных в секундах. Допустимые значения: от **0** до **900**(15 минут); если значение не указано, используется значение по умолчанию **60**(1 минута). Это значение и **sasl.login.refresh.buffer.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к **OAUTHBEARER**. | short | 60 | -- | средняя | для каждого брокера |
| **sasl.login.refresh.window.factor** | Поток обновления входа в систему будет находиться в режиме ожидания до тех пор, пока не будет достигнут указанный коэффициент окна относительно срока действия учётных данных, после чего он попытается обновить учётные данные. Валидные значения находятся в пределах от **0.5** (50%) до **1.0** (100%) включительно; если значение не указано, используется значение по умолчанию **0.8** (80%). В настоящее время применяется только к **OAUTHBEARER**. | double | 0.8 | -- | средняя | для каждого брокера |
| **sasl.login.refresh.window.jitter** | Максимальное количество случайных колебаний относительно срока действия учётных данных, которое добавляется ко времени сна потока обновления входа в систему. Допустимые значения находятся в диапазоне от **0** до **0.25** (25%) включительно; если значение не указано, используется значение по умолчанию **0.05** (5%). В настоящее время применяется только к **OAUTHBEARER**. | double | 0.05 | -- | средняя | для каждого брокера |
| **sasl.mechanism.inter.broker.protocol** | Механизм SASL, используемый для межброкерской связи. По умолчанию — **GSSAPI**. | string | GSSAPI | -- | средняя | для каждого брокера |
| **sasl.oauthbearer.jwks.endpoint.url** | URL-адрес поставщика OAuth/OIDC, из которого можно получить [JWKS (набор веб-ключей JSON)](https://datatracker.ietf.org/doc/html/rfc7517#section-5) поставщика. URL-адрес может быть основан на HTTP(S) или на основе файла. Если URL-адрес основан на HTTP(S), данные JWKS будут получены от поставщика OAuth/OIDC через настроенный URL-адрес при запуске брокера. Все текущие ключи будут кэшироваться на брокере для входящих запросов. Если получен запрос аутентификации для JWT, который включает значение утверждения заголовка **kid**, которого ещё нет в кэше, эндпоинт JWKS будет запрошен снова по требованию. Однако брокер опрашивает URL-адрес каждые **sasl.oauthbearer.jwks.endpoint.refresh.ms** миллисекунды, чтобы обновить кэш любыми предстоящими ключами до того, как будут получены любые запросы JWT, которые их включают. Если URL-адрес основан на файле, брокер загрузит файл JWKS из настроенного местоположения при запуске. Если JWT включает значение заголовка **kid**, которого нет в файле JWKS, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | средняя | только для чтения |
| **sasl.oauthbearer.token.endpoint.url** | URL-адрес поставщика идентификации OAuth/OIDC. Если URL-адрес основан на HTTP(S), это URL-адрес эндпоинта токена эмитента, к которому будут отправляться запросы на вход в систему на основе конфигурации в **sasl.jaas.config**. Если URL-адрес основан на файле, он указывает файл, содержащий токен доступа (в сериализованной форме JWT), выданный поставщиком идентификации OAuth/OIDC для использования для авторизации. | string | null | -- | средняя | только для чтения |
| **sasl.server.callback.handler.class** | Полное имя класса механизма обратного вызова сервера SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. Механизмы обратного вызова сервера должны иметь префикс слушателя и имя механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.plain.sasl.server.callback.handler.class= com.example.CustomPlainCallbackHandler**. | class | null | -- | средняя | только для чтения |
| **sasl.server.max.receive.size** | Максимальный размер приёма, разрешённый до и во время начальной аутентификации SASL. Размер приёма по умолчанию составляет 512 Кб. GSSAPI ограничивает запросы до 64 Кб, но по умолчанию мы разрешаем до 512 Кб для кастомных механизмов SASL. На практике механизмы PLAIN, SCRAM и OAUTH могут использовать гораздо меньшие ограничения. | int | 524288 | -- | средняя | только для чтения |
| **security.inter.broker.protocol** | Протокол безопасности, используемый для связи между брокерами. Допустимые значения: **PLAINTEXT**, **SSL**, **SASL\_PLAINTEXT**, **SASL\_SSL**. Одновременная установка этого свойства и свойства **inter.broker.listener.name** является ошибкой. | string | PLAINTEXT | [PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL] | средняя | только для чтения |
| **socket.connection.setup.timeout.max.ms** | Максимальное время, в течение которого клиент будет ожидать установления соединения с сокетом. Таймаут установки соединения будет увеличиваться экспоненциально для каждого последующего сбоя соединения вплоть до этого максимума. Чтобы избежать большого количества соединений, к таймауту будет применён коэффициент рандомизации **0.2**, что приведёт к случайному диапазону между 20% ниже и 20% выше вычисленного значения. | long | 30000 (30 секунд) | -- | средняя | только для чтения |
| **socket.connection.setup.timeout.ms** | Время, в течение которого клиент будет ожидать установления соединения с сокетом. Если соединение не будет установлено до истечения времени ожидания, клиенты закроют канал сокета. | long | 10000 (10 секунд) | -- | средняя | только для чтения |
| **socket.listen.backlog.size** | Максимальное количество ожидающих соединений на сокете. В Linux вам также может потребоваться соответствующим образом настроить параметры ядра **`somaxconn`** и **`tcp\_max\_syn\_backlog`**, чтобы конфигурация вступила в силу. | int | 50 | [1,...] | средняя | только для чтения |
| **ssl.cipher.suites** | Список наборов шифров. Это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL. По умолчанию поддерживаются все доступные наборы шифров. | list | "" | -- | средняя | только для чтения |
| **ssl.client.auth** | Настраивает брокера RT.StreamingKafka для запроса аутентификации клиента. Следующие настройки являются общими:   * **ssl.client.auth=required** – если установлено значение **required**, требуется аутентификация клиента. * **ssl.client.auth=requested** – это означает, что аутентификация клиента необязательна. в отличие от обязательного, если эта опция установлена, клиент может отказаться предоставлять аутентификационную информацию о себе. * **ssl.client.auth=none** – это означает, что аутентификация клиента не требуется. | string | none | [required, requested, none] | средняя | для каждого брокера |
| **ssl.enabled.protocols** | Список протоколов, включённых для SSL-соединений. По умолчанию используется значение **TLSv1.2,TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. При значении по умолчанию для Java 11 клиенты и серверы предпочтут TLSv1.3, если оба поддерживают его, и откажутся от TLSv1.2 в противном случае (при условии, что оба поддерживают как минимум TLSv1.2). Это значение по умолчанию должно подойти для большинства случаев. Также см. информацию по конфигурации для **ssl.protocol**. | list | TLSv1.2 | -- | средняя | для каждого брокера |
| **ssl.key.password** | Пароль закрытого ключа в файле хранилища ключей или ключ PEM, указанный в **ssl.keystore.key**. | password | null | -- | средняя | для каждого брокера |
| **ssl.keymanager.algorithm** | Алгоритм, используемый менеджером ключей для SSL-соединений. Значением по умолчанию является алгоритм менеджера ключей, настроенный для виртуальной машины Java. | string | SunX509 | -- | средняя | для каждого брокера |
| **ssl.keystore.certificate.chain** | Цепочка сертификатов в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM со списком сертификатов X.509. | password | null | -- | средняя | для каждого брокера |
| **ssl.keystore.key** | Закрытый ключ в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с ключами PKCS#8. Если ключ зашифрован, пароль ключа необходимо указать с помощью **ssl.key.password**. | password | null | -- | средняя | для каждого брокера |
| **ssl.keystore.location** | Местоположение файла хранилища ключей. Это необязательно для клиента и может использоваться для двусторонней аутентификации клиента. | string | null | -- | средняя | для каждого брокера |
| **ssl.keystore.password** | Пароль хранилища для файла хранилища ключей. Это необязательно для клиента и необходимо только в том случае, если настроен **ssl.keystore.location**. Пароль хранилища ключей не поддерживается для формата PEM. | password | null | -- | средняя | для каждого брокера |
| **ssl.keystore.type** | Формат файла хранилища ключей. Это необязательно для клиента. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя | для каждого брокера |
| **ssl.protocol** | Протокол SSL, используемый для создания **SSLContext**. По умолчанию используется **TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. Это значение должно подойти для большинства случаев использования. Разрешёнными значениями в последних JVM являются **TLSv1.2** и **TLSv1.3**. **TLS**, **TLSv1.1**, **SSL**, **SSLv2**и **SSLv3**могут поддерживаться в старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности. При значении по умолчанию для этой конфигурации и **ssl.enabled.protocols** клиенты перейдут на **TLSv1.2**, если сервер не поддерживает **TLSv1.3**. Если для этой конфигурации установлено значение **TLSv1.2**, клиенты не будут использовать **TLSv1.3**, даже если это одно из значений в **ssl.enabled.protocols**, а сервер поддерживает только **TLSv1.3**. | string | TLSv1.2 | -- | средняя | для каждого брокера |
| **ssl.provider** | Имя поставщика безопасности, используемого для SSL-соединений. Значение по умолчанию — поставщик безопасности по умолчанию для JVM. | string | null | -- | средняя | для каждого брокера |
| **ssl.trustmanager.algorithm** | Алгоритм, используемый доверительным управляющим для SSL-соединений. Значением по умолчанию является алгоритм доверительного управляющего, настроенный для JVM. | string | PKIX | -- | средняя | для каждого брокера |
| **ssl.truststore.certificates** | Доверенные сертификаты в формате, указанном в параметре **ssl.truststore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с сертификатами X.509. | password | null | -- | средняя | для каждого брокера |
| **ssl.truststore.location** | Расположение файла хранилища доверенных сертификатов. | string | null | -- | средняя | для каждого брокера |
| **ssl.truststore.password** | Пароль для файла хранилища доверенных сертификатов. Если пароль не установлен, настроенный файл хранилища доверенных сертификатов по-прежнему будет использоваться, но проверка целостности будет отключена. Пароль хранилища доверенных сертификатов не поддерживается для формата PEM. | password | null | -- | средняя | для каждого брокера |
| **ssl.truststore.type** | Формат файла хранилища доверенных сертификатов. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя | для каждого брокера |
| **zookeeper.clientCnxnSocket** | Обычно устанавливается значение **org.apache.zookeeper.ClientCnxnSocketNetty** при использовании TLS-подключения к ZooKeeper. Переопределяет любое явное значение, заданное через одноименное системное свойство **Zookeeper.clientCnxnSocket**. | string | null | -- | средняя | только для чтения |
| **zookeeper.ssl.client.enable** | Настройка клиента на использование TLS при подключении к ZooKeeper. Явное значение переопределяет любое значение, установленное через системное свойство **zookeeper.client.secure** (обратите внимание на другое имя). По умолчанию — **false**, если ни один из них не установлен; если **true**, должен быть установлен **zookeeper.clientCnxnSocket** (обычно **org.apache.zookeeper.ClientCnxnSocketNetty**); другие значения, которые нужно установить, могут включать **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**. | boolean | false | -- | средняя | только для чтения |
| **zookeeper.ssl.keystore.location** | Расположение хранилища ключей при использовании сертификата на стороне клиента с подключением TLS к ZooKeeper. Переопределяет любое явное значение, установленное через системное свойство **zookeeper.ssl.keyStore.location** (обратите внимание на **CamelCase**). | string | null | -- | средняя | только для чтения |
| **zookeeper.ssl.keystore.password** | Пароль хранилища ключей при использовании сертификата на стороне клиента с подключением TLS к ZooKeeper. Переопределяет любое явное значение, установленное через системное свойство **zookeeper.ssl.keyStore.password** (обратите внимание на **CamelCase**). Обратите внимание, что ZooKeeper не поддерживает пароль ключа, отличный от пароля хранилища ключей, поэтому обязательно установите пароль ключа в хранилище ключей идентичным паролю хранилища ключей; в противном случае попытка подключения к Zookeeper завершится неудачей. | password | null | -- | средняя | только для чтения |
| **zookeeper.ssl.keystore.type** | Тип хранилища ключей при использовании сертификата на стороне клиента с подключением TLS к ZooKeeper. Переопределяет любое явно заданное значение через системное свойство **zookeeper.ssl.keyStore.type** (обратите внимание на **CamelCase**). Значение по умолчанию **null**означает, что тип будет определён автоматически на основе расширения имени файла в хранилище ключей. | string | null | -- | средняя | только для чтения |
| **zookeeper.ssl.truststore.location** | Местоположение хранилища доверенных сертификатов при использовании TLS-подключения к ZooKeeper. Переопределяет любое явное значение, установленное через системное свойство **zookeeper.ssl.trustStore.location** (обратите внимание на **CamelCase**). | string | null | -- | средняя | только для чтения |
| **zookeeper.ssl.truststore.password** | Пароль хранилища доверенных сертификатов при использовании TLS-подключения к ZooKeeper. Переопределяет любое явное значение, установленное через системное свойство **zookeeper.ssl.trustStore.password** (обратите внимание на **CamelCase**). | password | null | -- | средняя | только для чтения |
| **zookeeper.ssl.truststore.type** | Тип хранилища доверенных сертификатов при использовании подключения TLS к ZooKeeper. Переопределяет любое явно заданное значение через системное свойство **zookeeper.ssl.trustStore.type** (обратите внимание на **CamelCase**). Значение по умолчанию **null**означает, что тип будет определён автоматически на основе расширения имени файла в хранилище доверенных сертификатов. | string | null | -- | средняя | только для чтения |
| **alter.config.policy.class.name** | Класс политики изменения конфигурации, который следует использовать для проверки. Класс должен имплементировать интерфейс **org.apache.kafka.server.policy.AlterConfigPolicy**. | class | null | -- | низкая | только для чтения |
| **alter.log.dirs.replication.quota.window.num** | Число выборок, сохраняемых в памяти для изменения квот репликации каталогов логов. | int | 11 | [1,...] | низкая | только для чтения |
| **alter.log.dirs.replication.quota.window.size.seconds** | Временной интервал каждой выборки для изменения квот репликации каталогов логов. | int | 1 | [1,...] | низкая | только для чтения |
| **authorizer.class.name** | Полное имя класса, имплементирующего интерфейс **org.apache.kafka.server.authorizer.Authorizer**, который используется брокером для авторизации. | string | "" | non-null string | низкая | только для чтения |
| **auto.include.jmx.reporter** | [**Устарело**]. Следует ли автоматически включать JmxReporter, даже если он не указан в **metric.reporters**. Эта конфигурация будет удалена в Kafka 4.0, вместо этого пользователям следует включить **org.apache.kafka.common.metrics.JmxReporter** в **metric.reporters**, чтобы включить JmxReporter. | boolean | true | -- | низкая | только для чтения |
| **client.quota.callback.class** | Полное имя класса, имплементирующего интерфейс **ClientQuotaCallback**, который используется для определения ограничений квоты, применяемой к клиентским запросам. По умолчанию применяются квоты **<user>** и **<client-id>**, хранящиеся в ZooKeeper. Для любого запроса применяется наиболее конкретная квота, соответствующая субъекту пользователя сеанса и идентификатору клиента запроса. | class | null | -- | низкая | только для чтения |
| **connection.failed.authentication.delay.ms** | Задержка закрытия соединения при неудачной аутентификации: это время (в миллисекундах), на которое будет задерживаться закрытие соединения при неудачной аутентификации. Это значение должно быть меньше, чем **connections.max.idle.ms**, чтобы предотвратить таймаут соединения. | int | 100 | [0,...] | низкая | только для чтения |
| **controller.quorum.retry.backoff.ms** | Время ожидания перед попыткой повторения неудачного запроса к данной партиции топика. Это позволяет избежать повторной отправки запросов в тесном цикле в некоторых сценариях сбоя. | int | 20 | -- | низкая | только для чтения |
| **controller.quota.window.num** | Количество выборок, сохраняемых в памяти для квот мутаций контроллера. | int | 11 | [1,...] | низкая | только для чтения |
| **controller.quota.window.size.seconds** | Временной интервал каждой выборки для квот мутаций контроллера. | int | 1 | [1,...] | низкая | только для чтения |
| **create.topic.policy.class.name** | Класс политики создания топика, который следует использовать для валидации. Класс должен имплементировать интерфейс**org.apache.kafka.server.policy.CreateTopicPolicy**. | class | null | -- | низкая | только для чтения |
| **delegation.token.expiry.check.interval.ms** | Интервал сканирования для удаления токенов делегирования с истекшим сроком действия. | long | 3600000 (1 час) | [1,...] | низкая | только для чтения |
| **kafka.metrics.polling.interval.secs** | Интервал опроса метрик (в секундах), который можно использовать в имплементациях **kafka.metrics.reporters**. | int | 10 | [1,...] | низкая | только для чтения |
| **kafka.metrics.reporters** | Список классов, которые будут использоваться в качестве кастомных генераторов отчётов по метрикам Yammer. Генераторы отчётов должны имплементировать признак **kafka.metrics.KafkaMetricsReporter**. Если клиент хочет предоставлять операции JMX в кастомном генераторе отчётов, этому кастомному генератору отчётов необходимо дополнительно имплементировать признак MBean, который расширяет признак **kafka.metrics.KafkaMetricsReporterMBean**, чтобы зарегистрированный компонент MBean соответствовал стандартному соглашению MBean. | list | "" | -- | низкая | только для чтения |
| **listener.security.protocol.map** | Маппинг имён слушателей и протоколов безопасности. Это должно быть определено для того, чтобы один и тот же протокол безопасности можно было использовать более чем в одном порту или IP. Например, внутренний и внешний трафик можно разделить, даже если для обоих требуется SSL. Конкретно, пользователь может определить слушателей с именами **INTERNAL**и **EXTERNAL**и этим свойством как: **INTERNAL:SSL,EXTERNAL:SSL**. Как показано, ключ и значение разделяются двоеточием, а записи маппинга — запятыми. Каждое имя слушателя должно появляться в маппинге только один раз. Для каждого слушателя можно настроить различные параметры безопасности (SSL и SASL), добавив к имени конфигурации нормализованный префикс (имя слушателя пишется строчными буквами). Например, чтобы установить другое хранилище ключей для слушателя **INTERNAL**, будет установлена конфигурация с именем **listener.name.internal.ssl.keystore.location**. Если конфигурация для имени слушателя не установлена, конфигурация вернётся к общей конфигурации (т.е. **ssl.keystore.location**). Обратите внимание, что в KRaft предполагается маппинг по умолчанию имён слушателей, определённых в **controller.listener.names**, с **PLAINTEXT**, если не указан явный маппинг и не используется другой протокол безопасности. | string | PLAINTEXT:PLAINTEXT, SSL:SSL, SASL\_PLAINTEXT: SASL\_PLAINTEXT, SASL\_SSL:SASL\_SSL | -- | низкая | для каждого брокера |
| **log.message.downconversion.enable** | Эта конфигурация определяет, включено ли понижающее преобразование форматов сообщений для удовлетворения запросов на потребление. Если установлено значение **false**, брокер не будет выполнять понижающее преобразование для потребителей, ожидающих более старого формата сообщения. Брокер отвечает ошибкой **UNSUPPORTED\_VERSION** на запросы потребления от таких старых клиентов. Эта конфигурация не применяется ни к каким преобразованиям формата сообщений, которые могут потребоваться для репликации подписчикам. | boolean | true | -- | низкая | для всего кластера |
| **metadata.max.idle.interval.ms** | Эта конфигурация определяет, как часто активный контроллер должен записывать неактивные записи в партиции метаданных. Если значение равно **0**, неиспользуемые записи не добавляются в партицию метаданных. Значение по умолчанию — **500**. | int | 500 | [0,...] | низкая | только для чтения |
| **metric.reporters** | Список классов для использования в качестве генераторов отчётов по метрикам. Имплементация интерфейса **org.apache.kafka.common.metrics.MetricsReporter**позволяет подключать классы, которые будут уведомляться о создании новой метрики. JmxReporter всегда включён для регистрации статистики JMX. | list | "" | -- | низкая | для всего кластера |
| **metrics.num.samples** | Количество выборок, поддерживаемых для вычисления метрик. | int | 2 | [1,...] | низкая | только для чтения |
| **metrics.recording.level** | Самый высокий уровень записи метрик. | string | INFO | -- | низкая | только для чтения |
| **metrics.sample.window.ms** | Окно времени, в течение которого вычисляется выборка метрик. | long | 30000 (30 секунд) | [1,...] | низкая | только для чтения |
| **password.encoder.cipher.algorithm** | Алгоритм шифрования, используемый для кодирования динамически настраиваемых паролей. | string | AES/CBC/PKCS5Padding | -- | низкая | только для чтения |
| **password.encoder.iterations** | Счётчик итераций, используемый для кодирования динамически настраиваемых паролей. | int | 4096 | [1024,...] | низкая | только для чтения |
| **password.encoder.key.length** | Длина ключа, используемая для кодирования динамически настраиваемых паролей. | int | 128 | [8,...] | низкая | только для чтения |
| **password.encoder.keyfactory.algorithm** | Алгоритм **SecretKeyFactory**, используемый для кодирования динамически настраиваемых паролей. По умолчанию — **PBKDF2WithHmacSHA512**, если доступно, и **PBKDF2WithHmacSHA1**в противном случае. | string | null | -- | низкая | только для чтения |
| **producer.id.expiration.ms** | Время в мс, в течение которого лидер партиции топика будет ждать до истечения срока действия идентификаторов поставщика. Срок действия идентификаторов поставщиков не истечет, пока транзакция, связанная с ними, ещё продолжается. Обратите внимание, что срок действия идентификаторов поставщиков может истечь раньше, если последняя запись из идентификатора поставщика будет удалена из-за настроек хранения топика. Установка этого значения равным или более высоким, чем значение **delivery.timeout.ms**, может помочь предотвратить истечение срока действия во время повторных попыток и защитить от дублирования сообщений, но значение по умолчанию должно быть разумным для большинства случаев использования. | int | 86400000 (1 день) | [1,...] | низкая | для всего кластера |
| **quota.window.num** | Количество выборок, сохраняемых в памяти для клиентских квот. | int | 11 | [1,...] | низкая | только для чтения |
| **quota.window.size.seconds** | Временной интервал каждой выборки для клиентских квот. | int | 1 | [1,...] | низкая | только для чтения |
| **replication.quota.window.num** | Количество выборок, сохраняемых в памяти для квот репликации. | int | 11 | [1,...] | низкая | только для чтения |
| **replication.quota.window.size.seconds** | Временной интервал каждой выборки для квот репликации. | int | 1 | [1,...] | низкая | только для чтения |
| **sasl.login.connect.timeout.ms** | [Опциональное] значение таймаута подключения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к **OAUTHBEARER**. | int | null | -- | низкая | только для чтения |
| **sasl.login.read.timeout.ms** | [Опциональное] значение таймаута чтения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к **OAUTHBEARER**. | int | null | -- | низкая | только для чтения |
| **sasl.login.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к **OAUTHBEARER**. | long | 10000 (10 секунд) | -- | низкая | только для чтения |
| **sasl.login.retry.backoff.ms** | [Опциональное] значение в миллисекундах для начального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке**sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к **OAUTHBEARER**. | long | 100 | -- | низкая | только для чтения |
| **sasl.oauthbearer.clock.skew.seconds** | [Опциональное] значение в секундах, позволяющее учитывать разницу между временем поставщика идентификации OAuth/OIDC и брокера. | int | 30 | -- | низкая | только для чтения |
| **sasl.oauthbearer.expected.audience** | [Опциональный] параметр, разделённый запятыми, который брокер будет использовать для проверки того, что JWT был выпущен для одной из ожидаемых аудиторий. JWT будет проверен на наличие стандартного утверждения OAuth **aud**, и если это значение установлено, брокер сопоставит значение из утверждения **aud**JWT, чтобы увидеть, есть ли точное совпадение. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачей. | list | null | -- | низкая | только для чтения |
| **sasl.oauthbearer.expected.issuer** | [Опциональный] параметр, который брокер должен использовать для проверки того, что JWT был создан ожидаемым эмитентом. JWT будет проверен на наличие стандартного утверждения OAuth **iss**, и если это значение установлено, брокер будет точно сопоставлять его с тем, что находится в утверждении **iss**JWT. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачей. | string | null | -- | низкая | только для чтения |
| **sasl.oauthbearer.jwks.endpoint.refresh.ms** | [Опциональное] значение в миллисекундах, в течение которого брокер должен ждать между обновлением своего кэша JWKS (набора веб-ключей JSON), содержащего ключи для проверки подписи JWT. | long | 3600000 (1 час) | -- | низкая | только для чтения |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками получить JWKS (набор веб-ключей JSON) от внешнего поставщика идентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting**. | long | 10000 (10 секунд) | -- | низкая | только для чтения |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms** | [Опциональное] значение в миллисекундах для первоначального ожидания между попытками получения JWKS (набор веб-ключей JSON) от внешнего поставщика идентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms setting**. | long | 100 | -- | низкая | только для чтения |
| **sasl.oauthbearer.scope.claim.name** | Утверждение OAuth для области часто называется **scope**, но этот [опциональный] параметр может предоставить другое имя для области, включённой в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | scope | -- | низкая | только для чтения |
| **sasl.oauthbearer.sub.claim.name** | Утверждение OAuth для субъекта часто называется **sub**, но этот [опциональный] параметр может предоставить другое имя для субъекта, включённого в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | sub | -- | низкая | только для чтения |
| **security.providers** | Список настраиваемых классов-создателей, каждый из которых возвращает поставщика, имплементирующего алгоритмы безопасности. Эти классы должны реализовывать интерфейс **org.apache.kafka.common.security.auth.SecurityProviderCreator**. | string | null | -- | низкая | только для чтения |
| **ssl.endpoint.identification.algorithm** | Алгоритм идентификации эндпоинта для валидации имени хоста сервера с использованием сертификата сервера. | string | https | -- | низкая | для каждого брокера |
| **ssl.engine.factory.class** | Класс типа **org.apache.kafka.common.security.auth.SslEngineFactory** для предоставления объектов механизму SSL. Значение по умолчанию — **org.apache.kafka.common.security.ssl.DefaultSslEngineFactory**. | class | null | -- | низкая | для каждого брокера |
| **ssl.principal.mapping.rules** | Список правил маппинга отличительного имени из сертификата клиента с коротким именем. Правила оцениваются по порядку, и первое правило, соответствующее основному имени, используется для маппинга его с коротким именем. Любые последующие правила в списке игнорируются. По умолчанию отличительное имя сертификата X.500 будет принципалом. Обратите внимание, что эта конфигурация игнорируется, если расширение **KafkaPrincipalBuilder**предоставляется конфигурацией **principal.builder.class**. | string | DEFAULT | -- | низкая | только для чтения |
| **ssl.secure.random.implementation** | Имплементация **SecureRandom PRNG**для использования в операциях шифрования SSL. | string | null | -- | низкая | для каждого брокера |
| **transaction.abort.timed.out.transaction.cleanup.** **interval.ms** | Интервал, с которым откатываются транзакции, время ожидания которых истекло. | int | 10000 (10 секунд) | [1,...] | низкая | только для чтения |
| **transaction.remove.expired.transaction.cleanup. interval.ms** | Интервал, с которым удаляются транзакции, срок действия которых истёк из-за того, что время **transactional.id.expiration.ms** прошло. | int | 3600000 (1 час) | [1,...] | низкая | только для чтения |
| **zookeeper.ssl.cipher.suites** | Указывает включённые наборы шифров, которые будут использоваться при верификации TLS ZooKeeper (csv). Переопределяет любое явно заданное значение через системное свойство **zookeeper.ssl.ciphersuites**. Значение по умолчанию **null**означает, что список включённых наборов шифров определяется используемой средой выполнения Java. | list | null | -- | низкая | только для чтения |
| **zookeeper.ssl.crl.enable** | Указывает, включать ли список отзыва сертификатов в протоколах TLS ZooKeeper. Переопределяет любое явно заданное значение через системное свойство **zookeeper.ssl.crl**. | boolean | false | -- | низкая | только для чтения |
| **zookeeper.ssl.enabled.protocols** | Указывает включённые протоколы в верификации TLS ZooKeeper (csv). Переопределяет любое явное значение, установленное через системное свойство **zookeeper.ssl.enabledProtocols**. Значение по умолчанию **null**означает, что включённый протокол будет значением свойства конфигурации **zookeeper.ssl.protocol**. | list | null | -- | низкая | только для чтения |
| **zookeeper.ssl.endpoint.identification.algorithm** | Указывает, включать ли проверку имени хоста в процессе верификации TLS ZooKeeper, при этом (без учёта регистра) **https**означает, что верификация имени хоста ZooKeeper включена, а явное пустое значение означает, что она отключена (отключение рекомендуется только в целях тестирования). Явное значение переопределяет любое значение **true**или **false**, установленное через системное свойство **zookeeper.ssl.hostnameVerification** (обратите внимание на другое имя и значения; **true**подразумевает **https**, а **false**подразумевает пустое значение). | string | HTTPS | -- | низкая | только для чтения |
| **zookeeper.ssl.ocsp.enable** | Указывает, включать ли Online Certificate Status Protocol в протоколах TLS ZooKeeper. Переопределяет любое явно заданное значение через системное свойство **zookeeper.ssl.ocsp**. | boolean | false | -- | низкая | только для чтения |
| **zookeeper.ssl.protocol** | Указывает протокол, который будет использоваться при верификации TLS ZooKeeper. Явное значение переопределяет любое значение, установленное через одноименное системное свойство **zookeeper.ssl.protocol**. | string | TLSv1.2 | -- | низкая | только для чтения |

Более подробную информацию о настройке брокера можно найти в классе scala **kafka.server.KafkaConfig**.

### 2.1.1 Обновление конфигураций брокера

Некоторые конфигурации брокера можно обновлять без перезапуска брокера. См. столбец **Режим обновления** в **п. 2.1**, чтобы узнать о режиме обновления каждой конфигурации брокера.

* только для чтения – для обновления требуется перезапуск брокера.
* для каждого брокера – может обновляться динамически для каждого брокера.
* для всего кластера – может обновляться динамически как значение по умолчанию для всего кластера. Также может быть обновлено как значение для каждого брокера для тестирования.

Чтобы изменить текущие конфигурации для брокера с идентификатором **0** (например, количество потоков очистки логов):

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-name 0 --alter --add-config log.cleaner.threads=2

Copy

Чтобы описать текущие конфигурации динамического брокера с идентификатором **0**:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-name 0 --describe

Copy

Чтобы удалить переопределение конфигурации и вернуться к статически настроенному значению или значению по умолчанию для идентификатора брокера **0** (например, количества потоков очистки логов):

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-name 0 --alter --delete-config log.cleaner.threads

Copy

Некоторые конфигурации можно настроить по умолчанию для всего кластера, чтобы поддерживать согласованные значения во всем кластере. Все брокеры в кластере будут обрабатывать обновление кластера по умолчанию. Например, чтобы обновить потоки очистки логов на всех брокерах:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-default --alter --add-config log.cleaner.threads=2

Copy

Чтобы описать текущие настроенные динамические конфигурации по умолчанию для всего кластера:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-default --describe

Copy

Все конфигурации, которые можно настроить на уровне кластера, также можно настроить на уровне каждого брокера (например, для тестирования). Если значение конфигурации определено на разных уровнях, используется следующий порядок приоритета:

1. Динамическая конфигурация каждого брокера, хранящаяся в ZooKeeper.
2. Динамическая конфигурация по умолчанию для всего кластера, хранящаяся в ZooKeeper.
3. Конфигурация статического брокера из **server.properties**.
4. Конфигурация RT.StreamingKafka по умолчанию, см. **п. 2.1**.

#### 2.1.1.1 Динамическое обновление конфигурации паролей

Значения конфигурации пароля, которые динамически обновляются, шифруются перед сохранением в ZooKeeper. Конфигурация брокера **password.encoder.secret** должна быть настроена в **server.properties**, чтобы включить динамическое обновление конфигураций паролей. У разных брокеров секрет может быть разным.

Секрет, используемый для кодирования пароля, может быть изменён при последовательном перезапуске брокеров. Старый секрет, используемый для кодирования паролей, которые в настоящее время используются в ZooKeeper, должен быть указан в статической конфигурации брокера **password.encoder.old.secret**, а новый секрет должен быть указан в **password.encoder.secret**. Все динамические конфигурации паролей, хранящиеся в ZooKeeper, будут перекодированы с использованием нового секрета при запуске брокера.

#### 2.1.1.2 Обновление конфигурации паролей в ZooKeeper перед запуском брокеров

Начиная с Kafka 2.0.0, **kafka-configs.sh** позволяет обновлять динамические конфигурации брокера с помощью ZooKeeper перед запуском брокеров для начальной загрузки. Это позволяет хранить все конфигурации паролей в зашифрованном виде, избегая необходимости использования открытых паролей в **server.properties**. Конфигурация брокера **password.encoder.secret** также должна быть указана, если в команду изменения включены какие-либо конфигурации паролей. Также могут быть указаны дополнительные параметры шифрования. Конфигурации кодировщика паролей не будут сохраняться в ZooKeeper. Например, чтобы сохранить пароль ключа SSL для слушателя **INTERNAL**на брокере **0**:

> bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file zk\_tls\_config.properties --entity-type brokers --entity-name 0 --alter --add-config

'listener.name.internal.ssl.key.password=key-password,password.encoder.secret=secret,password.encoder.iterations=8192'

Copy

Конфигурация **listener.name.internal.ssl.key.password** будет храниться в ZooKeeper в зашифрованном виде с использованием предоставленных конфигураций кодировщика. Закодированный секрет и итерации не сохраняются в ZooKeeper.

#### 2.1.1.3 Обновление хранилища ключей SSL существующего слушателя

Брокеры могут быть настроены на использование хранилищ ключей SSL с короткими сроками действия, чтобы снизить риск компрометации сертификатов. Хранилища ключей можно обновлять динамически без перезапуска брокера. Имя конфигурации должно начинаться с префикса слушателя **listener.name.{listenerName}.**, чтобы обновлялась только конфигурация хранилища ключей конкретного слушателя. Следующие конфигурации могут быть обновлены в одном запросе на изменение на уровне каждого брокера:

ssl.keystore.type

ssl.keystore.location

ssl.keystore.password

ssl.key.password

Copy

Если слушатель является слушателем между брокерами, обновление разрешено только в том случае, если новому хранилищу ключей (keystore) доверяет хранилище доверенных сертификатов (truststore) , настроенное для этого слушателя. Для других слушателей брокер не выполняет проверку доверия в хранилище ключей. Сертификаты должны быть подписаны тем же центром сертификации, который подписал старый сертификат, чтобы избежать ошибок аутентификации клиента.

#### 2.1.1.4 Обновление хранилища доверенных сертификатов SSL существующего слушателя

Хранилища доверенных сертификатов брокера можно обновлять динамически без перезапуска брокера для добавления или удаления сертификатов. Обновлённое хранилище доверенных сертификатов будет использоваться для аутентификации новых клиентских подключений. Имя конфигурации должно начинаться с префикса слушателя **listener.name.{listenerName}.**, так что обновляется только конфигурация хранилища доверенных сертификатов конкретного слушателя. Следующие конфигурации могут быть обновлены в одном запросе на изменение на уровне каждого брокера:

ssl.truststore.type

ssl.truststore.location

ssl.truststore.password

Copy

Если слушатель является слушателем между брокерами, обновление разрешено только в том случае, если существующему хранилищу ключей для этого слушателя доверяет новое хранилище доверенных сертификатов. Для других слушателей брокер не выполняет проверку доверия перед обновлением. Удаление сертификатов центров сертификации, используемых для подписи сертификатов клиентов, из нового хранилища доверенных сертификатов может привести к сбоям аутентификации клиента.

#### 2.1.1.5 Обновление конфигурации топика по умолчанию

Параметры конфигурации топика по умолчанию, используемые брокерами, могут быть обновлены без перезапуска брокера. Конфигурации применяются к топикам без переопределения конфигурации топика для эквивалентной конфигурации для каждого топика. Одна или несколько из этих конфигураций могут быть переопределены на уровне кластера по умолчанию, используемом всеми брокерами.

log.segment.bytes

log.roll.ms

log.roll.hours

log.roll.jitter.ms

log.roll.jitter.hours

log.index.size.max.bytes

log.flush.interval.messages

log.flush.interval.ms

log.retention.bytes

log.retention.ms

log.retention.minutes

log.retention.hours

log.index.interval.bytes

log.cleaner.delete.retention.ms

log.cleaner.min.compaction.lag.ms

log.cleaner.max.compaction.lag.ms

log.cleaner.min.cleanable.ratio

log.cleanup.policy

log.segment.delete.delay.ms

unclean.leader.election.enable

min.insync.replicas

max.message.bytes

compression.type

log.preallocate

log.message.timestamp.type

log.message.timestamp.difference.max.ms

Copy

Начиная с версии Kafka 2.0.0, нечистые выборы лидера автоматически включаются контроллером при динамическом обновлении конфигурации **unclean.leader.election.enable**. В Kafka версии 1.1.x изменения в **unclean.leader.election.enable** вступают в силу только при выборе нового контроллера. Переизбрание контроллера можно вызвать принудительно, выполнив:

> bin/zookeeper-shell.sh localhost

rmr /controller

Copy

#### 2.1.1.6 Обновление конфигураций средства очистки логов

Конфигурации очистки логов могут обновляться динамически на уровне кластера по умолчанию, используемом всеми брокерами. Изменения вступят в силу на следующей итерации очистки лога. Можно обновить одну или несколько из этих конфигураций:

log.cleaner.threads

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

log.cleaner.dedupe.buffer.size

log.cleaner.io.buffer.size

log.cleaner.io.buffer.load.factor

log.cleaner.backoff.ms

Copy

#### 2.1.1.7 Обновление конфигураций потоков

Размер различных пулов потоков, используемых брокером, может обновляться динамически на уровне кластера по умолчанию, используемом всеми брокерами. Обновления ограничены диапазоном от **currentSize / 2** до **currentSize \* 2**, чтобы гарантировать правильную обработку обновлений конфигурации.

num.network.threads

num.io.threads

num.replica.fetchers

num.recovery.threads.per.data.dir

log.cleaner.threads

background.threads

Copy

#### 2.1.1.8 Обновление конфигураций ConnectionQuota

Максимальное количество подключений, разрешенное брокером для данного IP-адреса/хоста, может обновляться динамически на уровне кластера по умолчанию, используемом всеми брокерами. Изменения будут применяться к созданию новых подключений, а количество существующих подключений будет учитываться новыми ограничениями.

max.connections.per.ip

max.connections.per.ip.overrides

Copy

#### 2.1.1.9 Добавление и удаление слушателей

Слушатели могут добавляться или удаляться динамически. При добавлении нового слушателя конфигурации безопасности слушателя должны быть предоставлены в виде конфигураций слушателя с префиксом **listener.name.{listenerName}.**. Если новый слушатель использует SASL, конфигурация JAAS слушателя должна быть предоставлена с использованием свойства конфигурации JAAS **sasl.jaas.config** с префиксом слушателя и механизма.

В Kafka версии 1.1.x слушатель, используемый межброкерским слушателем, не может обновляться динамически. Чтобы обновить межброкерского слушателя до нового слушателя, нового слушателя можно добавить на всех брокерах без перезапуска брокера. Затем потребуется последовательный перезапуск для обновления **inter.broker.listener.name**.

В дополнение ко всем конфигурациям безопасности новых слушателей следующие конфигурации могут обновляться динамически на уровне каждого брокера:

listeners

advertised.listeners

listener.security.protocol.map

Copy

Межброкерский слушатель должен быть настроен с использованием статической конфигурации брокера **inter.broker.listener.name** или **security.inter.broker.protocol**.

## 2.2 Конфигурации на уровне топика

Конфигурации, относящиеся к топикам, имеют как серверные настройки по умолчанию, так и необязательное переопределение для каждого топика. Если конфигурация для каждого топика не указана, используется сервер по умолчанию. Переопределение можно установить во время создания топика, указав одну или несколько опций **--config**. В этом примере создаётся топик с именем **my-topic** с настраиваемым максимальным размером сообщения и скоростью очистки:

> bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic my-topic --partitions 1 \

--replication-factor 1 --config max.message.bytes=64000 --config flush.messages=1

Copy

Переопределения также можно изменить или установить позже с помощью команды **--alter**. В этом примере обновляется максимальный размер сообщения для **my-topic**:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type topics --entity-name my-topic

--alter --add-config max.message.bytes=128000

Copy

Чтобы проверить переопределения, установленные для топика:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type topics --entity-name my-topic --describe

Copy

Чтобы удалить переопределение:

> bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type topics --entity-name my-topic

--alter --delete-config max.message.bytes

Copy

Ниже приведены конфигурации на уровне топика. Конфигурация сервера по умолчанию для этого свойства указана под заголовком **Свойство сервера по умолчанию**. Данное значение конфигурации сервера по умолчанию применяется к топику только в том случае, если оно не имеет явного переопределения конфигурации топика.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Свойство сервера по умолчанию** | **Важность** |
| --- | --- | --- | --- | --- | --- | --- |
| **cleanup.policy** | Эта конфигурация определяет политику хранения, которая будет использоваться в сегментах лога. Политика **delete**(которая используется по умолчанию) удаляет старые сегменты, когда достигается предел их времени хранения или размера. Политика **compact**включает сжатие лога, при котором сохраняются последние значения для каждого ключа. Также можно указать обе политики в списке, разделённом запятыми (например, **delete,compact**). В этом случае старые сегменты будут удалены в соответствии с конфигурацией времени хранения и размера, а оставшиеся сегменты будут сжаты. | list | delete | [compact, delete] | log.cleanup.policy | средняя |
| **compression.type** | Указывает окончательный тип сжатия для данного топика. Эта конфигурация поддерживает стандартные кодеки сжатия (**gzip**, **snappy**, **lz4**, **zstd**). Кроме того, она принимает **uncompressed**, что эквивалентно отсутствию сжатия; и **producer**, что означает сохранение исходного кодека сжатия, установленного поставщиком. | string | producer | [uncompressed, zstd, lz4, snappy, gzip, producer] | compression.type | средняя |
| **delete.retention.ms** | Время хранения маркеров удаления для топиков, сжатых логом. Этот параметр также даёт ограничение на время, в течение которого потребитель должен завершить чтение, если он начинает со смещения 0, чтобы гарантировать получение валидного снапшота конечного этапа (в противном случае маркеры удаления могут быть собраны до того, как они завершат сканирование). | long | 86400000 (1 день) | [0,...] | log.cleaner.delete.retention.ms | средняя |
| **file.delete.delay.ms** | Время ожидания перед удалением файла из файловой системы. | long | 60000 (1 минута) | [0,...] | log.segment.delete.delay.ms | средняя |
| **flush.messages** | Этот параметр позволяет указать интервал, через который мы будем принудительно выполнять **fsync**(синхронизацию)данных, записываемых в лог. Например, если бы для этого параметра было установлено значение **1**, мы бы выполняли **fsync**после каждого сообщения; если бы это было **5**, мы бы выполняли синхронизацию после каждых пяти сообщений. В общем, мы рекомендуем вам не устанавливать это значение, а использовать репликацию для обеспечения надёжности и разрешить фоновую очистку операционной системы, поскольку это более эффективно. | long | 9223372036854775807 | [1,...] | log.flush.interval.messages | средняя |
| **flush.ms** | Этот параметр позволяет указать временной интервал, через который мы будем принудительно выполнять **fsync**данных, записываемых в лог. Например, если бы для этого параметра было установлено значение **1000**, мы бы выполнили **fsync**после прохождения 1000 мс. В общем, мы рекомендуем вам не устанавливать это значение, а использовать репликацию для обеспечения надёжности и разрешить фоновую очистку операционной системы, поскольку это более эффективно. | long | 9223372036854775807 | [0,...] | log.flush.interval.ms | средняя |
| **follower.replication.throttled.replicas** | Список реплик, для которых репликация логов должна регулироваться на стороне подписчика. Список должен описывать набор реплик в форме **[PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:...** или, альтернативно, можно использовать подстановочный знак **\*** для регулирования всех реплик для этого топика. | list | "" | [partitionId]:[brokerId],[partitionId]:[brokerId],... | follower.replication.throttled. replicas | средняя |
| **index.interval.bytes** | Этот параметр определяет, как часто RT.StreamingKafka добавляет запись индекса к своему индексу смещения. Настройка по умолчанию гарантирует, что мы индексируем сообщение примерно каждые 4096 байт. Увеличение индексации позволяет чтениям перейти ближе к точной позиции в логе, но увеличивает индекс. Вероятно, вам не нужно это менять. | int | 4096 (4 кибибайт) | [0,...] | log.index.interval.bytes | средняя |
| **leader.replication.throttled.replicas** | Список реплик, для которых репликация логов должна регулироваться на стороне лидера. Список должен описывать набор реплик в форме **[PartitionId]:[BrokerId],[PartitionId]:[BrokerId]:...** или, альтернативно, можно использовать подстановочный знак **\*** для регулирования всех реплик для этого топика. | list | "" | [partitionId]:[brokerId],[partitionId]:[brokerId],... | leader.replication.throttled. replicas | средняя |
| **max.compaction.lag.ms** | Максимальное время, в течение которого сообщение не подлежит сжатию в логе. Применимо только для сжимаемых логов. | long | 9223372036854775807 | [1,...] | log.cleaner.max.compaction. lag.ms | средняя |
| **max.message.bytes** | Максимальный размер пакета записей, разрешённый RT.StreamingKafka (после сжатия, если сжатие включено). Если это значение увеличено и есть потребители выше 0.10.2, размер выборки потребителей также должен быть увеличен, чтобы они могли получать такие большие пакеты записей. В последней версии формата сообщений записи всегда группируются в пакеты для повышения эффективности. В предыдущих версиях формата сообщений несжатые записи не группировались в пакеты, и в этом случае это ограничение применяется только к одной записи. | int | 1048588 | [0,...] | message.max.bytes | средняя |
| **message.format.version** | **[Устарело]** Укажите версию формата сообщений, которую брокер будет использовать для добавления сообщений в логи. Значение этой конфигурации всегда предполагается равным **3.0**, если значение **inter.broker.protocol.version** равно **3.0** или выше (фактическое значение конфигурации игнорируется). В противном случае значение должно быть допустимым **ApiVersion**. Некоторые примеры: **0.10.0**, **1.1**, **2.8**, **3.0**. Устанавливая конкретную версию формата сообщения, пользователь подтверждает, что все существующие сообщения на диске меньше или равны указанной версии. Неправильная установка этого значения приведёт к поломке потребителей более старых версий, поскольку они будут получать сообщения в формате, который они не поддерживают. | string | 3.0-IV1 | [0.8.0, 0.8.1, 0.8.2, 0.9.0, 0.10.0-IV0, 0.10.0-IV1, 0.10.1-IV0, 0.10.1-IV1, 0.10.1-IV2, 0.10.2-IV0, 0.11.0-IV0, 0.11.0-IV1, 0.11.0-IV2, 1.0-IV0, 1.1-IV0, 2.0-IV0, 2.0-IV1, 2.1-IV0, 2.1-IV1, 2.1-IV2, 2.2-IV0, 2.2-IV1, 2.3-IV0, 2.3-IV1, 2.4-IV0, 2.4-IV1, 2.5-IV0, 2.6-IV0, 2.7-IV0, 2.7-IV1, 2.7-IV2, 2.8-IV0, 2.8-IV1, 3.0-IV0, 3.0-IV1, 3.1-IV0, 3.2-IV0, 3.3-IV0, 3.3-IV1, 3.3-IV2, 3.3-IV3, 3.4-IV0] | log.message.format.version | средняя |
| **message.timestamp.difference.max.ms** | Максимальная допустимая разница между отметкой времени, когда брокер получает сообщение, и отметкой времени, указанной в сообщении. Если **message.timestamp.type=CreateTime**, сообщение будет отклонено, если разница во времени превысит этот порог. Эта конфигурация игнорируется, если **message.timestamp.type=LogAppendTime**. | long | 9223372036854775807 | [0,...] | log.message.timestamp. difference.max.ms | средняя |
| **message.timestamp.type** | Определяет является ли временная метка в сообщении временем создания сообщения или временем добавления лога. Значение должно быть либо **CreateTime**, либо **LogAppendTime**. | string | CreateTime | [CreateTime, LogAppendTime] | log.message.timestamp.type | средняя |
| **min.cleanable.dirty.ratio** | Эта конфигурация определяет, как часто механизм сжатия логов будет пытаться очистить лог (при условии, что сжатие логов включено). По умолчанию мы не будем очищать лог, если более 50% лога сжато. Это соотношение ограничивает максимальное пространство, занимаемое в логе дубликатами (при 50% максимум 50% лога могут быть дубликатами). Более высокий коэффициент будет означать меньшее количество и более эффективную очистку, но будет означать больше ненужного места в логе. Если также указаны конфигурации **max.compaction.lag.ms** или **min.compaction.lag.ms**, то механизм сжатия логов считает, что лог пригоден для сжатия, как только:   * достигается пороговое значение коэффициента “загрязнения” и в логе были “грязные” (несжатые) записи в течение как минимум периода **min.compaction.lag.ms**, или * если в логе были “грязные” (несжатые) записи не более чем в течение периода **max.compaction.lag.ms**. | double | 0.5 | [0,...,1] | log.cleaner.min.cleanable.ratio | средняя |
| **min.compaction.lag.ms** | Минимальное время, в течение которого сообщение будет оставаться в логе несжатым. Применимо только для сжимаемых логов. | long | 0 | [0,...] | log.cleaner.min.compaction. lag.ms | средняя |
| **min.insync.replicas** | Когда поставщик устанавливает для **acks**значение **all**(или **-1**), эта конфигурация определяет минимальное количество реплик, которые должны подтвердить запись, чтобы запись считалась успешной. Если этот минимум не может быть достигнут, поставщик выдаст исключение (либо **NotEnoughReplicas**, либо **NotEnoughReplicasAfterAppend**).  При совместном использовании **min.insync.replicas** и **acks**вы можете обеспечить более высокие гарантии надёжности. Типичным сценарием будет создание топика с коэффициентом репликации **3**, установкой **min.insync.replicas** равным **2** и созданием **acks**со значением **all**. Это гарантирует, что поставщик вызовет исключение, если большинство реплик не получат запись. | int | 1 | [1,...] | min.insync.replicas | средняя |
| **preallocate** | **True**, если мы должны предварительно выделить файл на диске при создании нового сегмента лога. | boolean | false | -- | log.preallocate | средняя |
| **retention.bytes** | Эта конфигурация контролирует максимальный размер партиции (который состоит из сегментов лога), до которого мы можем удалить старые сегменты лога, чтобы освободить место, если мы используем политику хранения **delete**. По умолчанию ограничений по размеру нет, только ограничение по времени. Поскольку это ограничение применяется на уровне партиции, умножьте его на количество партиций, чтобы вычислить срок хранения топика в байтах. | long | -1 | -- | log.retention.bytes | средняя |
| **retention.ms** | Эта конфигурация контролирует максимальное время хранения лога, прежде чем мы будем удалять старые сегменты лога, чтобы освободить место, если мы используем политику хранения **delete**. Это представляет собой соглашение об уровне обслуживания о том, как скоро потребители должны прочитать свои данные. Если установлено значение **-1**, ограничение по времени не применяется. | long | 604800000 (7 дней) | [-1,...] | log.retention.ms | средняя |
| **segment.bytes** | Эта конфигурация управляет размером файла сегмента лога. Хранение и очистка всегда выполняются по одному файлу, поэтому больший размер сегмента означает меньшее количество файлов, но менее детальный контроль над хранением. | int | 1073741824 (1 гибибайт) | [14,...] | log.segment.bytes | средняя |
| **segment.index.bytes** | Эта конфигурация управляет размером индекса, который сопоставляет смещения с позициями файла. Мы заранее выделяем этот индексный файл и сжимаем его только после обновления лога. Обычно вам не нужно менять этот параметр. | int | 10485760 (10 мебибайт) | [4,...] | log.index.size.max.bytes | средняя |
| **segment.jitter.ms** | Максимальный случайный джиттер вычитается из запланированного времени сжатия сегмента, чтобы избежать накопления большого количества процессов, ждущих события. | long | 0 | [0,...] | log.roll.jitter.ms | средняя |
| **segment.ms** | Эта конфигурация управляет периодом времени, по истечении которого RT.StreamingKafka принудительно сжимает лог, даже если файл сегмента не заполнен, чтобы гарантировать, что при хранении можно удалить или сжать старые данные. | long | 604800000 (7 дней) | [1,...] | log.roll.ms | средняя |
| **unclean.leader.election.enable** | Указывает, следует ли разрешить выбор реплик, не входящих в набор ISR, в качестве лидера в крайнем случае, даже если это может привести к потере данных. | boolean | false | -- | unclean.leader.election.enable | средняя |
| **message.downconversion.enable** | Эта конфигурация определяет, включено ли понижающее преобразование форматов сообщений для удовлетворения запросов на потребление. Если установлено значение **false**, брокер не будет выполнять понижающее преобразование для потребителей, ожидающих более старого формата сообщения. Брокер отвечает ошибкой **UNSUPPORTED\_VERSION** на запросы потребления от таких старых клиентов. Эта конфигурация не применяется ни к каким преобразованиям формата сообщений, которые могут потребоваться для репликации подписчикам. | boolean | true | -- | log.message.downconversion. enable | низкая |

## 2.3 Конфигурации поставщика

Ниже представлена конфигурация поставщика.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **key.serializer** | Класс сериализатора для ключа, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serializer**. | class | -- | -- | высокая |
| **value.serializer** | Класс сериализатора для значения, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serializer**. | class | -- | -- | высокая |
| **bootstrap.servers** | Список пар хост/порт, которые будут использоваться для установления инициирующего подключения к кластеру RT.StreamingKafka. Клиент будет использовать все серверы независимо от того, какие серверы указаны здесь для начальной загрузки — этот список влияет только на исходные хосты, используемые для обнаружения полного набора серверов. Этот список должен иметь форму **host1:port1,host2:port2,…**. Поскольку эти серверы используются только для инициирующего подключения для обнаружения полного членства в кластере (которое может меняться динамически), этот список не обязательно должен содержать полный набор серверов (хотя вам может понадобиться больше одного, если сервер не работает). | list | "" | non-null string | высокая |
| **buffer.memory** | Общее количество байт памяти, которые поставщик может использовать для буферизации записей, ожидающих отправки на сервер. Если записи отправляются быстрее, чем они могут быть доставлены на сервер, поставщик заблокирует их на **max.block.ms**, после чего выдаст исключение.  Этот параметр должен примерно соответствовать общему объёму памяти, которую будет использовать поставщик, но не является жёсткой привязкой, поскольку не вся память, используемая поставщиком, используется для буферизации. Некоторая дополнительная память будет использоваться для сжатия (если сжатие включено), а также для обслуживания запросов на лету. | long | 33554432 | [0,...] | высокая |
| **compression.type** | Тип сжатия для всех данных, созданных поставщиком. По умолчанию установлено значение **none**(т.е. нет сжатия). Допустимые значения: **none**, **gzip**, **snappy**, **lz4**или **zstd**. Сжатие осуществляется для полных пакетов данных, поэтому эффективность пакетной обработки также будет влиять на степень сжатия (чем выше пакетирование, тем лучше сжатие). | string | none | [none, gzip, snappy, lz4, zstd] | высокая |
| **retries** | Установка значения больше нуля приведёт к тому, что клиент повторно отправит любую запись, отправка которой не удалась, с потенциально временной ошибкой. Обратите внимание, что эта повторная попытка ничем не отличается от ситуации, когда клиент повторно отправляет запись после получения ошибки. Запросы на создание будут неудачными до того, как будет исчерпано количество повторных попыток, если таймаут, настроенный в **delivery.timeout.ms**, истечёт раньше, чем будет получено успешное подтверждение. Пользователям обычно следует оставить эту конфигурацию не настроенной и вместо этого использовать **delivery.timeout.ms** для управления поведением повторных попыток.  Для включения идемпотентности требуется, чтобы это значение конфигурации было больше 0. Если установлены конфликтующие конфигурации и идемпотентность не включена явно, идемпотентность отключается.  Разрешение повторных попыток при установке для параметра **enable.idempotence** значения **false**и **max.in.flight.requests.per.connection** значения **1** потенциально изменит порядок записей, поскольку если два пакета отправляются в одну партицию, а первый завершается неудачей и повторяется, но второй завершается успешно, тогда записи из второго пакета могут появиться первыми. | int | 2147483647 | [0,...,2147483647] | высокая |
| **ssl.key.password** | Пароль закрытого ключа в файле хранилища ключей или ключ PEM, указанный в **ssl.keystore.key**. | password | null | -- | высокая |
| **ssl.keystore.certificate.chain** | Цепочка сертификатов в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM со списком сертификатов X.509. | password | null | -- | высокая |
| **ssl.keystore.key** | Закрытый ключ в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с ключами PKCS#8. Если ключ зашифрован, пароль ключа необходимо указать с помощью **ssl.key.password**. | password | null | -- | высокая |
| **ssl.keystore.location** | Расположение файла хранилища ключей. Это необязательно для клиента и может использоваться для двусторонней аутентификации клиента. | string | null | -- | высокая |
| **ssl.keystore.password** | Пароль хранилища для файла хранилища ключей. Это необязательно для клиента и необходимо только в том случае, если настроен **ssl.keystore.location**. Пароль хранилища ключей не поддерживается для формата PEM. | password | null | -- | высокая |
| **ssl.truststore.certificates** | Доверенные сертификаты в формате, указанном в параметре **ssl.truststore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с сертификатами X.509. | password | null | -- | высокая |
| **ssl.truststore.location** | Расположение файла хранилища доверенных сертификатов. | string | null | -- | высокая |
| **ssl.truststore.password** | Пароль для файла хранилища доверенных сертификатов. Если пароль не установлен, настроенный файл хранилища доверенных сертификатов по-прежнему будет использоваться, но проверка целостности будет отключена. Пароль хранилища доверенных сертификатов не поддерживается для формата PEM. | password | null | -- | высокая |
| **batch.size** | Поставщик попытается объединить записи в меньшее количество запросов всякий раз, когда несколько записей отправляются в одну и ту же партицию. Это повышает производительность как на клиенте, так и на сервере. Эта конфигурация управляет размером пакета по умолчанию в байтах.  Не будет предпринято никаких попыток пакетной обработки записей больше этого установленного размера.  Запросы, отправленные брокерам, будут содержать несколько пакетов, по одному для каждой партиции с данными, доступными для отправки.  Небольшой размер пакета сделает пакетную обработку менее распространённой и может снизить пропускную способность (нулевой размер пакета полностью отключит пакетную обработку). Очень большой размер пакета может использовать память немного более расточительно, поскольку мы всегда будем выделять буфер указанного размера пакета в ожидании дополнительных записей.  **Примечание**. Этот параметр определяет верхнюю границу размера отправляемого пакета. Если для этой партиции накоплено меньше указанного количества байт, мы «задержимся» на время **linger.ms**, ожидая появления новых записей. По умолчанию для этого параметра **linger.ms** установлено значение **0**, что означает, что мы немедленно отправим запись, даже если накопленный размер пакета меньше установленного значения в **batch.size**. | int | 16384 | [0,...] | средняя |
| **client.dns.lookup** | Управляет тем, как клиент использует поиск DNS. Если установлено значение **use\_all\_dns\_ips**, подключается к каждому возвращаемому IP-адресу последовательно, пока не будет установлено успешное соединение. После отключения используется следующий IP. После того, как все IP-адреса были использованы один раз, клиент снова разрешает IP-адреса из имени хоста (однако поиск DNS-имён как в JVM, так и в кэше ОС). Если установлено значение **solve\_canonical\_bootstrap\_servers\_only**, преобразует каждый адрес начальной загрузки в список канонических имён. После фазы начальной загрузки ведёт себя так же, как **use\_all\_dns\_ips**. | string | use\_all\_dns\_ips | [use\_all\_dns\_ips, resolve\_canonical\_bootstrap\_ servers\_only] | средняя |
| **client.id** | Строка идентификатора, передаваемая на сервер при выполнении запросов. Целью является возможность отслеживать источник запросов, помимо IP/порта, позволяя включать логическое имя приложения в лог запросов на стороне сервера. | string | "" | -- | средняя |
| **connections.max.idle.ms** | Закрывает неактивные соединения по истечении количества миллисекунд, указанного в этой конфигурации. | long | 540000 (9 минут) | -- | средняя |
| **delivery.timeout.ms** | Верхняя граница времени сообщения об успехе или неудаче после возврата вызова **send()**. Параметр ограничивает общее время задержки записи перед отправкой, время ожидания подтверждения от брокера (если оно ожидается), а также время, отведённое для повторной отправки сбоев. Поставщик может сообщить о невозможности отправить запись раньше значения этой конфигурации, если произошла неисправимая ошибка, количество повторных попыток исчерпано или запись добавлена в пакет, срок поставки которого истёк раньше. Значение этой конфигурации должно быть больше или равно сумме **request.timeout.ms** и **linger.ms**. | int | 120000 (2 минуты) | [0,...] | средняя |
| **linger.ms** | Поставщик группирует все записи, поступающие между передачами запросов, в один пакетный запрос. Обычно это происходит только при нагрузке, когда записи поступают быстрее, чем могут быть отправлены. Однако в некоторых случаях клиент может захотеть уменьшить количество запросов даже при умеренной нагрузке. Этот параметр достигает этого за счёт добавления небольшой искусственной задержки — то есть вместо немедленной отправки записи поставщик будет ждать заданную задержку, чтобы разрешить отправку других записей, чтобы отправки можно было объединить в пакет. Этот параметр можно рассматривать как аналог алгоритма Нейгла в TCP. Этот параметр определяет верхнюю границу задержки пакетной обработки: как только мы получим количество записей для партиции в рамках значения **batch.size**, они будут отправлены немедленно, независимо от этого параметра, однако, если для этой партиции накоплено меньше этого количества байт, мы “задержимся” на указанное время, ожидая появления новых записей. По умолчанию этот параметр равен **0** (т.е. нет задержки). Например, установка **linger.ms=5** приведёт к уменьшению количества отправляемых запросов, но увеличит задержку до 5 мс для записей, отправляемых в отсутствие нагрузки. | long | 0 | [0,...] | средняя |
| **max.block.ms** | Конфигурация определяет, как долго будут блокироваться методы **send()**, **partsFor()**, **initTransactions()**, **sendOffsetsToTransaction()**, **commitTransaction()** и **abortTransaction()** KafkaProducer. Для **send()** этот таймаут ограничивает общее время ожидания как выборки метаданных, так и выделения буфера (блокировка в пользовательских сериализаторах или партиционерах не учитывается в этот таймаут). Для **partitionsFor()** этот таймаут ограничивает время ожидания метаданных, если они недоступны. Методы, связанные с транзакциями, всегда блокируются, но могут истечь по таймауту, если координатор транзакций не был обнаружен или не ответил в течение таймаута. | long | 60000 (1 минута) | [0,...] | средняя |
| **max.request.size** | Максимальный размер запроса в байтах. Этот параметр ограничивает количество пакетов записей, которые поставщик отправляет в одном запросе, чтобы избежать отправки огромных запросов. Это также фактически ограничение на максимальный размер пакета несжатых записей. Обратите внимание, что сервер имеет собственное ограничение на размер пакета записей (после сжатия, если сжатие включено), которое может отличаться от этого. | int | 1048576 | [0,...] | средняя |
| **partitioner.class** | Класс, используемый для определения того, в какую партицию следует отправлять данные при создании записей. Доступные варианты:   * Если не установлено, используется логика партиционирования по умолчанию. Эта стратегия будет пытаться придерживаться партиции до тех пор, пока в партиции не будет создано хотя бы байт пакетного размера. Работает по стратегии:   + Если партиция не указана, но ключ присутствует, выбирается партиция на основе хеша ключа.   + Если партиция или ключ отсутствуют, выбирается прикреплённая партиция, которая изменяется, когда в партиции создаётся хотя бы байт пакетного размера. * **org.apache.kafka.clients.producer.RoundRobinPartitioner**: эта стратегия партиционирования заключается в том, что каждая запись в серии последовательных записей будет отправлена в разные партиции (независимо от того, указан ли ключ или нет), пока не закончатся партиции и начнётся заново.   Имплементация интерфейса **org.apache.kafka.clients.producer.Partitioner** позволяет подключить кастомный партиционер. | class | null | -- | средняя |
| **partitioner.ignore.keys** | Если установлено значение **true**, поставщик не будет использовать ключи записи для выбора партиции. Если значение **false**, поставщик выберет партицию на основе хеша ключа, если ключ присутствует. **Примечание**. Этот параметр не действует, если используется кастомный партиционер. | boolean | false | -- | средняя |
| **receive.buffer.bytes** | Размер приёмного буфера TCP (SO\_RCVBUF), который будет использоваться при чтении данных. Если значение равно **-1**, будет использоваться значение ОС по умолчанию. | int | 32768 (32 кибибайт) | [-1,...] | средняя |
| **request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. Оно должно быть больше, чем **replica.lag.time.max.ms** (конфигурация брокера), чтобы уменьшить вероятность дублирования сообщений из-за ненужных повторных попыток поставщика. | int | 30000 (30 секунд) | [0,...] | средняя |
| **sasl.client.callback.handler.class** | Полное имя класса обработчика обратного вызова клиента SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. | class | null | -- | средняя |
| **sasl.jaas.config** | Параметры контекста логирования JAAS для соединений SASL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан [здесь](https://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html). Формат значения: **loginModuleClass controlFlag (optionName=optionValue)\*;**. Для брокеров конфигурация должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, l**istener.name.sasl\_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;** | password | null | -- | средняя |
| **sasl.kerberos.service.name** | Имя принципала Kerberos, под которым работает RT.StreamingKafka. Параметр можно определить либо в конфигурации JAAS RT.StreamingKafka, либо в конфигурации RT.StreamingKafka. | string | null | -- | средняя |
| **sasl.login.callback.handler.class** | Полное имя класса обработчика обратного вызова для логирования SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. Для брокеров конфигурация обработчика обратного вызова при входе должна иметь префикс слушателя и имя механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example. CustomScramLoginCallbackHandler** | class | null | -- | средняя |
| **sasl.login.class** | Полное имя класса, имплементирующего интерфейс логирования. Для брокеров конфигурация логирования должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin** | class | null | -- | средняя |
| **sasl.mechanism** | Механизм SASL, используемый для клиентских подключений. Это может быть любой механизм, для которого доступен поставщик безопасности. **GSSAPI**— механизм по умолчанию. | string | GSSAPI | -- | средняя |
| **sasl.oauthbearer.jwks.endpoint.url** | URL-адрес поставщика OAuth/OIDC, из которого можно получить [JWKS (набор веб-ключей JSON)](https://datatracker.ietf.org/doc/html/rfc7517#section-5) поставщика. URL-адрес может быть основан на HTTP(S) или на основе файла. Если URL-адрес основан на HTTP(S), данные JWKS будут получены от поставщика OAuth/OIDC через настроенный URL-адрес при запуске брокера. Все текущие ключи будут кэшироваться на брокере для входящих запросов. Если получен запрос аутентификации для JWT, который включает значение утверждения заголовка **kid**, которого ещё нет в кэше, эндпоинт JWKS будет запрошен снова по требованию. Однако брокер опрашивает URL-адрес каждые **sasl.oauthbearer.jwks.endpoint.refresh.ms** миллисекунды, чтобы обновить кэш любыми предстоящими ключами до того, как будут получены любые запросы JWT, которые их включают. Если URL-адрес основан на файле, брокер загрузит файл JWKS из настроенного местоположения при запуске. Если JWT включает значение заголовка **kid**, которого нет в файле JWKS, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | средняя |
| **sasl.oauthbearer.token.endpoint.url** | URL-адрес поставщика удостоверений OAuth/OIDC. Если URL-адрес основан на HTTP(S), это URL-адрес эндпоинта токена эмитента, к которому будут отправляться запросы на вход в систему на основе конфигурации в **sasl.jaas.config**. Если URL-адрес основан на файле, он указывает файл, содержащий токен доступа (в сериализованной форме JWT), выданный поставщиком удостоверений OAuth/OIDC для использования для авторизации. | string | null | -- | средняя |
| **security.protocol** | Протокол, используемый для связи с брокерами. Допустимые значения: **PLAINTEXT**, **SSL**, **SASL\_PLAINTEXT**, **SASL\_SSL**. | string | PLAINTEXT | [PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL] | средняя |
| **send.buffer.bytes** | Размер буфера отправки TCP (SO\_SNDBUF), который будет использоваться при отправке данных. Если значение равно **-1**, будет использоваться значение ОС по умолчанию. | int | 131072 (128 кибибайт) | [-1,...] | средняя |
| **socket.connection.setup.timeout.max.ms** | Максимальное время, в течение которого клиент будет ожидать установления соединения с сокетом. Таймаут установки соединения будет увеличиваться экспоненциально для каждого последующего сбоя соединения вплоть до этого максимума. Чтобы избежать одновременного наплыва большого количества соединения, к таймауту будет применён коэффициент рандомизации **0.2**, что приведёт к случайному диапазону между 20% ниже и 20 % выше вычисленного значения. | long | 30000 (30 секунд) | -- | средняя |
| **socket.connection.setup.timeout.ms** | Время, в течение которого клиент будет ожидать установления соединения с сокетом. Если соединение не будет установлено до истечения времени ожидания, клиенты закроют канал сокета. | long | 10000 (10 секунд) | -- | средняя |
| **ssl.enabled.protocols** | Список протоколов, включённых для SSL-соединений. По умолчанию используется значение **TLSv1.2,TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. При значении по умолчанию для Java 11 клиенты и серверы предпочтут **TLSv1.3**, если оба поддерживают его, и выберут **TLSv1.2** в противном случае (при условии, что оба поддерживают как минимум TLSv1.2). Это значение по умолчанию должно подойти для большинства случаев. Также смотри информацию о конфигурации **ssl.protocol**. | list | TLSv1.2 | -- | средняя |
| **ssl.keystore.type** | Формат файла хранилища ключей. Параметр необязателен для клиента. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **ssl.protocol** | Протокол SSL, используемый для создания SSLContext. По умолчанию используется **TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. Это значение должно подойти для большинства случаев использования. Разрешёнными значениями в последних JVM являются **TLSv1.2** и **TLSv1.3**. **TLS**, **TLSv1.1**, **SSL**, **SSLv2** и **SSLv3**могут поддерживаться в старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности. При значении по умолчанию для этой конфигурации и **ssl.enabled.protocols** клиенты перейдут на **TLSv1.2**, если сервер не поддерживает **TLSv1.3**. Если для этой конфигурации установлено значение **TLSv1.2**, клиенты не будут использовать **TLSv1.3**, даже если это одно из значений в **ssl.enabled.protocols**, а сервер поддерживает только **TLSv1.3**. | string | TLSv1.2 | -- | средняя |
| **ssl.provider** | Имя поставщика безопасности, используемого для SSL-соединений. Значение по умолчанию — поставщик безопасности по умолчанию для JVM. | string | null | -- | средняя |
| **ssl.truststore.type** | Формат файла хранилища доверенных сертификатов. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **acks** | Количество подтверждений, которые поставщик требует получить от лидера, прежде чем считать запрос выполненным. Параметр контролирует долговечность отправляемых записей. Допускаются следующие настройки:   * **acks=0** — если установлено значение **0**, то поставщик вообще не будет ждать подтверждения от сервера. Запись будет немедленно добавлена в буфер сокета и считается отправленной. В этом случае нельзя гарантировать, что сервер получил запись, и конфигурация **retries**не вступит в силу (поскольку клиент обычно не узнает о каких-либо сбоях). Смещение, возвращаемое для каждой записи, всегда будет установлено на **-1**. * **acks=1** — это будет означать, что лидер запишет запись в свой локальный лог, но ответит, не дожидаясь полного подтверждения от всех подписчиков. В этом случае, если лидер выйдет из строя сразу после подтверждения записи, но до того, как подписчики реплицируют её, запись будет потеряна. * **acks=all** — это означает, что лидер будет ждать, пока полный набор синхронизированных реплик подтвердит запись. Это гарантирует, что запись не будет потеряна, пока жива хотя бы одна синхронизированная реплика. Это самая надёжная гарантия. Это эквивалентно настройке **acks=-1**.   Обратите внимание, что включение идемпотентности требует, чтобы это значение конфигурации было **all**. Если заданы конфликтующие конфигурации и идемпотентность не включена явно, идемпотентность отключается. | string | all | [all, -1, 0, 1] | низкая |
| **auto.include.jmx.reporter** | **[Устарело]** Следует ли автоматически включать JmxReporter, даже если он не указан в **metric.reporters**. Эта конфигурация будет удалена в RT.StreamingKafka 4.0, вместо этого пользователям следует включить **org.apache.kafka.common.metrics.JmxReporter** в **metric.reporters**, чтобы включить JmxReporter. | boolean | true | -- | низкая |
| **enable.idempotence** | Если установлено значение **true**, поставщик гарантирует, что в стрим будет записана ровно одна копия каждого сообщения. Если установлено значение **false**, поставщик повторяет попытки из-за сбоев брокера и т.д. может записывать в стрим дубликаты повторного сообщения. Обратите внимание, что для включения идемпотентности требуется, чтобы **max.in.flight.requests.per.connection** было меньше или равно **5** (с сохранением порядка сообщений для любого допустимого значения), количество **retries**должно быть больше **0**, а **acks**должны быть **all**.  Идемпотентность включена по умолчанию, если не заданы конфликтующие конфигурации. Если заданы конфликтующие конфигурации и идемпотентность не включена явно, идемпотентность отключается. Если идемпотентность включена явно и заданы конфликтующие конфигурации, создается исключение **ConfigException**. | boolean | true | -- | низкая |
| **interceptor.classes** | Список классов для использования в качестве перехватчиков. Имплементация интерфейса **org.apache.kafka.clients.producer.ProducerInterceptor** позволяет перехватывать (и, возможно, изменять) записи, полученные поставщиком, прежде чем они будут опубликованы в кластере RT.StreamingKafka. По умолчанию перехватчиков нет. | list | "" | non-null string | низкая |
| **max.in.flight.requests.per.connection** | Максимальное количество неподтверждённых запросов, которые клиент отправит за одно соединение перед блокировкой. Обратите внимание: если для этой конфигурации установлено значение больше **1**, а для параметра **enable.idempotence** установлено значение **false**, существует риск изменения порядка сообщений после неудачной отправки из-за повторных попыток (т.е. если **retries**включены); если повторные попытки отключены или если для параметра **enable.idempotence** установлено значение **true**, порядок будет сохранен. Кроме того, для включения идемпотентности требуется, чтобы значение этой конфигурации было меньше или равно **5**. Если установлены конфликтующие конфигурации и идемпотентность не включена явно, идемпотентность отключается. | int | 5 | [1,...] | низкая |
| **metadata.max.age.ms** | Период времени в миллисекундах, по истечении которого мы принудительно обновляем метаданные, даже если мы не заметили никаких изменений лидерства партиции, чтобы заранее обнаружить любые новые брокеры или партиции. | long | 300000 (5 минут) | [0,...] | низкая |
| **metadata.max.idle.ms** | Управляет тем, как долго поставщик будет кэшировать метаданные для бездействующего топика. Если время, прошедшее с момента последнего создания топика, превышает продолжительность простоя метаданных, то метаданные топика забываются, и при следующем доступе к ней будет выполнен запрос на получение метаданных. | long | 300000 (5 минут) | [5000,...] | низкая |
| **metric.reporters** | Список классов для использования в качестве генераторов отчётов по метрикам. Имплементация интерфейса **org.apache.kafka.common.metrics.MetricsReporter** позволяет подключать классы, которые будут уведомляться о создании новой метрики. JmxReporter всегда включён для регистрации статистики JMX. | list | "" | non-null string | низкая |
| **metrics.num.samples** | Количество примеров, поддерживаемых для вычисления метрик. | int | 2 | [1,...] | низкая |
| **metrics.recording.level** | Самый высокий уровень записи метрик. | string | INFO | [INFO, DEBUG, TRACE] | низкая |
| **metrics.sample.window.ms** | Окно времени, в течение которого вычисляется пример метрик. | long | 30000 (30 секунд) | [0,...] | низкая |
| **partitioner.adaptive.partitioning.enable** | Если установлено значение **true**, поставщик попытается адаптироваться к производительности брокера и отправлять больше сообщений в партиции, размещённые на более быстрых брокерах. Если значение **false**, поставщик попытается распределить сообщения равномерно. **Примечание**. Этот параметр не действует, если используется кастомный партиционер. | boolean | true | -- | низкая |
| **partitioner.availability.timeout.ms** | Если брокер не может обработать запросы из партиции в течение времени **partioner.availability.timeout.ms**, партиционер считает эту партицию недоступной. Если значение равно **0**, эта логика отключена. **Примечание**. Этот параметр не действует, если используется кастомный партиционер или для параметра **partitioner.adaptive.partitioning.enable** установлено значение **false**. | long | 0 | [0,...] | низкая |
| **reconnect.backoff.max.ms** | Максимальное время ожидания в миллисекундах при повторном подключении к брокеру, к которому неоднократно не удавалось подключиться. Если параметр предусмотрен, отсрочка на хост будет увеличиваться экспоненциально для каждого последующего сбоя соединения, вплоть до этого максимума. После расчёта увеличения задержки добавляется 20% случайного джиттера, чтобы избежать наплыва большого количества одновременных соединений. | long | 1000 (1 секунда) | [0,...] | низкая |
| **reconnect.backoff.ms** | Базовое время ожидания перед попыткой повторного подключения к данному хосту. Параметр позволяет избежать повторных подключений к хосту в плотном цикле. Эта отсрочка применяется ко всем попыткам подключения клиента к брокеру. | long | 50 | [0,...] | низкая |
| **retry.backoff.ms** | Время ожидания перед попыткой повторения неудачного запроса к данной партиции топика. Параметр позволяет избежать повторной отправки запросов в плотном цикле в некоторых сценариях сбоя. | long | 100 | [0,...] | низкая |
| **sasl.kerberos.kinit.cmd** | Путь к команде Kerberos **kinit**. | string | /usr/bin/kinit | -- | низкая |
| **sasl.kerberos.min.time.before.relogin** | Время ожидания потока входа между попытками обновления. | long | 60000 | -- | низкая |
| **sasl.kerberos.ticket.renew.jitter** | Процент случайного джиттера, добавленного ко времени обновления. | double | 0.05 | -- | низкая |
| **sasl.kerberos.ticket.renew.window.factor** | Поток входа будет находиться в спящем режиме до тех пор, пока не будет достигнут указанный временной интервал от последнего обновления до истечения срока действия билета, после чего он попытается продлить билет. | double | 0.8 | -- | низкая |
| **sasl.login.connect.timeout.ms** | [Опциональное] значение таймаута подключения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к **OAUTHBEARER**. | int | null | -- | низкая |
| **sasl.login.read.timeout.ms** | [Опциональное] значение таймаута чтения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к **OAUTHBEARER**. | int | null | -- | низкая |
| **sasl.login.refresh.buffer.seconds** | Время буферизации до истечения срока действия учётных данных, которое необходимо поддерживать при обновлении учётных данных, в секундах. Если в противном случае обновление произойдёт ближе к истечению срока действия, чем количество буферных секунд, тогда обновление будет перенесено вверх, чтобы сохранить как можно большую часть буферного времени. Допустимые значения: от **0** до **3600**(1 час). Если значение не указано, используется значение по умолчанию **300**(5 минут). Это значение и **sasl.login.refresh.min.period.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к **OAUTHBEARER**. | short | 300 | [0,...,3600] | низкая |
| **sasl.login.refresh.min.period.seconds** | Желаемое минимальное время ожидания потока обновления входа перед обновлением учётных данных в секундах. Допустимые значения: от **0** до **900**(15 минут). Если значение не указано, используется значение по умолчанию **60**(1 минута). Это значение и **sasl.login.refresh.buffer.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к **OAUTHBEARER**. | short | 60 | [0,...,900] | низкая |
| **sasl.login.refresh.window.factor** | Поток обновления входа в систему будет находиться в режиме ожидания до тех пор, пока не будет достигнут указанный коэффициент окна относительно срока действия учётных данных, после чего он попытается обновить учётные данные. Допустимые значения находятся в пределах от **0.5** (50%) до **1.0** (100%) включительно. Если значение не указано, используется значение по умолчанию **0.8** (80%). В настоящее время применяется только к **OAUTHBEARER**. | double | 0.8 | [0.5,...,1.0] | низкая |
| **sasl.login.refresh.window.jitter** | Максимальное количество случайного джиттера относительно срока действия учётных данных, которое добавляется ко времени сна потока обновления входа в систему. Допустимые значения находятся в диапазоне от **0** до **0.25** (25%) включительно. Если значение не указано, используется значение по умолчанию **0.05** (5%). В настоящее время применяется только к **OAUTHBEARER**. | double | 0.05 | [0.0,...,0.25] | низкая |
| **sasl.login.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к **OAUTHBEARER**. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.login.retry.backoff.ms** | [Опциональное] значение в миллисекундах для начального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к **OAUTHBEARER**. | long | 100 | -- | низкая |
| **sasl.oauthbearer.clock.skew.seconds** | [Опциональное] значение в секундах, позволяющее учитывать разницу между временем поставщика удостоверений OAuth/OIDC и брокера. | int | 30 | -- | низкая |
| **sasl.oauthbearer.expected.audience** | [Опциональный] параметр, разделённый запятыми, который брокер будет использовать для проверки того, что JWT был выпущен для одной из ожидаемых аудиторий. JWT будет проверен на наличие стандартного утверждения OAuth **aud**, и если это значение установлено, брокер сопоставит значение из утверждения **aud**JWT, чтобы увидеть, есть ли точное совпадение. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | list | null | -- | низкая |
| **sasl.oauthbearer.expected.issuer** | [Опциональный] параметр, который брокер должен использовать для проверки того, что JWT был создан ожидаемым эмитентом. JWT будет проверен на наличие стандартного утверждения OAuth **iss**, и если это значение установлено, брокер будет точно сопоставлять его с тем, что находится в утверждении **iss**JWT. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.refresh.ms** | [Опциональное] значение в миллисекундах, в течение которого брокер должен ждать между обновлением своего кэша JWKS (набора веб-ключей JSON), содержащего ключи для проверки подписи JWT. | long | 3600000 (1 час) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками получить JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms** | [Опциональное] значение в миллисекундах для первоначального ожидания между попытками получения JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 100 | -- | низкая |
| **sasl.oauthbearer.scope.claim.name** | Утверждение OAuth для области часто называется **scope**, но этот [опциональный] параметр может предоставить другое имя для области, включённой в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | scope | -- | низкая |
| **sasl.oauthbearer.sub.claim.name** | Утверждение OAuth для субъекта часто называется **sub**, но этот [опциональный] параметр может предоставить другое имя для субъекта, включённого в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | sub | -- | низкая |
| **security.providers** | Список настраиваемых классов-создателей, каждый из которых возвращает поставщика, реализующего алгоритмы безопасности. Эти классы должны имплементировать интерфейс **org.apache.kafka.common.security.auth.SecurityProviderCreator**. | string | null | -- | низкая |
| **ssl.cipher.suites** | Список наборов шифров. Это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL. По умолчанию поддерживаются все доступные наборы шифров. | list | null | -- | низкая |
| **ssl.endpoint.identification.algorithm** | Алгоритм идентификации эндпоинта для проверки имени хоста сервера с использованием сертификата сервера. | string | https | -- | низкая |
| **ssl.engine.factory.class** | Класс типа **org.apache.kafka.common.security.auth.SslEngineFactory** для предоставления объектов SSLEngine. Значение по умолчанию — **org.apache.kafka.common.security.ssl.DefaultSslEngineFactory**. | class | null | -- | низкая |
| **ssl.keymanager.algorithm** | Алгоритм, используемый менеджером ключей для SSL-соединений. Значением по умолчанию является алгоритм менеджера ключей, настроенный для Java Virtual Machine. | string | SunX509 | -- | низкая |
| **ssl.secure.random.implementation** | Имплементация SecureRandom PRNG для использования в операциях шифрования SSL. | string | null | -- | низкая |
| **ssl.trustmanager.algorithm** | Алгоритм, используемый менеджером доверия для SSL-соединений. Значением по умолчанию является алгоритм диспетчера доверия, настроенный для Java Virtual Machine. | string | PKIX | -- | низкая |
| **transaction.timeout.ms** | Максимальное время в мс, в течение которого координатор транзакции будет ждать обновления статуса транзакции от поставщика, прежде чем превентивно прервать текущую транзакцию. Если это значение больше, чем настройка **transaction.max.timeout.ms** в брокере, запрос завершится неудачей с ошибкой **InvalidTxnTimeoutException**. | int | 60000 (1 минута) | -- | низкая |
| **transactional.id** | **TransactionalId**, используемый для доставки транзакций. Параметр обеспечивает семантику надёжности, которая охватывает несколько сеансов поставщика, поскольку позволяет клиенту гарантировать, что транзакции, использующие один и тот же **TransactionalId**, были завершены до начала любых новых транзакций. Если **TransactionalId**не указан, то поставщик ограничен идемпотентной доставкой. Если настроен **TransactionalId**, подразумевается **enable.idempotence**. По умолчанию **TransactionId**не настроен, что означает, что транзакции использовать нельзя. Обратите внимание, что по умолчанию для транзакций требуется кластер как минимум из трёх брокеров, что является рекомендуемой настройкой для продуктивной среды; для разработки вы можете изменить это, изменив настройки брокера **transaction.state.log.replication.factor**. | string | null | non-empty string | низкая |

## 2.4 Конфигурация потребителя

Ниже представлена конфигурация потребителя.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **key.deserializer** | Класс десериализатора для ключа, имплементирующего интерфейс **org.apache.kafka.common.serialization.Deserializer**. | class | -- | -- | высокая |
| **value.deserializer** | Класс десериализатора для значения, имплементирующего интерфейс **org.apache.kafka.common.serialization.Deserializer**. | class | -- | -- | высокая |
| **bootstrap.servers** | Список пар хост/порт, которые будут использоваться для установления инициирующего подключения к кластеру RT.StreamingKafka. Клиент будет использовать все серверы независимо от того, какие серверы указаны здесь для начальной загрузки — этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Этот список должен иметь форму **host1:port1,host2:port2,...**. Поскольку эти серверы используются только для инициирующего подключения для обнаружения полного состава кластера (которое может меняться динамически), этот список не обязательно должен содержать полный набор серверов (хотя вам может понадобиться больше одного, если сервер не работает). | list | "" | non-null string | высокая |
| **fetch.min.bytes** | Минимальный объём данных, который сервер должен вернуть для запроса на выборку. Если данных недостаточно, запрос будет ждать, пока накопится необходимое количество данных, прежде чем ответить на запрос. Значение по умолчанию, равное 1 байту, означает, что на запросы выборки будет дан ответ, как только становится доступен один байт данных или время ожидания запроса выборки истекает. Установка значения больше 1 приведёт к тому, что сервер будет ждать накопления большего объёма данных, что может немного повысить пропускную способность сервера за счёт некоторой дополнительной задержки. | int | 1 | [0,...] | высокая |
| **group.id** | Уникальная строка, идентифицирующая группу потребителей, к которой принадлежит этот потребитель. Это свойство является обязательным, если потребитель использует либо функцию управления группами с помощью **subscribe(topic)**, либо стратегию управления смещением на основе RT.Streaming Kafka. | string | null | -- | высокая |
| **heartbeat.interval.ms** | Ожидаемое время между отправкой heartbeat-сообщений координатору потребителей при использовании средств управления группами RT.StreamingKafka. Heartbeat-сообщения используются для обеспечения активности сеанса потребителя и для облегчения перебалансировки, когда новые потребители присоединяются к группе или покидают её. Значение должно быть меньше, чем **session.timeout.ms**, но обычно его следует устанавливать не выше 1/3 этого значения. Его можно настроить ещё ниже, чтобы контролировать ожидаемое время нормальной ребалансировки. | int | 3000 (3 секунды) | -- | высокая |
| **max.partition.fetch.bytes** | Максимальный объём данных для каждой партиции, который будет возвращать сервер. Записи извлекаются потребителем пакетами. Если первый пакет записей в первой непустой партиции выборки превышает этот предел, пакет всё равно будет возвращён, чтобы гарантировать, что потребитель может продолжить работу. Максимальный размер пакета записей, принимаемый брокером, определяется через **message.max.bytes** (конфигурация брокера) или **max.message.bytes** (конфигурация топика). См. **fetch.max.bytes** для ограничения размера запроса потребителя. | int | 1048576 (1 мебибайт) | [0,...] | высокая |
| **session.timeout.ms** | Таймаут, используемый для обнаружения сбоев клиента при использовании средства управления группами RT.StreamingKafka. Клиент периодически отправляет сигналы подтверждения (heartbeat-сообщения), чтобы сообщить брокеру о своей работоспособности. Если до истечения таймаута этого сеанса брокер не получит никаких heartbeat-сообщений, то брокер удалит этого клиента из группы и инициирует ребалансировку. Обратите внимание, что значение должно находиться в допустимом диапазоне, настроенном в конфигурации брокера с помощью **group.min.session.timeout.ms** и **group.max.session.timeout.ms**. | int | 45000 (45 секунд) | -- | высокая |
| **ssl.key.password** | Пароль закрытого ключа в файле хранилища ключей или ключ PEM, указанный в **ssl.keystore.key**. | password | null | -- | высокая |
| **ssl.keystore.certificate.chain** | Цепочка сертификатов в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM со списком сертификатов X.509. | password | null | -- | высокая |
| **ssl.keystore.key** | Закрытый ключ в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с ключами PKCS#8. Если ключ зашифрован, пароль ключа необходимо указать с помощью **ssl.key.password**. | password | null | -- | высокая |
| **ssl.keystore.location** | Расположение файла хранилища ключей. Это необязательно для клиента и может использоваться для двусторонней аутентификации клиента. | string | null | -- | высокая |
| **ssl.keystore.password** | Пароль хранилища для файла хранилища ключей. Это необязательно для клиента и необходимо только в том случае, если настроен **ssl.keystore.location**. Пароль хранилища ключей не поддерживается для формата PEM. | password | null | -- | высокая |
| **ssl.truststore.certificates** | Доверенные сертификаты в формате, указанном в параметре **ssl.truststore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с сертификатами X.509. | password | null | -- | высокая |
| **ssl.truststore.location** | Расположение файла хранилища доверенных сертификатов. | string | null | -- | высокая |
| **ssl.truststore.password** | Пароль для файла хранилища доверенных сертификатов. Если пароль не установлен, настроенный файл хранилища доверенных сертификатов по-прежнему будет использоваться, но проверка целостности будет отключена. Пароль хранилища доверенных сертификатов не поддерживается для формата PEM. | password | null | -- | высокая |
| **allow.auto.create.topics** | Разрешает автоматическое создание топиков на брокере при подписке или назначении топика. Топик, на который вы подписаны, будет создан автоматически, только если брокер разрешает это с помощью конфигурации брокера **auto.create.topics.enable**. Для этой конфигурации необходимо установить значение **false**при использовании брокеров старше 0.11.0. | boolean | true | -- | средняя |
| **auto.offset.reset** | Что делать, если в RT.StreamingKafka нет начального смещения или если текущее смещение больше не существует на сервере (например, потому что эти данные были удалены):   * **earliest**-- автоматически сбрасывает смещение на самое раннее смещение. * **latest**-- автоматически сбрасывает смещение на последнее смещение. * **none**-- выдаёт исключение потребителю, если для группы потребителя не найдено предыдущее смещение. * что-нибудь еще -- выдаёт исключение потребителю. | string | latest | [latest, earliest, none] | средняя |
| **client.dns.lookup** | Управляет тем, как клиент использует поиск DNS. Если установлено значение **use\_all\_dns\_ips**, подключается к каждому возвращаемому IP-адресу последовательно, пока не будет установлено успешное соединение. После отключения используется следующий IP. После того, как все IP-адреса были использованы один раз, клиент снова разрешает IP-адреса из имени хоста (однако поиск DNS-имён как в JVM, так и в кэше ОС). Если установлено значение **resolve\_canonical\_bootstrap\_servers\_only**, преобразует каждый адрес начальной загрузки в список канонических имён. После фазы начальной загрузки это ведёт себя так же, как **use\_all\_dns\_ips**. | string | use\_all\_dns\_ips | [use\_all\_dns\_ips, resolve\_canonical\_bootstrap\_ servers\_only] | средняя |
| **connections.max.idle.ms** | Закрывает неактивные соединения по истечении количества миллисекунд, указанного в этой конфигурации. | long | 540000 (9 минут) | -- | средняя |
| **default.api.timeout.ms** | Указывает таймаут (в миллисекундах) для клиентских API. Эта конфигурация используется в качестве таймаута по умолчанию для всех клиентских операций, для которых не указан параметр **timeout**. | int | 60000 (1 минута) | [0,...] | средняя |
| **enable.auto.commit** | Если установлено **true**, смещение потребителя будет периодически фиксироваться в фоновом режиме. | boolean | true | -- | средняя |
| **exclude.internal.topics** | Следует ли исключать из подписки внутренние топики, соответствующие шаблону подписки. Всегда можно явно подписаться на внутренний топик. | boolean | true | -- | средняя |
| **fetch.max.bytes** | Максимальный объём данных, который сервер должен вернуть для запроса на выборку. Записи извлекаются потребителем пакетами, и если первый пакет записей в первой непустой партиции выборки больше этого значения, пакет записей всё равно будет возвращён, чтобы гарантировать, что потребитель может продолжить работу. Таким образом, это не абсолютный максимум. Максимальный размер пакета записей, принимаемый брокером, определяется через **message.max.bytes** (конфигурация брокера) или **max.message.bytes** (конфигурация топика). Обратите внимание, что потребитель выполняет несколько выборок параллельно. | int | 52428800 (50 мебибайт) | [0,...] | средняя |
| **group.instance.id** | Уникальный идентификатор инстанса потребителя, предоставленный конечным пользователем. Разрешены только непустые строки. Если этот параметр установлен, потребитель рассматривается как статический член, а это означает, что в любой момент времени в группе потребителей разрешён только один инстанс с этим идентификатором. Это можно использовать в сочетании с большим таймаутом сеанса, чтобы избежать перебалансировки групп, вызванной временной недоступностью (например, перезапуском процесса). Если этот параметр не установлен, потребитель присоединится к группе в качестве динамического члена, что является традиционным поведением. | string | null | non-empty string | средняя |
| **isolation.level** | Управляет чтением сообщений, написанных транзакционно. Если установлено значение **read\_commited**, **consumer.poll()** будет возвращать только те транзакционные сообщения, которые были закоммичены. Если установлено значение **read\_uncommited** (по умолчанию), **consumer.poll()** будет возвращать все сообщения, даже транзакционные сообщения, которые были прерваны. Нетранзакционные сообщения будут возвращаться безоговорочно в любом режиме.  Сообщения всегда будут возвращаться в порядке смещения. Следовательно, в режиме **read\_committed** метод **consumer.poll()** будет возвращать сообщения только до последнего стабильного смещения (last stable offset, LSO), которое меньше смещения первой открытой транзакции. В частности, любые сообщения, появляющиеся после сообщений, принадлежащих текущим транзакциям, будут храниться до тех пор, пока соответствующая транзакция не будет завершена. В результате потребители **read\_commited** не смогут читать до high watermark (максимальное доступное смещение для чтения), когда есть незавершённые транзакции.  Кроме того, когда в **read\_commit** метод **seekToEnd** вернет LSO. | string | read\_uncommitted | [read\_committed, read\_uncommitted] | средняя |
| **max.poll.interval.ms** | Максимальная задержка между вызовами **poll()** при использовании управления группами потребителей. Это устанавливает верхнюю границу времени, в течение которого потребитель может простаивать, прежде чем получить больше записей. Если **poll()** не вызывается до истечения этого таймаута, то потребитель считается неработоспособным, и группа выполнит ребалансировку, чтобы переназначить партиции другому участнику. Для потребителей, использующих ненулевой **group.instance.id**, достигших этого времени ожидания, партиции не будут переназначены немедленно. Вместо этого потребитель прекратит отправку heartbeat-сообщений, а партиции будут переназначены после истечения срока действия **session.timeout.ms**. Это отражает поведение статического потребителя, который отключился. | int | 300000 (5 минут) | [1,...] | средняя |
| **max.poll.records** | Максимальное количество записей, возвращаемых за один вызов **poll()**. Обратите внимание, что **max.poll.records** не влияет на базовое поведение выборки. Потребитель будет кэшировать записи из каждого запроса на выборку и постепенно возвращать их из каждого опроса. | int | 500 | [1,...] | средняя |
| **partition.assignment.strategy** | Список имён классов или типов классов, упорядоченных по предпочтениям, поддерживаемых стратегий назначения партиций, которые клиент будет использовать для распределения владения партициями между  инстансами-потребителями при использовании управления группами. Доступные варианты:   * **org.apache.kafka.clients.consumer.RangeAssignor** -- назначает партиции для каждого топика. * **org.apache.kafka.clients.consumer.RoundRobinAssignor** -- назначает партиции потребителям в циклическом порядке. * **org.apache.kafka.clients.consumer.StickyAssignor** -- гарантирует максимально сбалансированное назначение при сохранении как можно большего количества существующих назначений партиций. * **org.apache.kafka.clients.consumer.CooperativeStickyAssignor** -- следует той же логике **StickyAssignor**, но позволяет выполнять совместную перебалансировку.   Значение по умолчанию является **[RangeAssignor, CooperativeStickyAssignor]**, которое будет использовать **RangeAssignor**по умолчанию, но позволяет выполнить обновление до **CooperativeStickyAssignor**всего одним движением, которое удаляет **RangeAssignor**из списка.  Имплементация интерфейса **org.apache.kafka.clients.consumer.ConsumerPartitionAssignor** позволяет вам подключить кастомную стратегию назначения. | list | class org.apache.kafka.clients. consumer.RangeAssignor,class org.apache.kafka.clients. consumer.CooperativeStickyAssignor | non-null string | средняя |
| **receive.buffer.bytes** | Размер приёмного буфера TCP (SO\_RCVBUF), который будет использоваться при чтении данных. Если значение равно**-1**, будет использоваться ОС по умолчанию. | int | 65536 (64 кибибайт) | [-1,...] | средняя |
| **request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. | int | 30000 (30 секунд) | [0,...] | средняя |
| **sasl.client.callback.handler.class** | Полное имя класса обработчика обратного вызова клиента SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. | class | null | -- | средняя |
| **sasl.jaas.config** | Параметры контекста входа в систему JAAS для соединений SASL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан [здесь](http://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html). Формат значения: **loginModuleClass controlFlag (optionName=optionValue)\*;**. Для брокеров конфигурация должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, требуется **listener.name.sasl\_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;** | password | null | -- | средняя |
| **sasl.kerberos.service.name** | Имя принципала Kerberos, под которым работает RT.StreamingKafka. Это можно определить либо в конфигурации JAAS RT.StreamingKafka, либо в конфигурации RT.StreamingKafka. | string | null | -- | средняя |
| **sasl.login.callback.handler.class** | Полное имя класса обработчика обратного вызова для входа в систему SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. Для брокеров конфигурация обработчика обратного вызова при входе должна иметь префикс слушателя и имя механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.callback.handler.class= com.example.CustomScramLoginCallbackHandler**. | class | null | -- | средняя |
| **sasl.login.class** | Полное имя класса, имплементирующего интерфейс входа в систему. Для брокеров конфигурация входа должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin**. | class | null | -- | средняя |
| **sasl.mechanism** | Механизм SASL, используемый для клиентских подключений. Это может быть любой механизм, для которого доступен поставщик безопасности. **GSSAPI**— механизм по умолчанию. | string | GSSAPI | -- | средняя |
| **sasl.oauthbearer.jwks.endpoint.url** | URL-адрес поставщика OAuth/OIDC, из которого можно получить [JWKS (набор веб-ключей JSON)](https://datatracker.ietf.org/doc/html/rfc7517#section-5) поставщика. URL-адрес может быть основан на HTTP(S) или на основе файла. Если URL-адрес основан на HTTP(S), данные JWKS будут получены от поставщика OAuth/OIDC через настроенный URL-адрес при запуске брокера. Все текущие ключи будут кэшироваться на брокере для входящих запросов. Если получен запрос аутентификации для JWT, который включает значение утверждения заголовка **kid**, которого ещё нет в кэше, эндпоинт JWKS будет запрошен снова по требованию. Однако брокер опрашивает URL-адрес каждые **sasl.oauthbearer.jwks.endpoint.refresh.ms** миллисекунды, чтобы обновить кэш любыми предстоящими ключами до того, как будут получены любые запросы JWT, которые их включают. Если URL-адрес основан на файле, брокер загрузит файл JWKS из настроенного местоположения при запуске. Если JWT включает значение заголовка **kid**, которого нет в файле JWKS, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | средняя |
| **sasl.oauthbearer.token.endpoint.url** | URL-адрес поставщика удостоверений OAuth/OIDC. Если URL-адрес основан на HTTP(S), это URL-адрес эндпоинта токена эмитента, к которому будут отправляться запросы на вход в систему на основе конфигурации в **sasl.jaas.config**. Если URL-адрес основан на файле, он указывает файл, содержащий токен доступа (в сериализованной форме JWT), выданный поставщиком удостоверений OAuth/OIDC для использования для авторизации. | string | null | -- | средняя |
| **security.protocol** | Протокол, используемый для связи с брокерами. Допустимые значения: **PLAINTEXT**, **SSL**, **SASL\_PLAINTEXT**, **SASL\_SSL**. | string | PLAINTEXT | [PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL] | средняя |
| **send.buffer.bytes** | Размер буфера отправки TCP (SO\_SNDBUF), который будет использоваться при отправке данных. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 131072 (128 кибибайт) | [-1,...] | средняя |
| **socket.connection.setup.timeout.max.ms** | Максимальное время, в течение которого клиент будет ожидать установления соединения с сокетом. Таймаут установки соединения будет увеличиваться экспоненциально для каждого последующего сбоя соединения вплоть до этого максимума. Чтобы избежать наплыва одновременных соединения, к таймауту будет применён коэффициент рандомизации 0.2, что приведёт к случайному диапазону между 20% ниже и 20% выше вычисленного значения. | long | 30000 (30 секунд) | -- | средняя |
| **socket.connection.setup.timeout.ms** | Время, в течение которого клиент будет ожидать установления соединения с сокетом. Если соединение не будет установлено до истечения времени ожидания, клиенты закроют канал сокета. | long | 10000 (10 секунд) | -- | средняя |
| **ssl.enabled.protocols** | Список протоколов, включённых для SSL-соединений. По умолчанию используется значение **TLSv1.2,TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. При значении по умолчанию для Java 11 клиенты и серверы предпочтут **TLSv1.3**, если оба поддерживают его, и откажутся от TLSv1.2 в противном случае (при условии, что оба поддерживают как минимум TLSv1.2). Это значение по умолчанию должно подойти для большинства случаев. Также смотри информацию по конфигурации для **ssl.protocol**. | list | TLSv1.2 | -- | средняя |
| **ssl.keystore.type** | Формат файла хранилища ключей. Это необязательно для клиента. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**:**[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **ssl.protocol** | Протокол SSL, используемый для создания SSLContext. По умолчанию используется **TLSv1.3**при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. Это значение должно подойти для большинства случаев использования. Разрешёнными значениями в последних JVM являются **TLSv1.2** и **TLSv1.3**. **TLS**, **TLSv1.1**, **SSL**, **SSLv2**и **SSLv3**могут поддерживаться в старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности. При значении по умолчанию для этой конфигурации и **ssl.enabled.protocols** клиенты перейдут на **TLSv1.2**, если сервер не поддерживает **TLSv1.3**. Если для этой конфигурации установлено значение **TLSv1.2**, клиенты не будут использовать **TLSv1.3**, даже если это одно из значений в **ssl.enabled.protocols**, а сервер поддерживает только **TLSv1.3**. | string | TLSv1.2 | -- | средняя |
| **ssl.provider** | Имя поставщика безопасности, используемого для SSL-соединений. Значение по умолчанию — поставщик безопасности по умолчанию для JVM. | string | null | -- | средняя |
| **ssl.truststore.type** | Формат файла хранилища доверенных сертификатов. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **auto.commit.interval.ms** | Частота в миллисекундах, с которой потребительские смещения автоматически фиксируются в RT.StreamingKafka, если для параметра **enable.auto.commit** установлено значение **true**. | int | 5000 (5 секунд) | [0,...] | низкая |
| **auto.include.jmx.reporter** | [**Устарело**] Следует ли автоматически включать JmxReporter, даже если он не указан в **metric.reporters**. Эта конфигурация будет удалена в Kafka 4.0, вместо этого пользователям следует включить **org.apache.kafka.common.metrics.JmxReporter** в **metric.reporters**, чтобы включить JmxReporter. | boolean | true | -- | низкая |
| **check.crcs** | Автоматически проверяет CRC32 потребляемых записей. Это гарантирует, что сообщения не будут повреждены ни в сети, ни на диске. Эта проверка добавляет некоторую нагрузку, поэтому её можно отключить в случаях, когда требуется максимальная производительность. | boolean | true | -- | низкая |
| **client.id** | Строка идентификатора, передаваемая на сервер при выполнении запросов. Целью этого является возможность отслеживать источник запросов, помимо IP/порта, позволяя включать логическое имя приложения в журнал запросов на стороне сервера. | string | "" | -- | низкая |
| **client.rack** | Идентификатор стойки для этого клиента. Это может быть любое строковое значение, указывающее, где физически находится этот клиент. Это соответствует конфигурации брокера **broker.rack**. | string | "" | -- | низкая |
| **fetch.max.wait.ms** | Максимальное количество времени, которое сервер будет блокировать перед ответом на запрос выборки, если данных недостаточно для немедленного удовлетворения требования, заданного **fetch.min.bytes**. | int | 500 | [0,...] | низкая |
| **interceptor.classes** | Список классов для использования в качестве перехватчиков. Имплементация интерфейса **org.apache.kafka.clients.consumer.ConsumerInterceptor** позволяет перехватывать (и, возможно, изменять) записи, полученные потребителем. По умолчанию перехватчиков нет. | list | "" | non-null string | низкая |
| **metadata.max.age.ms** | Период времени в миллисекундах, по истечении которого мы принудительно обновляемся метаданные, даже если мы не заметили никаких изменений лидерства партиции, чтобы заранее обнаружить любые новые брокеры или партиции. | long | 300000 (5 минут) | [0,...] | низкая |
| **metric.reporters** | Список классов для использования в качестве генераторов отчётов по метрикам. Имплементация интерфейса **org.apache.kafka.common.metrics.MetricsReporter** позволяет подключать классы, которые будут уведомляться о создании новой метрики. JmxReporter всегда включён для регистрации статистики JMX. | list | "" | non-null string | низкая |
| **metrics.num.samples** | Количество выборок, поддерживаемых для вычисления метрик. | int | 2 | [1,...] | низкая |
| **metrics.recording.level** | Самый высокий уровень записи метрик. | string | INFO | [INFO, DEBUG, TRACE] | низкая |
| **metrics.sample.window.ms** | Окно времени, в течение которого вычисляется выборка метрик. | long | 30000 (30 секунд) | [0,...] | низкая |
| **reconnect.backoff.max.ms** | Максимальное время ожидания в миллисекундах при повторном подключении к брокеру, которому неоднократно не удавалось подключиться. Если параметр предусмотрен, отсрочка на хост будет увеличиваться экспоненциально для каждого последующего сбоя соединения, вплоть до этого максимума. После расчёта увеличения задержки добавляется 20% случайного джиттера, чтобы избежать большого количества одновременных запросов. | long | 1000 (1 секунда) | [0,...] | низкая |
| **reconnect.backoff.ms** | Базовое время ожидания перед попыткой повторного подключения к данному хосту. Параметр позволяет избежать повторных подключений к хосту в плотном цикле. Эта отсрочка применяется ко всем попыткам подключения клиента к брокеру. | long | 50 | [0,...] | низкая |
| **retry.backoff.ms** | Время ожидания перед попыткой повторения неудачного запроса к данной партиции топика. Параметр позволяет избежать повторной отправки запросов в плотном цикле в некоторых сценариях сбоя. | long | 100 | [0,...] | низкая |
| **sasl.kerberos.kinit.cmd** | Путь к команде Kerberos **kinit**. | string | /usr/bin/kinit | -- | низкая |
| **sasl.kerberos.min.time.before.relogin** | Время ожидания потока входа между попытками обновления. | long | 60000 | -- | низкая |
| **sasl.kerberos.ticket.renew.jitter** | Процент случайного джиттера, добавленного ко времени обновления. | double | 0.05 | -- | низкая |
| **sasl.kerberos.ticket.renew.window.factor** | Поток входа будет находиться в спящем режиме до тех пор, пока не будет достигнут указанный временной интервал от последнего обновления до истечения срока действия билета, после чего он попытается продлить билет. | double | 0.08 | -- | низкая |
| **sasl.login.connect.timeout.ms** | [Опциональное] значение таймаута подключения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к OAUTHBEARER. | int | null | -- | низкая |
| **sasl.login.read.timeout.ms** | [Опциональное] значение таймаута чтения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к OAUTHBEARER. | int | null | -- | низкая |
| **sasl.login.refresh.buffer.seconds** | Время буферизации до истечения срока действия учётных данных, которое необходимо поддерживать при обновлении учётных данных, в секундах. Если в противном случае обновление произойдет ближе к истечению срока действия, чем количество буферных секунд, тогда обновление будет поднято вверх, чтобы сохранить как можно большую часть буферного времени. Допустимые значения: от **0** до **3600**(1 час). Если значение не указано, используется значение по умолчанию **300**(5 минут). Это значение и **sasl.login.refresh.min.period.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к OAUTHBEARER. | short | 300 | [0,...,3600] | низкая |
| **sasl.login.refresh.min.period.seconds** | Желаемое минимальное время ожидания потока обновления входа перед обновлением учётных данных в секундах. Допустимые значения: от **0** до **900**(15 минут). Если значение не указано, используется значение по умолчанию **60**(1 минута). Это значение и **sasl.login.refresh.buffer.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к OAUTHBEARER. | short | 60 | [0,...,900] | низкая |
| **sasl.login.refresh.window.factor** | Поток обновления входа в систему будет находиться в режиме ожидания до тех пор, пока не будет достигнут указанный коэффициент окна относительно срока действия учётных данных, после чего он попытается обновить учётные данные. Валидные значения находятся в пределах от**0.5**(50%) до **1.0** (100%) включительно. Если значение не указано, используется значение по умолчанию **0.8** (80%). В настоящее время применяется только к OAUTHBEARER. | double | 0.8 | [0.5,...,1.0] | низкая |
| **sasl.login.refresh.window.jitter** | Максимальное количество случайных джиттеров относительно срока действия учётных данных, которое добавляется ко времени сна потока обновления входа в систему. Допустимые значения находятся в диапазоне от **0** до **0.25** (25%) включительно. Если значение не указано, используется значение по умолчанию **0.05** (5%). В настоящее время применяется только к OAUTHBEARER. | double | 0.05 | [0.0,...,0.25] | низкая |
| **sasl.login.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к OAUTHBEARER. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.login.retry.backoff.ms** | [Опциональное] значение в миллисекундах для начального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к OAUTHBEARER. | long | 100 | -- | низкая |
| **sasl.oauthbearer.clock.skew.seconds** | [Опциональное] значение в секундах, позволяющее учитывать разницу между временем поставщика удостоверений OAuth/OIDC и брокера. | int | 30 | -- | низкая |
| **sasl.oauthbearer.expected.audience** | [Опциональный] параметр, разделённый запятыми, который брокер будет использовать для проверки того, что JWT был выпущен для одной из ожидаемых аудиторий. JJWT будет проверен на наличие стандартного утверждения OAuth **aud**, и если это значение установлено, брокер сопоставит значение из утверждения **aud**JWT, чтобы увидеть, есть ли точное совпадение. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | list | null | -- | низкая |
| **sasl.oauthbearer.expected.issuer** | [Опциональный] параметр, который брокер должен использовать для проверки того, что JWT был создан ожидаемым эмитентом. JWT будет проверен на наличие стандартного утверждения OAuth **iss**, и если это значение установлено, брокер будет точно сопоставлять его с тем, что находится в утверждении **iss**JWT. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.refresh.ms** | [Опциональное] значение в миллисекундах, в течение которого брокер должен ждать между обновлением своего кэша JWKS (набора веб-ключей JSON), содержащего ключи для проверки подписи JWT. | long | 3600000 (1 час) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками получить JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms** | [Опциональное] значение в миллисекундах для первоначального ожидания между попытками получения JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 100 | -- | низкая |
| **sasl.oauthbearer.scope.claim.name** | Утверждение OAuth для области часто называется **scope**, но этот [опциональный] параметр может предоставить другое имя для области, включённой в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | scope | -- | низкая |
| **sasl.oauthbearer.sub.claim.name** | Утверждение OAuth для субъекта часто называется sub, но этот [опциональный] параметр может предоставить другое имя для субъекта, включённого в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | sub | -- | низкая |
| **security.providers** | Список настраиваемых классов-создателей, каждый из которых возвращает поставщика, реализующего алгоритмы безопасности. Эти классы должны имплементировать интерфейс **org.apache.kafka.common.security.auth.SecurityProviderCreator**. | string | null | -- | низкая |
| **ssl.cipher.suites** | Список наборов шифров. Это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL. По умолчанию поддерживаются все доступные наборы шифров. | list | null | -- | низкая |
| **ssl.endpoint.identification.algorithm** | Алгоритм идентификации эндпоинта для валидации имени хоста сервера с использованием сертификата сервера. | string | https | -- | низкая |
| **ssl.engine.factory.class** | Класс типа **org.apache.kafka.common.security.auth.SslEngineFactory** для предоставления объектов SSLEngine. Значение по умолчанию — **org.apache.kafka.common.security.ssl.DefaultSslEngineFactory**. | class | null | -- | низкая |
| **ssl.keymanager.algorithm** | Алгоритм, используемый менеджером ключей для SSL-соединений. Значением по умолчанию является алгоритм менеджера ключей, настроенный для JVM. | string | SunX509 | -- | низкая |
| **ssl.secure.random.implementation** | Имплементация SecureRandom PRNG для использования в операциях шифрования SSL. | string | null | -- | низкая |
| **ssl.trustmanager.algorithm** | Алгоритм, используемый менеджером доверия для SSL-соединений. Значением по умолчанию является алгоритм менеджера доверия, настроенный для JVM. | string | PKIX | -- | низкая |

## 2.5 Конфигурации Kafka Connect

Ниже приведена конфигурация фреймворка Kafka Connect.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **config.storage.topic** | Имя топика RT.StreamingKafka, в котором хранятся конфигурации коннектора. | string | -- | -- | высокая |
| **group.id** | Уникальная строка, идентифицирующая группу кластера Connect, к которой принадлежит этот воркер. | string | -- | -- | высокая |
| **key.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат ключей в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | -- | -- | высокая |
| **offset.storage.topic** | Имя топика RT.StreamingKafka, в котором хранятся смещения коннектора источника. | string | -- | -- | высокая |
| **status.storage.topic** | Имя топика RT.StreamingKafka, в котором хранится коннектор и статус задачи. | string | -- | -- | высокая |
| **value.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат значений в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | -- | -- | высокая |
| **bootstrap.servers** | Список пар хост/порт, которые будут использоваться для установления инициирующего подключения к кластеру RT.StreamingKafka. Клиент будет использовать все серверы независимо от того, какие серверы указаны здесь для начальной загрузки — этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Этот список должен иметь форму **host1:port1,host2:port2,...**. Поскольку эти серверы используются только для инициирующего подключения для обнаружения полного членства в кластере (которое может меняться динамически), этот список не обязательно должен содержать полный набор серверов (хотя вам может понадобиться больше одного, если сервер не работает). | list | localhost:9092 | -- | высокая |
| **exactly.once.source.support** | Включить ли однократную поддержку коннекторов источников в кластере, используя транзакции для записи исходных записей и их исходных смещений, а также путём проактивного ограждения старых поколений задач перед запуском новых.  Чтобы включить однократную поддержку источника в новом кластере, установите для этого свойства значение **enabled**. Чтобы включить поддержку в существующем кластере, сначала установите значение **preparing**для каждого воркера в кластере, а затем установите значение **enabled**. Для обоих изменений можно использовать последовательное обновление. | string | disabled | [DISABLED, ENABLED, PREPARING] | высокая |
| **heartbeat.interval.ms** | Ожидаемое время между отправкой heartbeat-сообщений координатору группы при использовании средств управления группами RT.StreamingKafka. Heartbeat-сообщения используются для подтверждения активности сеанса воркера и для облегчения перебалансировки, когда новые участники присоединяются к группе или покидают её. Значение должно быть меньше, чем **session.timeout.ms**, но обычно его следует устанавливать не выше 1/3 этого значения. Его можно настроить еще ниже, чтобы контролировать ожидаемое время нормальной ребалансировки. | int | 3000 (3 секунды) | -- | высокая |
| **rebalance.timeout.ms** | Максимально допустимое время для каждого воркера, чтобы присоединиться к группе после начала ребалансировки. По сути, это ограничение на количество времени, необходимое всем задачам для очистки любых ожидающих данных и коммита смещений. Если таймаут будет превышен, воркер будет удалён из группы, что приведёт к сбоям коммита смещения. | int | 60000 (1 минута) | -- | высокая |
| **session.timeout.ms** | Таймаут, используемый для обнаружения сбоев воркеров. Воркер периодически отправляет heartbeat-сообщения, чтобы сообщить брокеру о своей работоспособности. Если до истечения таймаута этого сеанса брокер не получит никаких heartbeat-сообщений, то брокер удалит воркера из группы и инициирует ребалансировку. Обратите внимание, что значение должно находиться в допустимом диапазоне, настроенном в конфигурации брокера с помощью **group.min.session.timeout.ms** и **group.max.session.timeout.ms**. | int | 10000 (10 секунд) | -- | высокая |
| **ssl.key.password** | Пароль закрытого ключа в файле хранилища ключей или ключ PEM, указанный в **ssl.keystore.key**. | password | null | -- | высокая |
| **ssl.keystore.certificate.chain** | Цепочка сертификатов в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM со списком сертификатов X.509. | password | null | -- | высокая |
| **ssl.keystore.key** | Закрытый ключ в формате, указанном в **ssl.keystore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с ключами PKCS#8. Если ключ зашифрован, пароль ключа необходимо указать с помощью **ssl.key.password**. | password | null | -- | высокая |
| **ssl.keystore.location** | Расположение файла хранилища ключей. Это необязательно для клиента и может использоваться для двусторонней аутентификации клиента. | string | null | -- | высокая |
| **ssl.keystore.password** | Пароль хранилища для файла хранилища ключей. Это необязательно для клиента и необходимо только в том случае, если настроен **ssl.keystore.location**. Пароль хранилища ключей не поддерживается для формата PEM. | password | null | -- | высокая |
| **ssl.truststore.certificates** | Доверенные сертификаты в формате, указанном в параметре **ssl.truststore.type**. Механизм SSL по умолчанию поддерживает только формат PEM с сертификатами X.509. | password | null | -- | высокая |
| **ssl.truststore.location** | Расположение файла хранилища доверенных сертификатов. | string | null | -- | высокая |
| **ssl.truststore.password** | Пароль для файла хранилища доверенных сертификатов. Если пароль не установлен, настроенный файл хранилища доверенных сертификатов по-прежнему будет использоваться, но проверка целостности будет отключена. Пароль хранилища доверенных сертификатов не поддерживается для формата PEM. | password | null | -- | высокая |
| **client.dns.lookup** | Управляет тем, как клиент использует поиск DNS. Если установлено значение **use\_all\_dns\_ips**, подключается к каждому возвращаемому IP-адресу последовательно, пока не будет установлено успешное соединение. После отключения используется следующий IP. После того, как все IP-адреса были использованы один раз, клиент снова разрешает IP-адреса из имени хоста (однако поиск DNS-имен как в JVM, так и в кэше ОС). Если установлено значение **resolve\_canonical\_bootstrap\_servers\_only**, преобразуйте каждый адрес начальной загрузки в список канонических имён. После фазы начальной загрузки он ведёт себя так же, как **use\_all\_dns\_ips**. | string | use\_all\_dns\_ips | [use\_all\_dns\_ips, resolve\_canonical\_bootstrap\_servers\_only] | средняя |
| **connections.max.idle.ms** | Закрывает неактивные соединения по истечении количества миллисекунд, указанного в этой конфигурации. | long | 540000 (9 минут) | -- | средняя |
| **connector.client.config.override.policy** | Имя класса или алиас имплементации **ConnectorClientConfigOverridePolicy**. Определяет, какие конфигурации клиента могут быть переопределены коннектором. Имплементация по умолчанию — **all**, что означает, что конфигурации коннектора могут переопределять все свойства клиента. Другие возможные политики в платформе включают **none**, чтобы запретить коннекторам переопределять свойства клиента, и **principal**, чтобы разрешить коннекторам переопределять только принципалов клиента. | string | all | -- | средняя |
| **receive.buffer.bytes** | Размер приёмного буфера TCP (SO\_RCVBUF), который будет использоваться при чтении данных. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 32768 (32 кибибайт) | [0,...] | средняя |
| **request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. | int | 40000 (40 секунд) | [0,...] | средняя |
| **sasl.client.callback.handler.class** | Полное имя класса обработчика обратного вызова клиента SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. | class | null | -- | средняя |
| **sasl.jaas.config** | Параметры контекста входа в систему JAAS для соединений SASL в формате, используемом файлами конфигурации JAAS. Формат файла конфигурации JAAS описан [здесь](http://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html). Формат значения: **loginModuleClass controlFlag (optionName=optionValue)\*;**. Для брокеров конфигурация должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, требуется **listener.name.sasl\_ssl.scram-sha-256.sasl.jaas.config=com.example.ScramLoginModule required;** | password | null | -- | средняя |
| **sasl.kerberos.service.name** | Основное имя Kerberos, под которым работает RT.StreamingKafka. Это можно определить либо в конфигурации JAAS RT.StreamingKafka, либо в конфигурации RT.StreamingKafka. | string | null | -- | средняя |
| **sasl.login.callback.handler.class** | Полное имя класса обработчика обратного вызова для входа в систему SASL, который имплементирует интерфейс **AuthenticateCallbackHandler**. Для брокеров конфигурация обработчика обратного вызова при входе должна иметь префикс слушателя и имя механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler** | class | null | -- | средняя |
| **sasl.login.class** | Полное имя класса, имплементирующий интерфейс входа в систему. Для брокеров конфигурация входа должна начинаться с префикса слушателя и имени механизма SASL в нижнем регистре. Например, **listener.name.sasl\_ssl.scram-sha-256.sasl.login.class=com.example.CustomScramLogin**. | class | null | -- | средняя |
| **sasl.mechanism** | Механизм SASL, используемый для клиентских подключений. Это может быть любой механизм, для которого доступен поставщик безопасности. **GSSAPI**— механизм по умолчанию. | string | GSSAPI | -- | средняя |
| **sasl.oauthbearer.jwks.endpoint.url** | URL-адрес поставщика OAuth/OIDC, из которого можно получить [JWKS (набор веб-ключей JSON)](https://datatracker.ietf.org/doc/html/rfc7517#section-5) поставщика. URL-адрес может быть основан на HTTP(S) или на основе файла. Если URL-адрес основан на HTTP(S), данные JWKS будут получены от поставщика OAuth/OIDC через настроенный URL-адрес при запуске брокера. Все текущие ключи будут кэшироваться на брокере для входящих запросов. Если получен запрос аутентификации для JWT, который включает значение утверждения заголовка **kid**, которого ещё нет в кэше, эндпоинт JWKS будет запрошен снова по требованию. Однако брокер опрашивает URL-адрес каждые **sasl.oauthbearer.jwks.endpoint.refresh.ms** миллисекунды, чтобы обновить кэш любыми предстоящими ключами до того, как будут получены любые запросы JWT, которые их включают. Если URL-адрес основан на файле, брокер загрузит файл JWKS из настроенного местоположения при запуске. Если JWT включает значение заголовка **kid**, которого нет в файле JWKS, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | средняя |
| **sasl.oauthbearer.token.endpoint.url** | URL-адрес поставщика удостоверений OAuth/OIDC. Если URL-адрес основан на HTTP(S), это URL-адрес эндпоинта токена эмитента, к которому будут отправляться запросы на вход в систему на основе конфигурации в **sasl.jaas.config**. Если URL-адрес основан на файле, он указывает файл, содержащий токен доступа (в сериализованной форме JWT), выданный поставщиком удостоверений OAuth/OIDC для использования для авторизации. | string | null | -- | средняя |
| **security.protocol** | Протокол, используемый для связи с брокерами. Допустимые значения: **PLAINTEXT**, **SSL**, **SASL\_PLAINTEXT**, **SASL\_SSL**. | string | PLAINTEXT | [PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL] | средняя |
| **send.buffer.bytes** | Размер буфера отправки TCP (SO\_SNDBUF), который будет использоваться при отправке данных. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 131072 (128 кибибайт) | [0,...] | средняя |
| **ssl.enabled.protocols** | Список протоколов, включённых для SSL-соединений. По умолчанию используется значение **TLSv1.2,TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. При значении по умолчанию для Java 11 клиенты и серверы предпочтут **TLSv1.3**, если оба поддерживают его, и откажутся от **TLSv1.2** в противном случае (при условии, что оба поддерживают как минимум **TLSv1.2**). Это значение по умолчанию должно подойти для большинства случаев. Также смотри информацию по конфигурации для **ssl.protocol**. | list | TLSv1.2 | -- | средняя |
| **ssl.keystore.type** | Формат файла хранилища ключей. Параметр необязательно для клиента. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **ssl.protocol** | Протокол SSL, используемый для создания SSLContext. По умолчанию используется **TLSv1.3** при работе с Java 11 или более поздней версии, в противном случае — **TLSv1.2**. Это значение должно подойти для большинства случаев использования. Разрешёнными значениями в последних JVM являются **TLSv1.2** и **TLSv1.3**. **TLS**, **TLSv1.1**, **SSL**, **SSLv2**и **SSLv3**могут поддерживаться в старых JVM, но их использование не рекомендуется из-за известных уязвимостей безопасности. При значении по умолчанию для этой конфигурации и **ssl.enabled.protocols** клиенты перейдут на **TLSv1.2**, если сервер не поддерживает **TLSv1.3**. Если для этой конфигурации установлено значение **TLSv1.2**, клиенты не будут использовать **TLSv1.3**, даже если это одно из значений в **ssl.enabled.protocols**, а сервер поддерживает только **TLSv1.3**. | string | TLSv1.2 | -- | средняя |
| **ssl.provider** | Имя поставщика безопасности, используемого для SSL-соединений. Значение по умолчанию — поставщик безопасности по умолчанию для JVM. | string | null | -- | средняя |
| **ssl.truststore.type** | Формат файла хранилища доверенных сертификатов. Значения, поддерживаемые в настоящее время по умолчанию **ssl.engine.factory.class**: **[JKS, PKCS12, PEM]**. | string | JKS | -- | средняя |
| **worker.sync.timeout.ms** | Если воркер не синхронизирован с другими воркерами и ему необходимо повторно синхронизировать конфигурации, он будет ждать определённое время, прежде чем отказать, покинуть группу и ждать периода отсрочки перед повторным присоединением. | int | 3000 (3 секунды) | -- | средняя |
| **worker.unsync.backoff.ms** | Если воркер не синхронизирован с другими воркерами и не может их догнать в рамках **worker.sync.timeout.ms**, он покинет кластер Connect на это время, прежде чем снова присоединиться. | int | 300000 (5 минут) | -- | средняя |
| **access.control.allow.methods** | Устанавливает методы, поддерживаемые для запросов между источниками, путём установки заголовка **Access-Control-Allow-Methods**. Значение по умолчанию заголовка **Access-Control-Allow-Methods** разрешает запросы перекрестного происхождения для **GET**, **POST**и **HEAD**. | string | "" | -- | низкая |
| **access.control.allow.origin** | Значение для установки заголовка **Access-Control-Allow-Origin** для запросов REST API. Чтобы включить доступ к перекрёстному источнику, установите для него домен приложения, которому должен быть разрешён доступ к API, или **\***, чтобы разрешить доступ из любого домена. Значение по умолчанию разрешает доступ только из домена REST API. | string | "" | -- | низкая |
| **admin.listeners** | Список URI, разделённых запятыми, которые будет прослушивать API REST администратора. Поддерживаемые протоколы: HTTP и HTTPS. Пустая строка (blank или emplty) отключит эту функцию. Поведение по умолчанию — использование обычного слушателя (заданного свойством **listeners**). | list | null | Список URL-адресов, разделённых запятыми, например: http://localhost:8080,https://localhost:8443 | низкая |
| **auto.include.jmx.reporter** | [**Устарело**] Следует ли автоматически включать JmxReporter, даже если он не указан в **metric.reporters**. Эта конфигурация будет удалена в Kafka 4.0, вместо этого пользователям следует включить **org.apache.kafka.common.metrics.JmxReporter** в **metric.reporters**, чтобы включить JmxReporter. | boolean | true | -- | низкая |
| **client.id** | Строка идентификатора, передаваемая на сервер при выполнении запросов. Целью параметра является возможность отслеживать источник запросов, помимо IP/порта, позволяя включать логическое имя приложения в лог запросов на стороне сервера. | string | "" | -- | низкая |
| **config.providers** | Имена классов **ConfigProvider**, разделённые запятыми, загружаемые и используемые в указанном порядке. Имплементация интерфейса **ConfigProvider**позволяет заменять ссылки на переменные в конфигурациях коннектора, например, для внешних секретов. | list | "" | -- | низкая |
| **config.storage.replication.factor** | Коэффициент репликации, используемый при создании конфигурации топика хранилища. Значение должно быть положительным числом, не превышающим количество брокеров в кластере RT.StreamingKafka, или **-1**, чтобы использовать значение брокера по умолчанию. | short | 3 | -- | низкая |
| **connect.protocol** | Режим совместимости для протокола RT.StreamingKafka Connect. | string | sessioned | [eager, compatible, sessioned] | низкая |
| **header.converter** | Класс **HeaderConverter**, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Это контролирует формат значений заголовков в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку это не зависит от коннекторов, позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. По умолчанию **SimpleHeaderConverter**используется для сериализации значений заголовков в строки и их десериализации путём вывода схем. | class | org.apache.kafka.connect.storage.SimpleHeaderConverter | -- | низкая |
| **inter.worker.key.generation.algorithm** | Алгоритм, используемый для генерации внутренних ключей запроса. Алгоритм **HmacSHA256**будет использоваться по умолчанию на JVM, которые его поддерживают; в других JVM значение по умолчанию не используется, и значение этого свойства необходимо вручную указать в конфигурации воркера. | string | HmacSHA256 | Любой алгоритм KeyGenerator, поддерживаемый воркером JVM | низкая |
| **inter.worker.key.size** | Размер ключа, используемого для подписи внутренних запросов, в битах. Если значение равно нулю, будет использоваться размер ключа по умолчанию для алгоритма генерации ключей. | int | null | -- | низкая |
| **inter.worker.key.ttl.ms** | Срок жизни сгенерированных сеансовых ключей, используемых для проверки внутреннего запроса (в миллисекундах). | int | 3600000 (1 час) | [0,...,2147483647] | низкая |
| **inter.worker.signature.algorithm** | Алгоритм, используемый для подписи внутренних запросов. Алгоритм **inter.worker.signature.algorithm** будет использоваться по умолчанию на JVM, которые его поддерживают. В других JVM значение по умолчанию не используется, и значение этого свойства необходимо вручную указать в конфигурации воркера. | string | HmacSHA256 | Любой алгоритм MAC, поддерживаемый рабочей JVM | низкая |
| **inter.worker.verification.algorithms** | Список разрешённых алгоритмов проверки внутренних запросов, который должен включать алгоритм, используемый для свойства **inter.worker.signature.algorithm**. Алгоритмы**[HmacSHA256]** будут использоваться по умолчанию на JVM, которые их предоставляют. В других JVM значение по умолчанию не используется, и значение этого свойства необходимо вручную указать в конфигурации воркера. | list | HmacSHA256 | Список одного или нескольких алгоритмов MAC, каждый из которых поддерживается воркером JVM. | низкая |
| **listeners** | Список URI, разделённых запятыми, которые будет прослушивать REST API. Поддерживаемые протоколы: HTTP и HTTPS.  Укажите имя хоста как **0.0.0.0** для привязки ко всем интерфейсам.  Оставьте имя хоста пустым, чтобы привязать его к интерфейсу по умолчанию.  Примеры валидных списков слушателей: **HTTP://myhost:8083,HTTPS://myhost:8084**. | list | http://:8083 | Список URL-адресов, разделённых запятыми, например: http://localhost:8080,https://localhost:8443. | низкая |
| **metadata.max.age.ms** | Период времени в миллисекундах, по истечении которого мы принудительно обновляем метаданные, даже если мы не заметили никаких изменений лидерства партиции, чтобы заранее обнаружить любые новые брокеры или партиции. | long | 300000 (5 минут) | [0,...] | низкая |
| **metric.reporters** | Список классов для использования в качестве генераторов отчётов по метрикам. Имплементация интерфейса **org.apache.kafka.common.metrics.MetricsReporter** позволяет подключать классы, которые будут уведомляться о создании новой метрики. JmxReporter всегда включён для регистрации статистики JMX. | list | "" | -- | низкая |
| **metrics.num.samples** | Количество выборок, поддерживаемых для вычисления метрик. | int | 2 | [1,...] | низкая |
| **metrics.recording.level** | Самый высокий уровень записи метрик. | string | INFO | [INFO, DEBUG] | низкая |
| **metrics.sample.window.ms** | Окно времени, в течение которого вычисляется выборка метрик. | long | 30000 (30 секунд) | [0,...] | низкая |
| **offset.flush.interval.ms** | Интервал, с которым можно попытаться закоммитить смещения для задач. | long | 60000 (1 минута) | -- | низкая |
| **offset.flush.timeout.ms** | Максимальное количество миллисекунд ожидания очистки записей и коммита данных смещения партиции в хранилище смещения перед отменой процесса и восстановлением данных смещения, которые будут закоммичены при следующей попытке. Это свойство не влияет на коннекторы источников, работающие с однократной поддержкой. | long | 5000 (5 секунд) | -- | низкая |
| **offset.storage.partitions** | Количество партиций, используемых при создании топика смещённого хранилища. | int | 25 | Положительное число или **-1**, чтобы использовать значение брокера по умолчанию. | низкая |
| **offset.storage.replication.factor** | Коэффициент репликации, используемый при создании топика смещённого хранилища. | short | 3 | Положительное число, не превышающее количество брокеров в кластере RT.StreamingKafka, или **-1**, чтобы использовать значение брокера по умолчанию. | низкая |
| **plugin.path** | Список разделённых запятыми (,) путей, содержащих плагины (коннекторы, конвертеры, трансформаторы). Список должен состоять из каталогов верхнего уровня, которые включают любую комбинацию:   * каталоги, непосредственно содержащие jar-файлы с плагинами и их зависимостями. * uber-jars с плагинами и их зависимостями. * каталоги, непосредственно содержащие структуру каталогов пакетов классов плагинов и их зависимостей.   **Примечание**. Для обнаружения зависимостей или плагинов будут использоваться симлинки.  Примеры: **plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors**  Не используйте переменные поставщика конфигурации в этом свойстве, поскольку необработанный путь используется сканером воркера до инициализации поставщиков конфигурации и использования для замены переменных. | list | null | -- | низкая |
| **reconnect.backoff.max.ms** | Максимальное время ожидания в миллисекундах при повторном подключении к брокеру, к которому неоднократно не удавалось подключиться. Если параметр предусмотрен, отсрочка на хост будет увеличиваться экспоненциально для каждого последующего сбоя соединения, вплоть до этого максимума. После расчёта увеличения задержки добавляется 20% случайного джиттера, чтобы избежать большого количества одновременных соединений. | long | 1000 (1 секунда) | [0,...] | низкая |
| **reconnect.backoff.ms** | Базовое время ожидания перед попыткой повторного подключения к данному хосту. Параметр позволяет избежать повторных подключений к хосту в плотном цикле. Эта отсрочка применяется ко всем попыткам подключения клиента к брокеру. | long | 50 | [0,...] | низкая |
| **response.http.headers.config** | Правила для заголовков HTTP-ответов REST API. | string | "" | Правила заголовков, разделённые запятыми, где каждое правило заголовка имеет форму **'[action] [header name]:[header value]'** и может быть заключено в двойные кавычки, если какая-либо часть правила заголовка содержит запятую | низкая |
| **rest.advertised.host.name** | Если этот параметр установлен, это имя хоста, которое будет передаваться другим воркерам для подключения. | string | null | -- | низкая |
| **rest.advertised.listener** | Устанавливает объявленного слушателя (HTTP или HTTPS), который будет предоставлен для использования другим воркерам. | string | null | -- | низкая |
| **rest.advertised.port** | Если этот параметр установлен, это порт, который будет предоставлен другим воркерам для подключения. | int | null | -- | низкая |
| **rest.extension.classes** | Имена классов **ConnectRestExtension**, разделённые запятыми, загружаемые и вызываемые в указанном порядке. Имплементация интерфейса **ConnectRestExtension**позволяет внедрять в Connect определяемые пользователем ресурсы REST API, такие как фильтры. Обычно используется для добавления кастомных возможностей, таких как ведение лога, безопасность и т.д. | list | "" | -- | низкая |
| **retry.backoff.ms** | Время ожидания перед попыткой повторения неудачного запроса к данной партиции топика. Параметр позволяет избежать повторной отправки запросов в плотном цикле в некоторых сценариях сбоя. | long | 100 | [0,...] | низкая |
| **sasl.kerberos.kinit.cmd** | Путь к команде Kerberos **kinit**. | string | /usr/bin/kinit | -- | низкая |
| **sasl.kerberos.min.time.before.relogin** | Время ожидания потока входа между попытками обновления. | long | 60000 | -- | низкая |
| **sasl.kerberos.ticket.renew.jitter** | Процент случайного джиттера, добавленного ко времени обновления. | double | 0.05 | -- | низкая |
| **sasl.kerberos.ticket.renew.window.factor** | Поток входа будет находиться в спящем режиме до тех пор, пока не будет достигнут указанный временной интервал от последнего обновления до истечения срока действия билета, после чего он попытается продлить билет. | double | 0.8 | -- | низкая |
| **sasl.login.connect.timeout.ms** | [Опциональное] значение таймаута подключения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к OAUTHBEARER. | int | null | -- | низкая |
| **sasl.login.read.timeout.ms** | [Опциональное] значение таймаута чтения внешнего поставщика аутентификации в миллисекундах. В настоящее время применяется только к OAUTHBEARER. | int | null | -- | низкая |
| **sasl.login.refresh.buffer.seconds** | Время буферизации до истечения срока действия учётных данных, которое необходимо поддерживать при обновлении учётных данных, в секундах. Если в противном случае обновление произойдет ближе к истечению срока действия, чем количество буферных секунд, тогда обновление будет перенесено вверх, чтобы сохранить как можно большую часть буферного времени. Допустимые значения: от **0** до **3600**(1 час). Если значение не указано, используется значение по умолчанию **300**(5 минут). Это значение и **sasl.login.refresh.min.period.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к OAUTHBEARER. | short | 300 | [0,...,3600] | низкая |
| **sasl.login.refresh.min.period.seconds** | Желаемое минимальное время ожидания потока обновления входа перед обновлением учётных данных в секундах. Допустимые значения: от **0** до **900**(15 минут). Если значение не указано, используется значение по умолчанию **60**(1 минута). Это значение и **sasl.login.refresh.buffer.seconds** игнорируются, если их сумма превышает оставшийся срок действия учётных данных. В настоящее время применяется только к OAUTHBEARER. | short | 60 | [0,...,900] | низкая |
| **sasl.login.refresh.window.factor** | Поток обновления входа в систему будет находиться в режиме ожидания до тех пор, пока не будет достигнут указанный коэффициент окна относительно срока действия учётных данных, после чего он попытается обновить учётные данные. Валидные значения находятся в пределах от **0.5** (50%) до **1.0** (100%) включительно. Если значение не указано, используется значение по умолчанию **0.8** (80%). В настоящее время применяется только к OAUTHBEARER. | double | 0.8 | [0.5,...,1.0] | низкая |
| **sasl.login.refresh.window.jitter** | Максимальное количество джиттера относительно срока действия учётных данных, которое добавляется ко времени сна потока обновления входа в систему. Допустимые значения находятся в диапазоне от **0** до **0.25** (25%) включительно. Если значение не указано, используется значение по умолчанию **0.05** (5%). В настоящее время применяется только к OAUTHBEARER. | double | 0.05 | [0.0,...,0.25] | низкая |
| **sasl.login.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к OAUTHBEARER. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.login.retry.backoff.ms** | [Опциональное] значение в миллисекундах для начального ожидания между попытками входа в систему внешнего поставщика аутентификации. Вход использует экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.login.retry.backoff.ms**, и удваивает продолжительность ожидания между попытками до максимальной длины ожидания, указанной в **sasl.login.retry.backoff.max.ms**. В настоящее время применяется только к OAUTHBEARER. | long | 100 | -- | низкая |
| **sasl.oauthbearer.clock.skew.seconds** | [Опциональное] значение в секундах, позволяющее учитывать разницу между временем поставщика удостоверений OAuth/OIDC и брокера. | int | 30 | -- | низкая |
| **sasl.oauthbearer.expected.audience** | [Опциональный] параметр, разделённый запятыми, который брокер будет использовать для проверки того, что JWT был выпущен для одной из ожидаемых аудиторий. JWT будет проверен на наличие стандартного утверждения OAuth **aud**, и если это значение установлено, брокер сопоставит значение из утверждения **aud**JWT, чтобы увидеть, есть ли точное совпадение. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | list | null | -- | низкая |
| **sasl.oauthbearer.expected.issuer** | [Опциональный] параметр, который брокер должен использовать для проверки того, что JWT был создан ожидаемым эмитентом. JWT будет проверен на наличие стандартного утверждения OAuth **iss**, и если это значение установлено, брокер будет точно сопоставлять его с тем, что находится в утверждении **iss**JWT. Если совпадений нет, брокер отклонит JWT, и аутентификация завершится неудачно. | string | null | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.refresh.ms** | [Опциональное] значение в миллисекундах, в течение которого брокер должен ждать между обновлением своего кэша JWKS (набора веб-ключей JSON), содержащего ключи для проверки подписи JWT. | long | 3600000 (1 час) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms** | [Опциональное] значение в миллисекундах для максимального ожидания между попытками получить JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 10000 (10 секунд) | -- | низкая |
| **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms** | [Опциональное] значение в миллисекундах для первоначального ожидания между попытками получения JWKS (набор веб-ключей JSON) от внешнего поставщика аутентификации. При извлечении JWKS используется экспоненциальный алгоритм отсрочки с начальным ожиданием, основанным на настройке **sasl.oauthbearer.jwks.endpoint.retry.backoff.ms**, и длина ожидания между попытками удваивается до максимальной длины ожидания, указанной в **sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms**. | long | 100 | -- | низкая |
| **sasl.oauthbearer.scope.claim.name** | Утверждение OAuth для области часто называется **scope**, но этот [опциональный] параметр может предоставить другое имя для области, включённой в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | scope | -- | низкая |
| **sasl.oauthbearer.sub.claim.name** | Утверждение OAuth для субъекта часто называется **sub**, но этот [опциональный] параметр может предоставить другое имя для субъекта, включённого в утверждения полезных данных JWT, если поставщик OAuth/OIDC использует другое имя для этого утверждения. | string | sub | -- | низкая |
| **scheduled.rebalance.max.delay.ms** | Максимальная задержка, которая запланирована для ожидания возвращения одного или нескольких отвалившихся воркеров перед повторной балансировкой и переназначением их коннекторов и задач группе. В этот период коннекторы и задачи отвалившихся воркеров остаются нераспределёнными. | int | 300000 (5 минут) | [0,...,2147483647] | низкая |
| **socket.connection.setup.timeout.max.ms** | Максимальное время, в течение которого клиент будет ожидать установления соединения с сокетом. Таймаут установки соединения будет увеличиваться экспоненциально для каждого последующего сбоя соединения вплоть до этого максимума. Чтобы избежать большого количества одновременных соединений, к таймауту будет применён коэффициент рандомизации **0.2**, что приведёт к случайному диапазону между 20% ниже и 20% выше вычисленного значения. | long | 30000 (30 секунд) | [0,...] | низкая |
| **socket.connection.setup.timeout.ms** | Время, в течение которого клиент будет ожидать установления соединения с сокетом. Если соединение не будет установлено до истечения времени ожидания, клиенты закроют канал сокета. | long | 10000 (10 секунд) | [0,...] | низкая |
| **ssl.cipher.suites** | Список наборов шифров. Это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL. По умолчанию поддерживаются все доступные наборы шифров. | list | null | -- | низкая |
| **ssl.client.auth** | Настраивает брокера RT.StreamingKafka для запроса аутентификации клиента. Следующие настройки являются общими:   * **ssl.client.auth=required** – если установлено значение **required**, требуется аутентификация клиента. * **ssl.client.auth=requested** – это означает, что аутентификация клиента необязательна. В отличие от **required**, если эта опция установлена, клиент может отказаться предоставлять аутентификационную информацию о себе. * **ssl.client.auth=none** – это означает, что аутентификация клиента не требуется. | string | none | [required, requested, none] | низкая |
| **ssl.endpoint.identification.algorithm** | Алгоритм идентификации эндпоинта для валидации имени хоста сервера с использованием сертификата сервера. | string | https | -- | низкая |
| **ssl.engine.factory.class** | Класс типа **org.apache.kafka.common.security.auth.SslEngineFactory** для предоставления объектов SSLEngine. Значение по умолчанию — **org.apache.kafka.common.security.ssl.DefaultSslEngineFactory**. | class | null | -- | низкая |
| **ssl.keymanager.algorithm** | Алгоритм, используемый менеджером ключей для SSL-соединений. Значением по умолчанию является алгоритм менеджера ключей, настроенный для JVM. | string | SunX509 | -- | низкая |
| **ssl.secure.random.implementation** | Имплементация SecureRandom PRNG для использования в операциях шифрования SSL. | string | null | -- | низкая |
| **ssl.trustmanager.algorithm** | Алгоритм, используемый менеджером доверия для SSL-соединений. Значением по умолчанию является алгоритм менеджера доверия, настроенный для JVM. | string | PKIX | -- | низкая |
| **status.storage.partitions** | Количество партиций, используемых при создании топика хранения статусов. | int | 5 | Положительное число или **-1**, чтобы использовать значение брокера по умолчанию | низкая |
| **status.storage.replication.factor** | Коэффициент репликации, используемый при создании топика хранения статусов. | short | 3 | Положительное число, не превышающее количество брокеров в кластере RT.StreamingKafka, или **-1**, чтобы использовать значение брокера по умолчанию. | низкая |
| **task.shutdown.graceful.timeout.ms** | Время ожидания корректного завершения работы задач. Это общее количество времени, а не на каждую задачу. У всех задач сработало завершение работы, после этого задачи последовательно ожидают завершения. | long | 5000 (5 секунд) | -- | низкая |
| **topic.creation.enable** | Разрешить ли автоматическое создание топиков, используемых коннекторами источников, если коннекторы источников настроены со свойствами **topic.creation.**. Каждая задача будет использовать клиент администратора для создания своих топиков и не будет зависеть от автоматического создания топиков брокерами RT.StreamingKafka. | boolean | true | -- | низкая |
| **topic.tracking.allow.reset** | Если установлено значение **true**, это позволяет запросам пользователей сбросить набор активных топиков для каждого коннектора. | boolean | true | -- | низкая |
| **topic.tracking.enable** | Включает отслеживание набора активных топиков для каждого коннектора во время выполнения. | boolean | true | -- | низкая |

### 2.5.1 Конфигурации коннектора источника

Ниже приведена конфигурация коннектора источника.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **name** | Глобальное уникальное имя для этого коннектора. | string | -- | Непустая строка без управляющих символов ISO | высокая |
| **connector.class** | Имя или алиас класса для этого коннектора. Должен быть подклассом **org.apache.kafka.connect.connector.Connector**. Если коннектор — **org.apache.kafka.connect.file.FileStreamSinkConnector**, вы можете либо указать это полное имя, либо использовать **FileStreamSink**или **FileStreamSinkConnector**, чтобы сделать конфигурацию немного короче. | string | -- | -- | высокая |
| **tasks.max** | Максимальное количество задач, которые можно использовать для этого коннектора. | int | 1 | [1,...] | высокая |
| **key.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат ключей в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | null | -- | низкая |
| **value.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат значений в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | null | -- | низкая |
| **header.converter** | Класс **HeaderConverter**, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат значений заголовков в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку это не зависит от коннекторов, позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. По умолчанию **SimpleHeaderConverter**используется для сериализации значений заголовков в строки и их десериализации путём вывода схем. | class | null | -- | низкая |
| **config.action.reload** | Действие, которое Connect должен выполнить над коннектором, когда изменения в конфигурациях внешних поставщиков приводят к изменению свойств конфигурации коннектора. Значение **none**указывает, что Connect ничего не будет делать. Значение **restart**указывает, что Connect должен перезапустить/перезагрузить коннектор с обновлёнными свойствами конфигурации. Фактически перезапуск может быть запланирован на будущее, если внешний поставщик конфигурации укажет, что срок действия значения конфигурации истечёт в будущем. | string | restart | [none, restart] | низкая |
| **transforms** | Алиасы трансформаций, которые будут применяться к записям. | list | "" | Ненулевая строка, уникальные алиасы трансформаций | низкая |
| **predicates** | Алиасы предикатов, используемых трансформациями. | list | "" | Ненулевая строка, уникальные алиасы предикатов | низкая |
| **errors.retry.timeout** | Максимальная продолжительность в миллисекундах, в течение которой будет повторяться неудачная операция. По умолчанию установлено значение **0**, что означает, что повторные попытки выполняться не будут. Используйте **-1** для бесконечных повторов. | long | 0 | -- | средняя |
| **errors.retry.delay.max.ms** | Максимальная продолжительность в миллисекундах между последовательными повторными попытками. Джиттер будет добавлен к задержке, как только будет достигнут этот предел, чтобы предотвратить с большим количеством одновременных соединений. | long | 60000 (1 минута) | -- | средняя |
| **errors.tolerance** | Поведение для допуска ошибок во время работы коннектора. **none**— это значение по умолчанию, которое сигнализирует о том, что любая ошибка приведёт к немедленному сбою задачи коннектора; **all**изменяет поведение, пропуская проблемные записи. | string | none | [none, all] | средняя |
| **errors.log.enable** | Если установлено **true**, записывается каждая ошибка, а также сведения о неудачной операции и проблемной записи в лог приложения Connect. По умолчанию установлено значение **false**, при котором сообщается только о недопустимых ошибках. | boolean | false | -- | средняя |
| **errors.log.include.messages** | Включать ли в лог запись Connect, которая привела к сбою. Для записей приёмника будут регистрироваться топик, партиция, смещение и временная метка. Для исходных записей будут регистрироваться ключ и значение (и их схемы), все заголовки, а также временная метка, топик RT.StreamingKafka, партиция RT.StreamingKafka, исходная партиция и смещение источника. По умолчанию установлено значение **false**, что предотвращает запись ключей, значений и заголовков записей в файлы логов. | boolean | false | -- | средняя |
| **topic.creation.groups** | Группы конфигураций для топиков, созданных исходными коннекторами. | list | "" | Ненулевая строка, уникальные группы создания топиков | низкая |
| **exactly.once.support** | Разрешённые значения запрошены, обязательны. Если установлено значение **required**, принудительно выполняется предварительная проверка коннектора, чтобы убедиться, что он может предоставить однократную семантику с заданной конфигурацией. Некоторые коннекторы могут обеспечивать одноразовую семантику, но не сигнализировать Connect о том, что они это поддерживают; в этом случае перед его созданием следует внимательно ознакомиться с документацией коннектора и установить для этого свойства значение **requested**. Кроме того, если для параметра установлено значение **required**, но у исполнителя, выполняющего предварительную проверку, не включена однократная поддержка для исходных коннекторов, запросы на создание или проверку коннектора завершатся ошибкой. | string | requested | [REQUIRED, REQUESTED] | средняя |
| **transaction.boundary** | Допустимые значения: **poll**, **interval**, **connector**. Если установлено значение **poll**, новая транзакция поставщика будет запущена и закоммичена для каждого пакета записей, которые каждая задача из этого коннектора предоставляет Connect. Если установлено значение **connector**, полагается на границы транзакции, определённые коннектором. Обратите внимание, что не все коннекторы способны определять свои собственные границы транзакций, и в этом случае попытки создать инстанс коннектора с этим значением завершится неудачей. Наконец, если установлено значение **interval**, транзакции коммитится только по истечении заданного пользователем интервала времени. | string | poll | [INTERVAL, POLL, CONNECTOR] | средняя |
| **transaction.boundary.interval.ms** | Если для параметра **transaction.boundary** установлено значение **interval**, определяет интервал коммита транзакции поставщика задачами коннектора. Если значение не установлено, по умолчанию используется значение свойства **offset.flush.interval.ms** на уровне воркера. Параметр не имеет никакого эффекта, если указан другой параметр **transaction.boundary**. | long | null | [0,...] | низкая |
| **offsets.storage.topic** | Имя отдельного топика смещений, который будет использоваться для этого коннектора. Если параметр пустой или не указан, будет использоваться имя топика глобальных смещений воркера. Если параметр указан, будет создан топик смещений, если он ещё не существует в кластере RT.StreamingKafka, на который нацелен этот коннектор (который может отличаться от топика, используемого для топика глобальных смещений воркера, если свойство **bootstrap.servers** поставщика коннектора было переопределено из воркера). Применимо только в распределённом режиме. В автономном режиме установка этого свойства не будет иметь никакого эффекта. | string | null | Непустая строка | низкая |

### 2.5.2 Конфигурации sink-коннектора

Ниже представлена конфигурация sink-коннектора.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **name** | Глобальное уникальное имя для этого коннектора. | string | -- | непустая строка без управляющих символов ISO | высокая |
| **connector.class** | Имя или алиас класса для этого коннектора. Должен быть подклассом **org.apache.kafka.connect.connector.Connector**. Если коннектор — **org.apache.kafka.connect.file.FileStreamSinkConnector**, вы можете либо указать это полное имя, либо использовать **FileStreamSink**или **FileStreamSinkConnector**, чтобы сделать конфигурацию немного короче. | string | -- | -- | высокая |
| **tasks.max** | Максимальное количество задач, которые можно использовать для этого коннектора. | int | 1 | [1,...] | высокая |
| **topics** | Список топиков для потребления, разделённых запятыми. | list | "" | -- | высокая |
| **topics.regex** | Регулярное выражение, дающее топики для потребления. Внутри регулярное выражение компилируется в **java.util.regex.Pattern**. Должен быть указан только один из параметров: **topics**или **topics.regex**. | string | "" | валидное регулярное выражение | высокая |
| **key.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат ключей в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | null | -- | низкая |
| **value.converter** | Класс конвертера, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат значений в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку он не зависит от коннекторов, он позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. | class | null | -- | низкая |
| **header.converter** | Класс **HeaderConverter**, используемый для конвертации формата RT.StreamingKafka Connect в сериализованную форму, записанную в RT.StreamingKafka. Параметр контролирует формат значений заголовков в сообщениях, записываемых или читаемых из RT.StreamingKafka, и, поскольку это не зависит от коннекторов, позволяет любому коннектору работать с любым форматом сериализации. Примеры распространённых форматов включают JSON и Avro. По умолчанию **SimpleHeaderConverter**используется для сериализации значений заголовков в строки и их десериализации путём вывода схем. | class | null | -- | низкая |
| **config.action.reload** | Действие, которое Connect должен выполнить над коннектором, когда изменения в конфигурациях внешних поставщиков приводят к изменению свойств конфигурации коннектора. Значение **none**указывает, что Connect ничего не будет делать. Значение **restart**указывает, что Connect должен перезапустить/перезагрузить коннектор с обновлёнными свойствами конфигурации. Фактически перезапуск может быть запланирован на будущее, если внешний поставщик конфигурации укажет, что срок действия значения конфигурации истечёт в будущем. | string | restart | [none, restart] | низкая |
| **transforms** | Алиасы трансформаций, которые будут применяться к записям. | list | "" | Ненулевая строка, уникальные алиасы трансформаций | низкая |
| **predicates** | Алиасы предикатов, используемых трансформациями. | list | "" | Ненулевая строка, уникальные алиасы предикатов | низкая |
| **errors.retry.timeout** | Максимальная продолжительность в миллисекундах, в течение которой будет повторяться неудачная операция. По умолчанию установлено значение **0**, что означает, что повторные попытки выполняться не будут. Используйте **-1** для бесконечных повторов. | long | 0 | -- | средняя |
| **errors.retry.delay.max.ms** | Максимальная продолжительность в миллисекундах между последовательными повторными попытками. Джиттер будет добавлен к задержке, как только будет достигнут этот предел, чтобы предотвратить с большим количеством одновременных соединений. | long | 60000 (1 минута) | -- | средняя |
| **errors.tolerance** | Поведение для допуска ошибок во время работы коннектора. **none**— это значение по умолчанию, которое сигнализирует о том, что любая ошибка приведёт к немедленному сбою задачи коннектора; **all**изменяет поведение, пропуская проблемные записи. | string | none | [none, all] | средняя |
| **errors.log.enable** | Если установлено **true**, записывается каждая ошибка, а также сведения о неудачной операции и проблемной записи в лог приложения Connect. По умолчанию установлено значение **false**, при котором сообщается только о недопустимых ошибках. | boolean | false | -- | средняя |
| **errors.log.include.messages** | Включать ли в лог запись Connect, которая привела к сбою. Для записей приёмника будут регистрироваться топик, партиция, смещение и временная метка. Для исходных записей будут регистрироваться ключ и значение (и их схемы), все заголовки, а также временная метка, топик RT.StreamingKafka, партиция RT.StreamingKafka, исходная партиция и смещение источника. По умолчанию установлено значение **false**, что предотвращает запись ключей, значений и заголовков записей в файлы логов. | boolean | false | -- | средняя |
| **errors.deadletterqueue.topic.name** | Имя топика, который будет использоваться в качестве очереди недоставленных писем (dead letter queue, DLQ) для сообщений, которые приводят к ошибке при обработке этим sink-коннектором, его трансформациями или конвертациями. По умолчанию имя топика пустое (blank), что означает, что никакие сообщения не должны записываться в DLQ. | string | "" | -- | средняя |
| **errors.deadletterqueue.topic.replication.factor** | Коэффициент репликации, используемый для создания топика очереди недоставленных писем, когда он ещё не существует. | short | 3 | -- | средняя |
| **errors.deadletterqueue.context.headers.enable** | Если установлено **true**, добавьте заголовки, содержащие контекст ошибки, к сообщениям, записываемым в очередь недоставленных писем. Чтобы избежать конфликта с заголовками исходной записи, все ключи заголовков контекста ошибки, все ключи заголовков контекста ошибки будут начинаться с **\_\_connect.errors.**. | boolean | false | -- | средняя |

## 2.6 Конфигурации Kafka Streams

Ниже приведена конфигурация клиентской библиотеки Kafka Streams.

| **Параметр** | **Описание** | **Тип** | **Значение по умолчанию** | **Допустимые значения** | **Важность** |
| --- | --- | --- | --- | --- | --- |
| **application.id** | Идентификатор приложения обработки стримов. Он должно быть уникальным в кластере RT.StreamingKafka. Он используется как:   * префикс идентификатора клиента по умолчанию. * идентификатор группы для управления членством. * префикс топика лога изменений. | string | -- | -- | высокая |
| **bootstrap.servers** | Список пар хост/порт, которые будут использоваться для установления инициирующего подключения к кластеру RT.StreamingKafka. Клиент будет использовать все серверы независимо от того, какие серверы указаны здесь для начальной загрузки — этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Этот список должен иметь форму **host1:port1,host2:port2,...**. Поскольку эти серверы используются только для инициирующего подключения для обнаружения полного членства в кластере (которое может меняться динамически), этот список не обязательно должен содержать полный набор серверов (хотя вам может понадобиться больше одного, если сервер не работает). | list | -- | -- | высокая |
| **num.standby.replicas** | Количество резервных реплик для каждой задачи. | int | 0 | -- | высокая |
| **state.dir** | Расположение каталога для хранилища состояний. Этот путь должен быть уникальным для каждого инстанса стримов, использующего одну и ту же базовую файловую систему. | string | /var/folders/0t/68svdzmx1sld0 mxjl8dgmmzm0000gq/T//kafka-streams | -- | высокая |
| **acceptable.recovery.lag** | Максимально допустимая задержка (количество смещений, которые необходимо догнать), чтобы клиент считался догоняющим, достаточным для получения активного назначения задачи. После назначения он всё равно восстановит остальную часть лога изменений перед обработкой. Чтобы избежать паузы в обработке во время ребалансировки, эта конфигурация должна соответствовать времени восстановления менее минуты для данной рабочей нагрузки. Должно быть не менее **0**. | long | 10000 | [0,...] | средняя |
| **cache.max.bytes.buffering** | Максимальное количество байт памяти, которые будут использоваться для буферизации во всех стримах. | long | 10485760 | [0,...] | средняя |
| **client.id** | Строка префикса идентификатора, используемая как идентификатор клиентов для внутреннего потребителя, поставщика и потребителя восстановления, с шаблоном **<client.id>-StreamThread-<threadSequenceNumber$gt;-<consumer|producer|restore-consumer>**. | string | "" | -- | средняя |
| **default.deserialization.exception.handler** | Класс обработки исключений, имплементирующий интерфейс **org.apache.kafka.streams.errors.DeserializationExceptionHandler**. | class | org.apache.kafka.streams. errors.LogAndFailExceptionHandler | -- | средняя |
| **default.key.serde** | Класс сериализатора/десериализатора по умолчанию для ключа, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serde**. Обратите внимание, что при использовании оконного класса **serde**необходимо установить внутренний класс **serde**, который имплементирует интерфейс **org.apache.kafka.common.serialization.Serde** через **default.windowed.key.serde.inner** или **default.windowed.value.serde.inner** как правило. | class | null | -- | средняя |
| **default.list.key.serde.inner** | Внутренний класс списка **serde**по умолчанию для ключа, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serde**. Эта конфигурация будет прочитана тогда и только тогда, когда для конфигурации **default.key.serde** установлено значение **org.apache.kafka.common.serialization.Serdes.ListSerde**. | class | null | -- | средняя |
| **default.list.key.serde.type** | Класс по умолчанию для ключа, имплементирующего интерфейс **java.util.List**. Эта конфигурация будет прочитана тогда и только тогда, когда для конфигурации **default.key.serde** установлено значение **org.apache.kafka.common.serialization.Serdes.ListSerde**. Обратите внимание, что когда используется класс списка **serde**, необходимо установить внутренний класс **serde**, который имплементирует интерфейс **org.apache.kafka.common.serialization.Serde** через **default.list.key.serde.inner**. | class | null | -- | средняя |
| **default.list.value.serde.inner** | Внутренний класс по умолчанию для списка **serde**для значения, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serde**. Эта конфигурация будет прочитана тогда и только тогда, когда для конфигурации **default.value.serde** установлено значение **org.apache.kafka.common.serialization.Serdes.ListSerde**. | class | null | -- | средняя |
| **default.list.value.serde.type** | Класс по умолчанию для значения, имплементирующего интерфейс **java.util.List**. Эта конфигурация будет прочитана тогда и только тогда, когда для конфигурации **default.value.serde** установлено значение **org.apache.kafka.common.serialization.Serdes.ListSerde**. Обратите внимание, что при использовании класса **serde**списка необходимо установить внутренний класс **serde**, который имплементирует интерфейс **org.apache.kafka.common.serialization.Serde** через **default.list.value.serde.inner**. | class | null | -- | средняя |
| **default.production.exception.handler** | Класс обработки исключений, имплементирующий интерфейс **org.apache.kafka.streams.errors.ProductionExceptionHandler**. | class | org.apache.kafka.streams.errors. DefaultProductionExceptionHandler | -- | средняя |
| **default.timestamp.extractor** | Класс экстрактора меток времени по умолчанию, имплементирующий интерфейс **org.apache.kafka.streams.processor.TimestampExtractor**. | class | org.apache.kafka.streams. processor.FailOnInvalidTimestamp | -- | средняя |
| **default.value.serde** | Класс сериализатора/десериализатора по умолчанию для значения, имплементирующего интерфейс **org.apache.kafka.common.serialization.Serde**. Обратите внимание, что при использовании оконного класса **serde**необходимо установить внутренний класс **serde**, который имплементирует интерфейс **org.apache.kafka.common.serialization.Serde** через **default.windowed.key.serde.inner** или **default.windowed.value.serde.inner** как правило. | class | null | -- | средняя |
| **max.task.idle.ms** | Эта конфигурация контролирует, могут ли join-соединения и merge-слияния привести к результатам не в том порядке. Значение конфигурации — это максимальное время в миллисекундах, в течение которого задача стрима будет простаивать, когда она полностью занята некоторыми (но не всеми) входными партициями, чтобы дождаться, пока поставщики отправят дополнительные записи, и избежать потенциальной обработки записей не по порядку через несколько входных стримов. Значение по умолчанию (ноль) не ожидает, пока поставщики отправят дополнительные записи, но ожидает получения данных, которые уже присутствуют на брокерах. Это значение по умолчанию означает, что записи, которые уже присутствуют в брокерах, стримы будут обрабатывать их в порядке временных меток. Установите значение **-1**, чтобы полностью отключить простои и обрабатывать любые локально доступные данные, даже если это может привести к нарушению порядка обработки. | long | 0 | -- | средняя |
| **max.warmup.replicas** | Максимальное количество warmup-реплик (дополнительных резервных реплик помимо настроенного числа **num.standbys**), которое можно назначить одновременно, чтобы задача оставалась доступной на одном инстансе, пока она прогревается на другом инстансе, которому она была переназначена. Используется для регулирования объёма дополнительного трафика брокера и состояния кластера, который может использоваться для обеспечения высокой доступности. Должно быть не менее **1**. | int | 2 | [1,...] | средняя |
| **num.stream.threads** | Количество потоков для выполнения обработки стрима. | int | 1 | -- | средняя |
| **processing.guarantee** | Гарантия обработки, которую следует использовать. Возможные значения: **at\_least\_once** (по умолчанию) и **exactly\_once\_v2** (требуется версия брокера 2.5 или выше). Устаревшими параметрами являются **exactly\_once** (требуется версия брокера 0.11.0 или выше) и **exactly\_once\_beta** (требуется версия брокера 2.5 или выше). Обратите внимание, что для однократной обработки по умолчанию требуется кластер как минимум из трёх брокеров, что является рекомендуемой настройкой для промышленной среды; для разработки вы можете изменить это, изменив настройки брокера **transaction.state.log.replication.factor** и **transaction.state.log.min.isr**. | string | at\_least\_once | [at\_least\_once, exactly\_once, exactly\_once\_beta, exactly\_once\_v2] | средняя |
| **rack.aware.assignment.tags** | Список ключей клиентских тегов, используемых для распространения резервных реплик между инстансами RT.StreamingKafka Streams. После настройки RT.StreamingKafka Streams приложит все усилия, чтобы распределить резервные задачи по каждому измерению тега клиента. | list | "" | список, содержащий максимум 5 элементов | средняя |
| **replication.factor** | Коэффициент репликации для топиков лога изменений и топиков перераспределения партиций, созданных приложением обработки стримов. Значение по умолчанию **-1** (что означает: использовать коэффициент репликации брокера по умолчанию) требует брокера версии 2.4 или новее. | int | -1 | -- | средняя |
| **security.protocol** | Протокол, используемый для связи с брокерами. Допустимые значения: **PLAINTEXT**, **SSL**, **SASL\_PLAINTEXT**, **SASL\_SSL**. | string | PLAINTEXT | [PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL] | средняя |
| **statestore.cache.max.bytes** | Максимальное количество байт памяти, которые будут использоваться для кэша хранилища состояний во всех потоках. | long | 10485760 (10 мебибайт) | [0,...] | средняя |
| **task.timeout.ms** | Максимальное время в миллисекундах, на которое задача может остановиться из-за внутренних ошибок и повторять попытки, пока не возникнет ошибка. При таймауте 0 мс задача выдаст ошибку из-за первой внутренней ошибки. Если время ожидания превышает 0 мс, задача повторит попытку хотя бы один раз, прежде чем возникнет ошибка. | long | 300000 (5 минут) | [0,...] | средняя |
| **topology.optimization** | Конфигурация, сообщающая RT.StreamingKafka Streams, следует ли оптимизировать топологию и какие оптимизации применять. Допустимые значения: **"+NO\_OPTIMIZATION+"**, **"+OPTIMIZE+"** или список конкретных оптимизаций, разделенных запятыми: **("+REUSE\_KTABLE\_SOURCE\_TOPICS+", "+MERGE\_REPARTITION\_TOPICS+" + "SINGLE\_STORE\_SELF\_JOIN+")**. **"NO\_OPTIMIZATION"** по умолчанию. | string | none | org.apache.kafka.streams. StreamsConfig$$Lambda$3/ 521645586@49 | средняя |
| **application.server** | Пара хост:порт, указывающая на определяемый пользователем эндпоинт, который можно использовать для обнаружения хранилища состояний и интерактивных запросов в этом инстансе RT.StreamingKafka Streams. | string | "" | -- | низкая |
| **auto.include.jmx.reporter** | [**Устарело**] Следует ли автоматически включать JmxReporter, даже если он не указан в **metric.reporters**. Эта конфигурация будет удалена в Kafka 4.0, вместо этого пользователям следует включить **org.apache.kafka.common.metrics.JmxReporter** в **metric.reporters**, чтобы включить JmxReporter. | boolean | true | -- | низкая |
| **buffered.records.per.partition** | Максимальное количество записей для буферизации на партицию. | int | 1000 | -- | низкая |
| **built.in.metrics.version** | Используемая версия встроенных метрик. | string | latest | [latest] | низкая |
| **commit.interval.ms** | Частота в миллисекундах, с которой коммитится ход обработки. Для хотя бы однократной обработки (at-least-once) коммит означает сохранение позиции (т.е. смещений) процессора. Для однократной обработки (exactly-once) это означает коммит транзакции, который включает в себя сохранение позиции и делает закоммиченные данные в выходном топике видимыми для потребителей с уровнем изоляции **read\_commited**. Обратите внимание: если для параметра **processing.guarantee** установлены значения **exactly\_once\_v2**, **exactly\_once**, значение по умолчанию — **100**, в противном случае значение по умолчанию — **30000**. | long | 30000 (30 секунд) | [0,...] | низкая |
| **connections.max.idle.ms** | Закрывает неактивные соединения по истечении количества миллисекунд, указанного в этой конфигурации. | long | 540000 (9 минут) | -- | низкая |
| **default.dsl.store** | Тип хранилища состояний по умолчанию, используемый операторами DSL. | string | rocksDB | [rocksDB, in\_memory] | низкая |
| **metadata.max.age.ms** | Период времени в миллисекундах, по истечении которого мы принудительно обновляем метаданные, даже если мы не заметили никаких изменений лидерства партиции, чтобы заранее обнаружить любые новые брокеры или партиции. | long | 300000 (5 минут) | [0,...] | низкая |
| **metric.reporters** | Список классов для использования в качестве генераторов отчётов по метрикам. Имплементация интерфейса **org.apache.kafka.common.metrics.MetricsReporter** позволяет подключать классы, которые будут уведомляться о создании новой метрики. JmxReporter всегда включён для регистрации статистики JMX. | list | "" | -- | низкая |
| **metrics.num.samples** | Количество выборок, поддерживаемых для вычисления метрик. | int | 2 | [1,...] | низкая |
| **metrics.recording.level** | Самый высокий уровень записи метрик. | string | INFO | [INFO, DEBUG, TRACE] | низкая |
| **metrics.sample.window.ms** | Окно времени, в течение которого вычисляется выборка метрик. | long | 30000 (30 секунд) | [0,...] | низкая |
| **poll.ms** | Время в миллисекундах, в течение которого блокируется ожидание ввода. | long | 100 | -- | низкая |
| **probing.rebalance.interval.ms** | Максимальное время ожидания в миллисекундах перед запуском повторной балансировки для проверки реплик прогрева, которые завершили прогрев и готовы стать активными. Пробные ребалансировки будут продолжать выполняться до тех пор, пока распределение не будет сбалансировано. Должно быть не менее 1 минуты. | long | 600000 (10 минут) | [60000,...] | низкая |
| **receive.buffer.bytes** | Размер приёмного буфера TCP (SO\_RCVBUF), который будет использоваться при чтении данных. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 32768 (32 кибибайт) | [-1,...] | низкая |
| **reconnect.backoff.max.ms** | Максимальное время ожидания в миллисекундах при повторном подключении к брокеру, к которому неоднократно не удавалось подключиться. Если параметр предусмотрен, отсрочка на хост будет увеличиваться экспоненциально для каждого последующего сбоя соединения, вплоть до этого максимума. После расчёта увеличения задержки добавляется 20% случайного джиттера, чтобы избежать большого количества одновременных соединений. | long | 1000 (1 секунда) | [0,...] | низкая |
| **reconnect.backoff.ms** | Базовое время ожидания перед попыткой повторного подключения к данному хосту. Параметр позволяет избежать повторных подключений к хосту в узком цикле. Эта отсрочка применяется ко всем попыткам подключения клиента к брокеру. | long | 50 | [0,...] | низкая |
| **repartition.purge.interval.ms** | Частота в миллисекундах, с которой удаляются полностью использованные записи из топиков перераспределения партиций. Очистка произойдет как минимум через это время с момента последней очистки, но может быть отложена на более позднее время. Обратите внимание: в отличие от **commit.interval.ms**, значение по умолчанию для этого значения остается неизменным, если для **processing.guarantee** установлено значение **exactly\_once\_v2**). | long | 30000 (30 секунд) | [0,...] | низкая |
| **request.timeout.ms** | Конфигурация контролирует максимальное время, в течение которого клиент будет ждать ответа на запрос. Если ответ не получен до истечения таймаута, клиент при необходимости повторно отправит запрос или откажет в запросе, если повторные попытки исчерпаны. | int | 40000 (40 секунд) | [0,...] | низкая |
| **retries** | Установка значения больше нуля приведёт к тому, что клиент повторно отправит любой запрос, который завершится неудачно с потенциально временной ошибкой. Рекомендуется установить значение либо нулевое, либо **MAX\_VALUE** и использовать соответствующие параметры таймаута, чтобы контролировать, как долго клиент должен повторять запрос. | int | 0 | [0,...,2147483647] | низкая |
| **retry.backoff.ms** | Время ожидания перед попыткой повторения неудачного запроса к данной партиции топика. Параметр позволяет избежать повторной отправки запросов в тесном цикле в некоторых сценариях сбоя. | long | 100 | [0,...] | низкая |
| **rocksdb.config.setter** | Класс установщика конфигурации Rocks DB или имя класса, который имплементирует интерфейс **org.apache.kafka.streams.state.RocksDBConfigSetter**. | class | null | -- | низкая |
| **send.buffer.bytes** | Размер буфера отправки TCP (SO\_SNDBUF), который будет использоваться при отправке данных. Если значение равно **-1**, будет использоваться ОС по умолчанию. | int | 131072 (128 кибибайт) | [-1,...] | низкая |
| **state.cleanup.delay.ms** | Время ожидания в миллисекундах перед удалением состояния после того, как партиция мигрировала. Будут удалены только каталоги состояний, которые не были изменены как минимум для **state.cleanup.delay.ms**. | long | 600000 (10 минут) | -- | низкая |
| **upgrade.from** | Позволяет выполнить обновление с обратной совместимостью. Это необходимо при обновлении с [0.10.0, 1.1] до 2.0+ или при обновлении с [2.0, 2.3] до 2.4+. При обновлении с 3.3 на более новую версию указывать этот конфиг не требуется. По умолчанию установлено значение **null**. Допустимые значения: "0.10.0", "0.10.1", "0.10.2", "0.11.0", "1.0", "1.1", "2.0", "2.1", "2.2", "2.3", "2.4", "2.5", "2.6", "2.7", "2.8", "3.0", "3.1", "3.2", "3.3" (для обновления с соответствующей старой версии). | string | null | [null, 0.10.0, 0.10.1, 0.10.2, 0.11.0, 1.0, 1.1, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 3.0, 3.1, 3.2, 3.3] | низкая |
| **window.size.ms** | Устанавливает размер окна для десериализатора, чтобы вычислить время окончания окна. | long | null | -- | низкая |
| **windowed.inner.class.serde** | Сериализатор/десериализатор по умолчанию для внутреннего класса оконной записи. Должен имплементировать интерфейс **org.apache.kafka.common.serialization.Serde**. Обратите внимание, что установка этой конфигурации в приложении RT.StreamingKafka Streams приведёт к ошибке, поскольку она предназначена для использования только из обычного потребительского клиента. | string | null | -- | низкая |
| **windowstore.changelog.additional.retention.ms** | Добавленный в окно MaintenanceMs гарантирует, что данные не будут удалены из лога преждевременно. Допускает дрейф часов. По умолчанию – 1 день. | long | 86400000 (1 день) | -- | низкая |

# 3 Безопасность

## 3.1 Обзор безопасности

В настоящее время в RT.StreamingKafka поддерживаются следующие меры безопасности:

1. Аутентификация подключений к брокерам от клиентов (поставщиков и потребителей), других брокеров и инструментов с использованием SSL или SASL. RT.StreamingKafka поддерживает следующие механизмы SASL:

* SASL/GSSAPI (Kerberos;
* SASL/PLAIN;
* SASL/SCRAM-SHA-256 и SASL/SCRAM-SHA-512;
* SASL/OAUTHBEARER.

2. Аутентификация подключений брокеров к ZooKeeper.

3. Шифрование данных, передаваемых между брокерами и клиентами, между брокерами или между брокерами и инструментами с использованием SSL (обратите внимание, что при включении SSL происходит снижение производительности, величина которого зависит от типа ЦП и реализации JVM).

4. Авторизация операций чтения/записи клиентами.

5. Авторизация с поддержкой интеграции с внешними сервисами авторизации.

В приведённых ниже разделах объясняется, как настроить и использовать функции безопасности как в клиентах, так и в брокерах.

## 3.2 Конфигурация слушателя

Чтобы защитить кластер RT.StreamingKafka, необходимо защитить каналы, используемые для связи с серверами. Каждый сервер должен определить набор слушателей, которые используются для получения запросов от клиентов, а также от других серверов. Каждый слушатель может быть настроен для аутентификации клиентов с использованием различных механизмов и для обеспечения шифрования трафика между сервером и клиентом. В этом разделе представлены основы настройки слушателей.

Серверы RT.StreamingKafka поддерживают прослушивание соединений на нескольких портах. Это настраивается с помощью свойства **listeners**в конфигурации сервера, которое принимает разделённый запятыми список слушателей, которых необходимо включить. На каждом сервере должен быть определён хотя бы один слушатель. Формат каждого слушателя, определённого в списке **listeners**, приведён ниже:

{LISTENER\_NAME}://{hostname}:{port}

Copy

**LISTENER\_NAME** обычно представляет собой описательное имя, определяющее цель слушателя. Например, во многих конфигурациях используется отдельный слушатель клиентского трафика, поэтому в конфигурации соответствующий слушатель может называться **CLIENT**:

listeners=CLIENT://localhost:9092

Copy

Протокол безопасности каждого слушателя определяется в отдельной конфигурации: **listener.security.protocol.map**. Значение представляет собой разделённый запятыми список каждого слушателя, сопоставленного с его протоколом безопасности. Например, следующая конфигурация значения указывает, что слушатель **CLIENT**будет использовать **SSL**, а слушатель **BROKER**будет использовать **PLAINTEXT**.

listener.security.protocol.map=CLIENT:SSL,BROKER:PLAINTEXT

Copy

Возможные варианты протокола безопасности приведены ниже:

1. **PLAINTEXT**.
2. **SSL**.
3. **SASL\_PLAINTEXT**.
4. **SASL\_SSL**.

Протокол **PLAINTEXT** не обеспечивает безопасности и не требует дополнительной настройки. В следующих разделах описано, как настроить остальные протоколы.

Если каждый слушатель использует отдельный протокол безопасности, также можно использовать имя протокола безопасности в качестве имени слушателя в **listeners**. Используя приведённый выше пример, мы могли бы пропустить определение слушателей **CLIENT**и **BROKER**, используя следующее определение:

listeners=SSL://localhost:9092,PLAINTEXT://localhost:9093

Copy

Однако мы рекомендуем пользователям указывать явные имена слушателей, поскольку это делает предполагаемое использование каждого слушателя более понятным.

Среди слушателей в этом списке можно объявить слушателя, который будет использоваться для связи между брокерами, задав в конфигурации имя слушателя **inter.broker.listener.name**. Основная цель слушателя между брокерами — репликация партиций. Если он не определён, то межброкерский слушатель определяется протоколом безопасности, определённым в **security.inter.broker.protocol**, который по умолчанию имеет значение **PLAINTEXT**.

Для устаревших кластеров, которые используют Zookeeper для хранения метаданных кластера, можно объявить отдельного слушателя, который будет использоваться для распространения метаданных от активного контроллера к брокерам. Это определяется в **control.plane.listener.name**. Активный контроллер будет использовать этого слушателя, когда ему необходимо отправить обновления метаданных брокерам в кластере. Преимущество использования такого слушателя заключается в том, что он использует отдельный поток обработки, что снижает вероятность того, что трафик приложений будет препятствовать своевременному распространению изменений метаданных (таких как лидер партиции и обновления ISR). Обратите внимание, что значение по умолчанию равно **null**, что означает, что контроллер будет использовать того же слушателя, который определён в **inter.broker.listener**.

В кластере KRaft брокером является любой сервер, на котором включена роль **broker**в файле **process.roles**, а контроллером — любой сервер, на котором включена роль **controller**. Конфигурация слушателя зависит от роли. Слушатель, определённый **inter.broker.listener.name**, используется исключительно для запросов между брокерами. С другой стороны, контроллеры должны использовать отдельного слушателя, который определяется конфигурацией **controller.listener.names**. Для него нельзя установить то же значение, что и для межброкерского слушателя.

Контроллеры получают запросы как от других контроллеров, так и от брокеров. По этой причине, даже если на сервере не включена роль **controller**(т.е. он является просто брокером), он всё равно должен определить слушателя контроллера вместе со всеми свойствами безопасности, необходимыми для его настройки. Например, мы можем использовать следующую конфигурацию для standalone-брокера:

process.roles=broker

listeners=BROKER://localhost:9092

inter.broker.listener.name=BROKER

controller.quorum.voters=0@localhost:9093

controller.listener.names=CONTROLLER

listener.security.protocol.map=BROKER:SASL\_SSL,CONTROLLER:SASL\_SSL

Copy

В этом примере слушатель контроллера по-прежнему настроен на использование протокола безопасности **SASL\_SSL**, но он не включён в **listeners**, поскольку брокер не предоставляет самого слушателя контроллера. Порт, который будет использоваться в этом случае, берётся из конфигурации **controller.quorum.voters**, которая определяет полный список контроллеров.

Для серверов KRaft, на которых включена роль брокера и контроллера, конфигурация аналогична. Единственное отличие состоит в том, что слушатель контроллера должен быть включён в список **listeners**:

process.roles=broker,controller

listeners=BROKER://localhost:9092,CONTROLLER://localhost:9093

inter.broker.listener.name=BROKER

controller.quorum.voters=0@localhost:9093

controller.listener.names=CONTROLLER

listener.security.protocol.map=BROKER:SASL\_SSL,CONTROLLER:SASL\_SSL

Copy

Требуется, чтобы порт, определённый в **controller.quorum.voters**, точно соответствовал одному из открытых слушателей контроллера. Например, здесь слушатель **CONTROLLER**привязан к порту 9093. Строка подключения, определённая **controller.quorum.voters**, должна затем также использовать порт 9093, как это происходит здесь.

Контроллер будет принимать запросы от всех слушателей, определённых в **controller.listener.names**. Обычно имеется только один слушатель контроллера, но возможно и больше. Например, это даёт возможность сменить активного слушателя с одного порта или протокола безопасности на другой посредством смены кластера (одна смена для предоставления нового слушателя и одна смена для удаления старого слушателя). Если определено несколько слушателей контроллера, первый в списке будет использоваться для исходящих запросов.

В RT.StreamingKafka принято использовать отдельного слушателя для клиентов. Это позволяет изолировать внутрикластерных слушателей на сетевом уровне. В случае с слушателем контроллера в KRaft, слушатель должен быть изолирован, так как клиенты всё равно с ним не работают. Ожидается, что клиенты будут подключаться к любому другому слушателю, настроенному на брокере.

В следующем разделе описано, как включить SSL на слушателе для шифрования, а также аутентификации. В следующем разделе будут рассмотрены дополнительные механизмы аутентификации с использованием SASL.

## 3.3 Шифрование и аутентификация с использованием SSL

RT.StreamingKafka позволяет клиентам использовать SSL для шифрования трафика, а также аутентификации. По умолчанию SSL отключён, но при необходимости его можно включить. В следующих разделах подробно объясняется, как настроить собственную инфраструктуру PKI, использовать ее для создания сертификатов и настроить Kafka для их использования.

### 3.3.1 Генерация ключа SSL и сертификата для каждого брокера

Первым шагом развёртывания одного или нескольких брокеров с поддержкой SSL является создание пары открытого/закрытого ключей для каждого сервера. Поскольку RT.StreamingKafka ожидает, что все ключи и сертификаты будут храниться в хранилищах ключей, для этой задачи мы будем использовать команду Java keytool. Инструмент поддерживает два разных формата хранилища ключей: формат **jks**, специфичный для Java, который на данный момент устарел, а также **PKCS12**. PKCS12 является форматом по умолчанию в Java версии 9. Чтобы гарантировать использование этого формата независимо от используемой версии Java, во всех следующих командах явно указывается формат PKCS12.

> keytool -keystore {keystorefile} -alias localhost -validity {validity} -genkey -keyalg RSA -storetype pkcs12

Copy

Вам необходимо указать два параметра в приведённой выше команде:

* **keystorefile**-- файл хранилища ключей, в котором хранятся ключи (а затем и сертификат) этого брокера. Файл хранилища ключей содержит закрытые и открытые ключи этого брокера, поэтому его необходимо хранить в безопасности. В идеале этот шаг выполняется на брокере RT.StreamingKafka, на котором будет использоваться ключ, поскольку этот ключ никогда не должен передаваться или покидать сервер, для которого он предназначен.
* **validity**-- срок действия ключа в днях. Обратите внимание, что этот срок отличается от срока действия сертификата, который будет определён при подписании сертификата. Вы можете использовать один и тот же ключ для запроса нескольких сертификатов: если срок действия вашего ключа составляет 10 лет, но ваш центр сертификации будет подписывать только сертификаты, действительные в течение одного года, вы можете использовать один и тот же ключ с 10 сертификатами с течением времени.

Чтобы получить сертификат, который можно использовать с только что созданным закрытым ключом, необходимо создать запрос на подпись сертификата. Этот запрос на подпись, подписанный доверенным центром сертификации, приводит к созданию фактического сертификата, который затем можно установить в хранилище ключей и использовать для целей аутентификации.

Чтобы сгенерировать запросы на подпись сертификата, выполните следующую команду для всех хранилищ ключей сервера, созданных на данный момент.

> keytool -keystore server.keystore.jks -alias localhost -validity {validity} -genkey -keyalg RSA -destkeystoretype pkcs12 -ext SAN=DNS:{FQDN},IP:{IPADDRESS1}

Copy

В этой команде предполагается, что вы хотите добавить в сертификат информацию об имени хоста. Если это не так, вы можете опустить параметр расширения **-ext SAN=DNS:{FQDN},IP:{IPADDRESS1}**.

Проверка имени хоста, если она включена, представляет собой процесс проверки атрибутов сертификата, предоставленного сервером, к которому вы подключаетесь, по фактическому имени хоста или IP-адресу этого сервера, чтобы убедиться, что вы действительно подключаетесь к правильному серверу.

Основная причина этой проверки — предотвращение атак «человек посередине». Для RT.StreamingKafka эта проверка долгое время была отключена по умолчанию, но начиная с версии 2.0.0 проверка имени хоста серверов включена по умолчанию для клиентских подключений, а также для межброкерских соединений.

Проверку имени хоста сервера можно отключить, задав для **ssl.endpoint.identification.algorithm** пустую строку.

Для динамически настроенных слушателей брокера проверку имени хоста можно отключить с помощью **kafka-configs.sh**:

> bin/kafka-configs.sh --bootstrap-server localhost:9093 --entity-type brokers --entity-name 0 --alter --add-config "listener.name.internal.ssl.endpoint.identification.algorithm="

Copy

Если проверка имени хоста включена, клиенты будут проверять полное доменное имя (FQDN) или IP-адрес сервера по одному из следующих двух полей:

* Общее имя (Common Name, CN).
* Альтернативное имя субъекта (Subject Alternative Name, SAN).

Хотя RT.StreamingKafka проверяет оба поля, использование поля CN для проверки имени хоста устарело, и его следует избегать, если это возможно. Кроме того, поле SAN гораздо более гибкое, позволяя объявлять в сертификате несколько записей DNS и IP.

Ещё одним преимуществом является то, что если поле SAN используется для проверки имени хоста, для общего имени можно установить более значимое значение для целей авторизации. Поскольку нам необходимо, чтобы поле SAN содержалось в подписанном сертификате, оно будет указано при формировании запроса на подпись. Его также можно указать при создании пары ключей, но он не будет автоматически скопирован в запрос на подпись.

Чтобы добавить поле SAN, добавьте следующий аргумент **-ext SAN=DNS:{FQDN},IP:{IPADDRESS}** к команде **keytool**:

> keytool -keystore server.keystore.jks -alias localhost -validity {validity} -genkey -keyalg RSA -destkeystoretype pkcs12 -ext SAN=DNS:{FQDN},IP:{IPADDRESS1}

Copy

### 3.3.2 Создание собственного центра сертификации

После этого шага каждая машина в кластере имеет пару открытого и закрытого ключей, которую уже можно использовать для шифрования трафика, и запрос на подпись сертификата, который является основой для создания сертификата. Чтобы добавить возможности аутентификации, этот запрос на подпись должен быть подписан доверенным центром, который будет создан на этом этапе.

Центр сертификации (CA) отвечает за подпись сертификатов. Центры сертификации работают так же, как правительство, которое выдает паспорта: правительство ставит печать (подписывает) на каждом паспорте, так что паспорт становится трудно подделать. Другие правительства проверяют штампы, чтобы убедиться в подлинности паспорта. Точно так же центр сертификации подписывает сертификаты, а криптография гарантирует, что подписанный сертификат сложно подделать с вычислительной точки зрения. Таким образом, пока центр сертификации является подлинным и заслуживающим доверия центром, клиенты имеют надёжную уверенность в том, что они подключаются к подлинным машинам.

В этом руководстве мы будем собственным центром сертификации. При настройке производственного кластера в корпоративной среде эти сертификаты обычно подписываются корпоративным центром сертификации, которому доверяют во всей компании.

Из-за [ошибки](https://www.openssl.org/docs/man1.1.1/man1/x509.html#BUGS)в OpenSSL модуль **x509** не копирует запрошенные поля расширения из CSR в окончательный сертификат. Поскольку мы хотим, чтобы в нашем сертификате присутствовало расширение SAN, чтобы обеспечить проверку имени хоста, вместо этого мы будем использовать модуль **ca**. Это требует некоторой дополнительной настройки, прежде чем мы создадим нашу пару ключей CA.

Сохраните следующий листинг в файл **openssl-ca.cnf**и при необходимости измените значения достоверности и общих атрибутов.

HOME = .

RANDFILE = $ENV::HOME/.rnd

####################################################################

[ ca ]

default\_ca = CA\_default # The default ca section

[ CA\_default ]

base\_dir = .

certificate = $base\_dir/cacert.pem # The CA certifcate

private\_key = $base\_dir/cakey.pem # The CA private key

new\_certs\_dir = $base\_dir # Location for new certs after signing

database = $base\_dir/index.txt # Database index file

serial = $base\_dir/serial.txt # The current serial number

default\_days = 1000 # How long to certify for

default\_crl\_days = 30 # How long before next CRL

default\_md = sha256 # Use public key default MD

preserve = no # Keep passed DN ordering

x509\_extensions = ca\_extensions # The extensions to add to the cert

email\_in\_dn = no # Don't concat the email in the DN

copy\_extensions = copy # Required to copy SANs from CSR to cert

####################################################################

[ req ]

default\_bits = 4096

default\_keyfile = cakey.pem

distinguished\_name = ca\_distinguished\_name

x509\_extensions = ca\_extensions

string\_mask = utf8only

####################################################################

[ ca\_distinguished\_name ]

countryName = Country Name (2 letter code)

countryName\_default = DE

stateOrProvinceName = State or Province Name (full name)

stateOrProvinceName\_default = Test Province

localityName = Locality Name (eg, city)

localityName\_default = Test Town

organizationName = Organization Name (eg, company)

organizationName\_default = Test Company

organizationalUnitName = Organizational Unit (eg, division)

organizationalUnitName\_default = Test Unit

commonName = Common Name (e.g. server FQDN or YOUR name)

commonName\_default = Test Name

emailAddress = Email Address

emailAddress\_default = test@test.com

####################################################################

[ ca\_extensions ]

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always, issuer

basicConstraints = critical, CA:true

keyUsage = keyCertSign, cRLSign

####################################################################

[ signing\_policy ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

####################################################################

[ signing\_req ]

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer

basicConstraints = CA:FALSE

keyUsage = digitalSignature, keyEncipherment

Copy

Затем создайте базу данных и файл серийных номеров, они будут использоваться для отслеживания того, какие сертификаты были подписаны этим центром сертификации. Оба они представляют собой просто текстовые файлы, которые находятся в том же каталоге, что и ваши ключи ЦС.

> echo 01 > serial.txt

> touch index.txt

Copy

Выполнив эти шаги, вы готовы создать свой центр сертификации, который позже будет использоваться для подписи сертификатов.

> openssl req -x509 -config openssl-ca.cnf -newkey rsa:4096 -sha256 -nodes -out cacert.pem -outform PEM

Copy

Центр сертификации — это просто пара открытого/закрытого ключей и сертификат, который подписывается сам собой и предназначен только для подписи других сертификатов.

Эту пару ключей следует хранить в полной безопасности: если кто-то получит к ней доступ, он сможет создавать и подписывать сертификаты, которым будет доверять ваша инфраструктура, а это означает, что он сможет выдавать себя за кого угодно при подключении к любой службе, доверяющей этому ЦС.

Следующим шагом будет добавление сгенерированного центра сертификации в \*\*хранилище доверенных сертификатов клиентов\*\*, чтобы клиенты могли доверять этому центру сертификации:

> keytool -keystore client.truststore.jks -alias CARoot -import -file ca-cert

Copy

**Примечание**. Если вы настраиваете брокеров RT.StreamingKafka на требование аутентификации клиента, установив для **ssl.client.auth** значение **required**или **required**в конфигурации брокеров RT.StreamingKafka, тогда вы также должны предоставить хранилище доверенных сертификатов для брокеров RT.StreamingKafka, и в нём должны быть все сертификаты ЦС, которыми были подписаны ключи клиентов.

> keytool -keystore server.truststore.jks -alias CARoot -import -file ca-cert

Copy

В отличие от хранилища ключей на шаге 1, в котором хранятся собственные идентификаторы каждой машины, в хранилище доверенных сертификатов клиента хранятся все сертификаты, которым клиент должен доверять. Импорт сертификата в хранилище доверенных сертификатов также означает доверие ко всем сертификатам, подписанным этим сертификатом. Как и в приведённой выше аналогии, доверие правительству (ЦС) также означает доверие ко всем выданным им паспортам (сертификатам). Этот атрибут называется цепочкой доверия, и он особенно полезен при развёртывании SSL в большом кластере RT.StreamingKafka. Вы можете подписать все сертификаты в кластере с помощью одного центра сертификации, и все машины будут использовать одно и то же хранилище доверенных сертификатов, которое доверяет этому центру сертификации. Таким образом, все машины смогут аутентифицировать все остальные машины.

### 3.3.3 Подписание сертификата

Подпишите сертификат его с помощью ЦС:

> openssl ca -config openssl-ca.cnf -policy signing\_policy -extensions signing\_req -out {server certificate} -infiles {certificate signing request}

Copy

Наконец, вам необходимо импортировать сертификат ЦС и подписанный сертификат в хранилище ключей:

> keytool -keystore {keystore} -alias CARoot -import -file {CA certificate}

> keytool -keystore {keystore} -alias localhost -import -file cert-signed

Copy

Определения параметров следующие:

* **keystore**-- расположение хранилища ключей.
* **CA certificate** -- сертификат ЦС.
* **certificate signing request** -- csr, созданный с использованием ключа сервера.
* **server certificate** -- файл для записи подписанного сертификата сервера.

В результате у вас останется одно хранилище доверенных сертификатов под названием **truststore.jks** — оно может быть одинаковым для всех клиентов и брокеров и не содержит никакой конфиденциальной информации, поэтому нет необходимости защищать его.

Кроме того, у вас будет один файл **server.keystore.jks** для каждой ноды, которая будет содержать ключи этой ноды, сертификат и сертификат вашего центра сертификации.

#### 3.3.3.1 SSL-ключ и сертификаты в формате PEM

Начиная с версии 2.7.0 хранилища ключей и доверенных сертификатов SSL можно настроить для брокеров и клиентов RT.StreamingKafka непосредственно в конфигурации в формате PEM. Это позволяет избежать необходимости хранить отдельные файлы в файловой системе и воспользоваться преимуществами функций защиты паролем конфигурации RT.StreamingKafka. PEM также может использоваться в качестве типа хранилища для файловых хранилищ ключей и доверенных сертификатов в дополнение к JKS и PKCS12. Чтобы настроить хранилище ключей PEM непосредственно в конфигурации брокера или клиента, закрытый ключ в формате PEM должен быть указан в **ssl.keystore.key**, а цепочка сертификатов в формате PEM должна быть предоставлена в **ssl.keystore.certificate.chain**. Чтобы настроить хранилище доверенных сертификатов, доверенные сертификаты, например. общедоступный сертификат ЦС должен быть указан в **ssl.truststore.certificates**. Поскольку PEM обычно хранится в виде многострочных строк в формате Base64, значение конфигурации может быть включено в конфигурацию RT.StreamingKafka в виде многострочных строк со строками, заканчивающимися обратной косой чертой ('\') для продолжения строки.

Конфигурации паролей хранилища **ssl.keystore.password** и **ssl.truststore.password** не используются для PEM. Если закрытый ключ зашифрован с использованием пароля, пароль ключа должен быть указан в **ssl.key.password**. Приватные ключи могут предоставляться в незашифрованном виде без пароля. В этом случае в производственных развёртываниях конфигурации должны быть зашифрованы или экспортированы с использованием функции защиты паролем в RT.StreamingKafka. Обратите внимание, что механизм SSL по умолчанию имеет ограниченные возможности для расшифровки зашифрованных закрытых ключей, когда для шифрования используются внешние инструменты, такие как OpenSSL. Сторонние библиотеки, такие как BouncyCastle, могут быть интегрированы с кастомной SslEngineFactory для поддержки более широкого спектра зашифрованных закрытых ключей.

### 3.3.4 Распространённые ошибки

В приведённых выше разделах показан процесс создания собственного центра сертификации и использования его для подписи сертификатов для вашего кластера. Хотя это очень полезно для песочницы, разработки, тестирования и подобных систем, обычно это некорректный процесс создания сертификатов для производственного кластера в корпоративной среде. Предприятия обычно используют свой собственный центр сертификации, и пользователи могут отправлять CSR для подписания с помощью этого центра сертификации, что даёт преимущество в том, что пользователи не несут ответственности за обеспечение безопасности центра сертификации, а также в качестве центрального органа, которому каждый может доверять. Однако это также отнимает у пользователя значительную часть контроля над процессом подписания сертификатов. Довольно часто лица, управляющие корпоративными центрами сертификации, применяют жёсткие ограничения к сертификатам, что может вызвать проблемы при попытке использовать эти сертификаты с RT.StreamingKafka.

1. [Расширенное использование ключей](https://tools.ietf.org/html/rfc5280#section-4.2.1.12). Сертификаты могут содержать поле расширения, которое определяет цель использования сертификата. Если это поле пустое, ограничений на использование нет, но если здесь указано какое-либо использование, действительные реализации SSL должны обеспечить соблюдение этого использования. Соответствующие варианты использования RT.StreamingKafka:

* Аутентификация клиента (Client authentication).
* Аутентификация сервера (Server authentication).

Брокерам RT.StreamingKafka необходимо разрешить оба этих использования, поскольку для внутрикластерной связи каждый брокер будет вести себя и как клиент, и как сервер по отношению к другим брокерам. Корпоративные центры сертификации нередко имеют профиль подписи для веб-серверов и используют его также для RT.StreamingKafka, который будет содержать только значение использования **serverAuth** и приведёт к сбою установления связи SSL.

2. Промежуточные сертификаты. Корпоративные корневые центры сертификации часто отключаются по соображениям безопасности. Для обеспечения стабильного использования создаются так называемые промежуточные центры сертификации, которые затем используются для подписи окончательных сертификатов. При импорте в хранилище ключей сертификата, подписанного промежуточным ЦС, необходимо обеспечить всю цепочку доверия вплоть до корневого ЦС. Это можно сделать, просто объединив файлы сертификатов в один файл сертификата, а затем импортировав его с помощью **keytool**.

3. Не удалось скопировать поля расширения. Операторы ЦС часто не решаются копировать и запрашивать поля расширения из CSR и предпочитают указывать их самостоятельно, поскольку это затрудняет получение злоумышленником сертификатов с потенциально вводящими в заблуждение или мошенническими значениями. Рекомендуется дважды проверить подписанные сертификаты, содержат ли они все запрошенные поля SAN, чтобы обеспечить корректную проверку имени хоста. Следующую команду можно использовать для вывода на консоль сведений о сертификате, которые следует сравнить с тем, что было первоначально запрошено:

> openssl x509 -in certificate.crt -text -noout

Copy

### 3.3.5 Настройка брокеров Kafka

Если SSL не включён для связи между брокерами (см. ниже, как его включить), потребуются порты **PLAINTEXT**и **SSL**.

listeners=PLAINTEXT://host.name:port,SSL://host.name:port

Copy

На стороне брокера необходимы следующие конфигурации SSL:

ssl.keystore.location=/var/private/ssl/server.keystore.jks

ssl.keystore.password=test1234

ssl.key.password=test1234

ssl.truststore.location=/var/private/ssl/server.truststore.jks

ssl.truststore.password=test1234

Copy

**Примечание**. **ssl.truststore.password** технически необязателен, но настоятельно рекомендуется. Если пароль не установлен, доступ к хранилищу доверенных сертификатов по-прежнему доступен, но проверка целостности отключена. Дополнительные настройки, на которые стоит обратить внимание:

1. **ssl.client.auth=none** (**required**-- требуется аутентификация клиента, **requested**-- запрашивается аутентификация клиента, и клиент без сертификатов всё равно может подключиться. Использование **requested**не рекомендуется, поскольку оно создает ложное ощущение клиенты безопасности и неправильно настроенные клиенты всё равно будут успешно подключаться.)
2. **ssl.cipher.suites** [опционально]. Набор шифров — это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL. (По умолчанию — пустой список)
3. **ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1** (перечислите протоколы SSL, которые вы собираетесь принимать от клиентов. Обратите внимание, что SSL устарел в пользу TLS, и использование SSL в рабочей среде не рекомендуется).
4. **ssl.keystore.type=JKS**.
5. **ssl.truststore.type=JKS**.
6. **ssl.secure.random.implementation=SHA1PRNG**.

Если вы хотите включить SSL для связи между брокерами, добавьте следующее в файл **server.properties** (по умолчанию -- **PLAINTEXT**):

security.inter.broker.protocol=SSL

Copy

Из-за правил импорта в некоторых странах реализация Oracle ограничивает стойкость криптографических алгоритмов, доступных по умолчанию. Если необходимы более сильные алгоритмы (например, AES с 256-битными ключами), необходимо получить [файлы политики юрисдикции неограниченной силы JCE](http://www.oracle.com/technetwork/java/javase/downloads/index.html) и установить их в JDK/JRE. Дополнительную информацию см. в [документации поставщиков JCA](https://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html).

JRE/JDK будет иметь генератор псевдослучайных чисел (PRNG) по умолчанию, который используется для операций шифрования, поэтому не требуется настраивать имплементацию, используемую **ssl.secure.random.implementation**. Однако в некоторых имплементациях возникают проблемы с производительностью (в частности, выбранный по умолчанию в системах Linux **NativePRNG** использует глобальную блокировку). В случаях, когда производительность SSL-соединений становится проблемой, рассмотрите возможность явно указать используемую имплементацию. Имплементация **SHA1PRNG**не является блокирующей и показала очень хорошие характеристики производительности при большой нагрузке (50 МБ/с создаваемых сообщений плюс трафик репликации для каждого брокера).

Как только вы запустите брокера, вы сможете увидеть в файле **server.log**.

with addresses: PLAINTEXT -> EndPoint(192.168.64.1,9092,PLAINTEXT),SSL -> EndPoint(192.168.64.1,9093,SSL)

Copy

Чтобы быстро проверить, правильно ли настроены хранилище ключей и доверенное хранилище сервера, вы можете выполнить следующую команду:

> openssl s\_client -debug -connect localhost:9093 -tls1

Copy

**Примечание**. **TLSv1**должен быть указан под **ssl.enabled.protocols**.

В выводе этой команды вы должны увидеть сертификат сервера:

-----BEGIN CERTIFICATE-----

{variable sized random bytes}

-----END CERTIFICATE-----

subject=/C=US/ST=CA/L=Santa Clara/O=org/OU=org/CN=Sriharsha Chintalapani

issuer=/C=US/ST=CA/L=Santa Clara/O=org/OU=org/CN=kafka/emailAddress=test@test.com

Copy

Если сертификат не отображается или появляются другие сообщения об ошибках, значит, ваше хранилище ключей настроено неправильно.

### 3.3.6 Настройка клиентов

SSL поддерживается только для новых поставщиков и потребителей RT.StreamingKafka, старый API не поддерживается. Конфигурации SSL будут одинаковыми как для поставщика, так и для потребителя.

Если аутентификация клиента в брокере не требуется, то ниже приведён пример минимальной конфигурации:

security.protocol=SSL

ssl.truststore.location=/var/private/ssl/client.truststore.jks

ssl.truststore.password=test1234

Copy

**Примечание**. **ssl.truststore.password** технически необязателен, но настоятельно рекомендуется. Если пароль не установлен, доступ к хранилищу доверенных сертификатов по-прежнему доступен, но проверка целостности отключена. Если требуется аутентификация клиента, необходимо создать хранилище ключей, как на шаге 1, а также настроить следующее:

ssl.keystore.location=/var/private/ssl/client.keystore.jks

ssl.keystore.password=test1234

ssl.key.password=test1234

Copy

Другие настройки конфигурации, которые также могут потребоваться в зависимости от наших требований и конфигурации брокера:

1. **ssl.provider** [опционально]. Имя поставщика безопасности, используемого для SSL-соединений. Значение по умолчанию — поставщик безопасности по умолчанию для JVM.
2. **ssl.cipher.suites** [опционально]. Набор шифров — это именованная комбинация аутентификации, шифрования, MAC и алгоритма обмена ключами, используемая для согласования настроек безопасности для сетевого подключения с использованием сетевого протокола TLS или SSL.
3. **ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1**. В нём должен быть указан хотя бы один из протоколов, настроенных на стороне брокера.
4. **ssl.truststore.type=JKS**.
5. **ssl.keystore.type=JKS**.

Примеры использования **console-producer** и **console-consumer**:

> kafka-console-producer.sh --bootstrap-server localhost:9093 --topic test --producer.config client-ssl.properties

> kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic test --consumer.config client-ssl.properties

Copy

## 3.4 Аутентификация с использованием SASL

### 3.4.1 Конфигурация JAAS

RT.StreamingKafka использует службу аутентификации и авторизации Java ([Java Authentication and Authorization Service, JAAS](https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html)) для настройки SASL.

#### 3.4.1.1 Конфигурация JAAS для брокеров

**KafkaServer**— это имя раздела в файле JAAS, используемое каждым **KafkaServer**/**Broker**. В этом разделе представлены параметры конфигурации SASL для брокера, включая любые клиентские подключения SASL, установленные брокером для связи между брокерами. Если несколько слушателей настроены на использование SASL, перед именем раздела может стоять имя слушателя в нижнем регистре, за которым следует точка, например: **sasl\_ssl.KafkaServer**.

Раздел **Client**используется для аутентификации SASL-соединения с Zookeeper. Это также позволяет брокерам устанавливать SASL ACL на нодах Zookeeper, который блокирует эти ноды, чтобы только брокеры могли его изменить. Необходимо иметь одно и то же основное имя для всех брокеров. Если вы хотите использовать имя раздела, отличное от **Client**, установите для системного свойства **zookeeper.sasl.clientconfig** соответствующее имя (например, **-Dzookeeper.sasl.clientconfig=ZkClient**).

ZooKeeper по умолчанию использует **zookeeper**в качестве имени службы. Если вы хотите изменить это, задайте для системного свойства **zookeeper.sasl.client.username** соответствующее имя (например, **-Dzookeeper.sasl.client.username=zk**).

Брокеры также могут настроить JAAS, используя свойство конфигурации брокера **sasl.jaas.config**. Имя свойства должно начинаться с префикса слушателя, включая механизм SASL, т.е. **listener.name.{listenerName}.{saslMechanism}.sasl.jaas.config**. В значении конфигурации можно указать только один модуль входа. Если на слушателя настроено несколько механизмов, конфигурации должны быть предоставлены для каждого механизма с использованием слушателя и префикса механизма. Например,

listener.name.sasl\_ssl.scram-sha-256.sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="admin" \

password="admin-secret";

listener.name.sasl\_ssl.plain.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \

username="admin" \

password="admin-secret" \

user\_admin="admin-secret" \

user\_alice="alice-secret";

Copy

Если конфигурация JAAS определена на разных уровнях, используется следующий порядок приоритета:

1. Свойство конфигурации брокера **listener.name.{listenerName}.{saslMechanism}.sasl.jaas.config**.
2. Раздел **{listenerName}.KafkaServer** статической конфигурации JAAS.
3. Раздел **KafkaServer**статической конфигурации JAAS.

Обратите внимание, что конфигурацию ZooKeeper JAAS можно настроить только с использованием статической конфигурации JAAS.

#### 3.4.1.2 Конфигурация JAAS для клиентов

Клиенты могут настроить JAAS, используя свойство конфигурации клиента **sasl.jaas.config** или используя статический файл конфигурации JAAS, аналогичный брокерам.

1. Конфигурация JAAS с использованием свойства конфигурации клиента

Клиенты могут указать конфигурацию JAAS как свойство поставщика или потребителя без создания физического файла конфигурации. Этот режим также позволяет различным поставщикам и потребителям в одной JVM использовать разные учётные данные, указывая разные свойства для каждого клиента. Если указаны как статическое системное свойство конфигурации JAAS **java.security.auth.login.config**, так и свойство клиента **sasl.jaas.config**, будет использоваться свойство клиента.

2. Конфигурация JAAS с использованием статического файла конфигурации.

Чтобы настроить аутентификацию SASL на клиентах с использованием статического файла конфигурации JAAS:

* Добавьте файл конфигурации JAAS с разделом входа клиента с именем **KafkaClient**. Настройте модуль входа в **KafkaClient**для выбранного механизма. Например, учётные данные **GSSAPI**можно настроить как:

KafkaClient {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/etc/security/keytabs/kafka\_client.keytab"

principal="kafka-client-1@EXAMPLE.COM";

};

Copy

* Передайте расположение файла конфигурации JAAS в качестве параметра JVM каждой клиентской JVM. Например:

-Djava.security.auth.login.config=/etc/kafka/kafka\_client\_jaas.conf

Copy

### 3.4.2 Конфигурация SASL

SASL может использоваться с **PLAINTEXT**или **SSL**в качестве транспортного уровня с использованием протокола безопасности **SASL\_PLAINTEXT**или **SASL\_SSL**соответственно. Если используется **SASL\_SSL**, необходимо также настроить **SSL**.

1. SASL-механизмы. RT.StreamingKafka поддерживает следующие механизмы SASL:

* GSSAPI (Kerberos).
* PLAIN.
* SCRAM-SHA-256.
* SCRAM-SHA-512.
* OAUTHBEARER.

2. Конфигурация SASL для брокеров.

* Настройте порт SASL в файле **server.properties**, добавив хотя бы один из **SASL\_PLAINTEXT**или **SASL\_SSL**к параметру слушателей, который содержит одно или несколько значений, разделённых запятыми:

listeners=SASL\_PLAINTEXT://host.name:port

Copy

* Если вы настраиваете только порт **SASL**(или хотите, чтобы брокеры RT.StreamingKafka аутентифицировали друг друга с помощью **SASL**), убедитесь, что вы установили один и тот же протокол **SASL**для связи между брокерами:

security.inter.broker.protocol=SASL\_PLAINTEXT (or SASL\_SSL)

Copy

* Выберите один или несколько поддерживаемых механизмов для включения в брокере и следуйте инструкциям по настройке SASL для этого механизма. При необходимости включите несколько механизмов в брокере.

3. Конфигурация SASL для клиентов.

Аутентификация SASL поддерживается только для нового поставщика и потребителя Java RT.StreamingKafka, старый API не поддерживается.

Чтобы настроить аутентификацию SASL на клиентах, выберите механизм SASL, включенный в брокере для аутентификации клиентов, и следуйте инструкциям по настройке SASL для выбранного механизма.

### 3.4.3 Аутентификация с использованием SASL/Kerberos

#### 3.4.3.1 Предварительные условия

1.**Kerberos**. Если ваша организация уже использует сервер Kerberos (например, с помощью Active Directory), нет необходимости устанавливать новый сервер только для RT.StreamingKafka. В противном случае вам придётся его установить. Скорее всего, у вашего поставщика Linux есть пакеты для Kerberos и краткое руководство по его установке и настройке ([Ubuntu](https://help.ubuntu.com/community/Kerberos), [Redhat](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Smart_Cards/installing-kerberos.html)). Обратите внимание: если вы используете Oracle Java, вам необходимо загрузить файлы политики JCE для вашей версии Java и скопировать их в **$JAVA\_HOME/jre/lib/security**.

2. **Создание принципалов Kerberos**. Если вы используете сервер Kerberos или Active Directory организации, попросите своего администратора Kerberos указать принципала для каждого брокера RT.StreamingKafka в вашем кластере и для каждого пользователя операционной системы, который будет получать доступ к RT.StreamingKafka с аутентификацией Kerberos (через клиенты и инструменты).

Если вы установили собственный Kerberos, вам нужно будет создать принципалов самостоятельно, используя следующие команды:

> sudo /usr/sbin/kadmin.local -q 'addprinc -randkey kafka/{hostname}@{REALM}'

> sudo /usr/sbin/kadmin.local -q "ktadd -k /etc/security/keytabs/{keytabname}.keytab kafka/{hostname}@{REALM}"

Copy

3. **Убедитесь, что все хосты доступны по именам хостов**. Требование Kerberos заключается в том, что все ваши хосты могут быть разрешены по их полным доменным именам.

#### 3.4.3.2 Настройка брокеров

1. Добавьте соответствующим образом модифицированный файл JAAS, аналогичный приведённому ниже, в каталог конфигурации каждого брокера RT.StreamingKafka, в этом примере назовем его **kafka\_server\_jaas.conf** (обратите внимание, что у каждого брокера должна быть своя собственная вкладка ключей):

KafkaServer {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/etc/security/keytabs/kafka\_server.keytab"

principal="kafka/kafka1.hostname.com@EXAMPLE.COM";

};

// Zookeeper client authentication

Client {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/etc/security/keytabs/kafka\_server.keytab"

principal="kafka/kafka1.hostname.com@EXAMPLE.COM";

};

Copy

Раздел **KafkaServer**в файле JAAS сообщает брокеру, какого принципала использовать, а также расположение таблицы ключей, где хранится этот принципал. Это позволяет брокеру войти в систему, используя таблицу ключей, указанную в этом разделе.

2. Передайте JAAS и, при необходимости, расположение файлов **krb5**в качестве параметров JVM каждому брокеру RT.StreamingKafka (более подробную информацию см. [здесь](https://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/KerberosReq.html)):

-Djava.security.krb5.conf=/etc/kafka/krb5.conf

-Djava.security.auth.login.config=/etc/kafka/kafka\_server\_jaas.conf

Copy

3. Убедитесь, что вкладки, настроенные в файле JAAS, доступны для чтения пользователю операционной системы, запускающему брокер RT.StreamingKafka.

4. Настройте порт SASL и механизмы SASL в **server.properties**. Например:

listeners=SASL\_PLAINTEXT://host.name:port

security.inter.broker.protocol=SASL\_PLAINTEXT

sasl.mechanism.inter.broker.protocol=GSSAPI

sasl.enabled.mechanisms=GSSAPI

Copy

Мы также должны настроить имя службы в **server.properties**, которое должно совпадать с основным именем брокеров RT.StreamingKafka. В приведённом выше примере принципалом является **kafka/kafka1.hostname.com@EXAMPLE.com**, поэтому:

sasl.kerberos.service.name=kafka

Copy

#### 3.4.3.3 Настройка клиентов

Чтобы настроить аутентификацию SASL на клиентах:

1. Клиенты (поставщики, потребители, воркеры и т.д.) будут аутентифицироваться в кластере со своим собственным принципалом (обычно с тем же именем, что и пользователь, запускающий клиент), поэтому заведите этих принципалов по мере необходимости. Затем настройте свойство конфигурации JAAS для каждого клиента. Разные клиенты внутри JVM могут работать как разные пользователи, если указать разных принципалов. Свойство **sasl.jaas.config** в **producer.properties** или **consumer.properties** описывает, как такие клиенты, как поставщик и потребитель, могут подключаться к Kafka Broker. Ниже приведён пример конфигурации для клиента, использующего вкладку ключей (рекомендуется для длительных процессов):

sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required \

useKeyTab=true \

storeKey=true \

keyTab="/etc/security/keytabs/kafka\_client.keytab" \

principal="kafka-client-1@EXAMPLE.COM";

Copy

Для утилит командной строки, таких как **kafka-console-consumer** или **kafka-console-producer**, **kinit**можно использовать вместе с **useTicketCache=true**, например:

sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required \

useTicketCache=true;

Copy

Конфигурацию JAAS для клиентов альтернативно можно указать как параметр JVM, аналогично брокерам, как описано выше. Клиенты используют раздел входа с именем **KafkaClient**. Эта опция разрешает только одному пользователю все клиентские подключения из JVM.

2. Убедитесь, что вкладки ключей, настроенные в конфигурации JAAS, доступны для чтения пользователю операционной системы, запускающему клиент RT.StreamingKafka.

3. При необходимости передайте расположение файлов **krb5**в качестве параметров JVM каждой клиентской JVM:

-Djava.security.krb5.conf=/etc/kafka/krb5.conf

Copy

4. Настройте следующие свойства в **producer.properties** или **consumer.properties**:

security.protocol=SASL\_PLAINTEXT (or SASL\_SSL)

sasl.mechanism=GSSAPI

sasl.kerberos.service.name=kafka

Copy

### 3.4.4 Аутентификация с использованием SASL/PLAIN

SASL/PLAIN — это простой механизм аутентификации по имени пользователя и паролю, который обычно используется с TLS для шифрования для имплементации безопасной аутентификации. RT.StreamingKafka поддерживает имплементацию SASL/PLAIN по умолчанию, которую можно расширить для производственного использования.

В имплементации по умолчанию **principal.builder.class** имя пользователя используется в качестве аутентифицированного **Principal**для настройки списков управления доступом (ACL) и т.д.

#### 3.4.4.1 Настройка брокеров

1. Добавьте соответствующим образом модифицированный файл JAAS, аналогичный приведённому ниже, в каталог конфигурации каждого брокера RT.StreamingKafka, в этом примере назовем его **kafka\_server\_jaas.conf**:

KafkaServer {

org.apache.kafka.common.security.plain.PlainLoginModule required

username="admin"

password="admin-secret"

user\_admin="admin-secret"

user\_alice="alice-secret";

};

Copy

Эта конфигурация определяет двух пользователей (**admin**и **alice**). Свойства **username**и **password**в разделе**KafkaServer** используются брокером для инициирования подключений к другим брокерам. В этом примере **admin**— это пользователь для межброкерской связи. Набор свойств **user\_userName** определяет пароли для всех пользователей, подключающихся к брокеру, и брокер проверяет все клиентские соединения, в том числе от других брокеров, используя эти свойства.

2. Передайте расположение файла конфигурации JAAS в качестве параметра JVM каждому брокеру RT.StreamingKafka:

-Djava.security.auth.login.config=/etc/kafka/kafka\_server\_jaas.conf

Copy

3. Настройте порт SASL и механизмы SASL в **server.properties**. Например:

listeners=SASL\_SSL://host.name:port

security.inter.broker.protocol=SASL\_SSL

sasl.mechanism.inter.broker.protocol=PLAIN

sasl.enabled.mechanisms=PLAIN

Copy

#### 3.4.4.2 Настройка клиентов

Чтобы настроить аутентификацию SASL на клиентах:

1. Настройте свойство конфигурации JAAS для каждого клиента в файле **producer.properties** или **consumer.properties**. Модуль входа описывает, как клиенты, такие как поставщик и потребитель, могут подключиться к RT.StreamingKafka Broker. Ниже приведён пример конфигурации клиента для механизма **PLAIN**:

sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \

username="alice" \

password="alice-secret";

Copy

Опции **username**и **password**используются клиентами для настройки пользователя для клиентских подключений. В этом примере клиенты подключаются к брокеру как пользователь **alice**. Разные клиенты внутри JVM могут подключаться как разные пользователи, если указать разные имена пользователей и пароли в **sasl.jaas.config**.

Конфигурацию JAAS для клиентов альтернативно можно указать как параметр JVM, аналогично брокерам. Клиенты используют раздел входа с именем **KafkaClient**. Эта опция разрешает только одному пользователю все клиентские подключения из JVM.

2. Настройте следующие свойства в файле **producer.properties** или **consumer.properties**:

security.protocol=SASL\_SSL

sasl.mechanism=PLAIN

Copy

#### 3.4.4.3 Использование SASL/PLAIN в производстве

1. SASL/PLAIN следует использовать только с SSL в качестве транспортного уровня, чтобы гарантировать, что jnrhsnst пароли не передаются по сети без шифрования.
2. Имплементация SASL/PLAIN в RT.StreamingKafka по умолчанию определяет имена пользователей и пароли в файле конфигурации JAAS. Начиная с Kafka версии 2.0, вы можете избежать хранения открытых паролей на диске, настроив собственные обработчики обратного вызова, которые получают имя пользователя и пароль из внешнего источника, используя параметры конфигурации **sasl.server.callback.handler.class** и **sasl.client.callback.handler.class**.
3. В производственных системах внешние серверы аутентификации могут реализовывать аутентификацию по паролю. Начиная с Kafka версии 2.0, вы можете подключить свои собственные обработчики обратного вызова, которые используют внешние серверы аутентификации для проверки пароля, настроив **sasl.server.callback.handler.class**.

### 3.4.5 Аутентификация с использованием SASL/SCRAM

Механизм хранения данных и протокол аутентификации посредством пароля (Salted Challenge Response Authentication Mechanism, SCRAM) — это семейство механизмов SASL, которое решает проблемы безопасности с помощью традиционных механизмов, выполняющих аутентификацию по имени пользователя и паролю, таких как **PLAIN**и **DIGEST-MD5**. Механизм определён в [RFC 5802](https://tools.ietf.org/html/rfc5802). RT.StreamingKafka поддерживает [**SCRAM-SHA-256**](https://tools.ietf.org/html/rfc7677) и **SCRAM-SHA-512**, которые можно использовать с TLS для выполнения безопасной аутентификации. В имплементации **principal.builder.class** по умолчанию имя пользователя используется в качестве аутентифицированного **Principal**для настройки списков ACL и т.д. Имплементация SCRAM по умолчанию в RT.StreamingKafka хранит учётные данные SCRAM в Zookeeper и подходит для использования в установках RT.StreamingKafka, где Zookeeper находится в частной сети.

#### 3.4.5.1 Создание учётных данных SCRAM

Реализация SCRAM в RT.StreamingKafka использует Zookeeper в качестве хранилища учётных данных. Учётные данные можно создать в Zookeeper с помощью **kafka-configs.sh**. Для каждого включённого механизма SCRAM необходимо создать учётные данные путём добавления конфигурации с именем механизма. Учётные данные для межброкерской связи должны быть созданы до запуска брокеров RT.StreamingKafka. Учётные данные клиента могут создаваться и обновляться динамически, а обновлённые учётные данные будут использоваться для аутентификации новых подключений.

Создайте учётные данные SCRAM для пользователя **alice**с паролем **alice-secret**:

> bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file zk\_tls\_config.properties --alter --add-config 'SCRAM-SHA-256=[iterations=8192,password=alice-secret],SCRAM-SHA-512=[password=alice-secret]' --entity-type users --entity-name alice

Copy

Если итерации не указаны, используется количество итераций по умолчанию, равное **4096**. Создаётся случайная соль, и идентификатор SCRAM, состоящий из соли, итераций, StoredKey и ServerKey, сохраняется в Zookeeper. Подробную информацию об идентификаторе SCRAM и отдельных полях см. в [RFC 5802](https://tools.ietf.org/html/rfc5802).

В следующих примерах также требуется пользователь **admin**для связи между брокерами, которого можно создать с помощью:

> bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file zk\_tls\_config.properties --alter --add-config 'SCRAM-SHA-256=[password=admin-secret],SCRAM-SHA-512=[password=admin-secret]' --entity-type users --entity-name admin

Copy

Существующие учётные данные могут быть перечислены с помощью опции **--describe**:

> bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file zk\_tls\_config.properties --describe --entity-type users --entity-name alice

Copy

Учётные данные можно удалить для одного или нескольких механизмов SCRAM с помощью опции **--alter --delete-config**:

> bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file zk\_tls\_config.properties --alter --delete-config 'SCRAM-SHA-512' --entity-type users --entity-name alice

Copy

#### 3.4.5.2 Настройка брокеров

1. Добавьте соответствующим образом модифицированный файл JAAS, аналогичный приведённому ниже, в каталог конфигурации каждого брокера RT.StreamingKafka, в этом примере назовем его **kafka\_server\_jaas.conf**:

KafkaServer {

org.apache.kafka.common.security.scram.ScramLoginModule required

username="admin"

password="admin-secret";

};

Copy

Свойства **username**и **password**в разделе **KafkaServer**используются брокером для инициирования подключений к другим брокерам. В этом примере **admin**— это пользователь для межброкерской связи.

2. Передайте расположение файла конфигурации JAAS в качестве параметра JVM каждому брокеру RT.StreamingKafka:

-Djava.security.auth.login.config=/etc/kafka/kafka\_server\_jaas.conf

Copy

3. Настройте порт SASL и механизмы SASL в **server.properties**. Например:

listeners=SASL\_SSL://host.name:port

security.inter.broker.protocol=SASL\_SSL

sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256 (or SCRAM-SHA-512)

sasl.enabled.mechanisms=SCRAM-SHA-256 (or SCRAM-SHA-512)

Copy

#### 3.4.5.3 Настройка клиентов

Чтобы настроить аутентификацию SASL на клиентах:

1. Настройте свойство конфигурации JAAS для каждого клиента в **producer.properties** или **consumer.properties**. Модуль входа описывает, как клиенты, такие как поставщик и потребитель, могут подключиться к Kafka Broker. Ниже приведён пример конфигурации клиента для механизмов SCRAM:

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="alice" \

password="alice-secret";

Copy

Опции **username**и **password** используются клиентами для настройки пользователя для клиентских подключений. В этом примере клиенты подключаются к брокеру как пользователь **alice**. Разные клиенты внутри JVM могут подключаться как разные пользователи, если указаны разные имена пользователей и пароли в **sasl.jaas.config**.

Конфигурацию JAAS для клиентов альтернативно можно указать как параметр JVM, аналогично брокерам. Клиенты используют раздел входа с именем **KafkaClient**. Эта опция разрешает только одному пользователю все клиентские подключения из JVM.

2. Настройте следующие свойства в **producer.properties** или **consumer.properties**:

security.protocol=SASL\_SSL

sasl.mechanism=SCRAM-SHA-256 (or SCRAM-SHA-512)

Copy

#### 3.4.5.4 Вопросы безопасности для SASL/SCRAM

1. Имплементация SASL/SCRAM по умолчанию в RT.StreamingKafka хранит учётные данные SCRAM в Zookeeper. Это подходит для производственного использования в установках, где Zookeeper безопасен и находится в частной сети.
2. RT.StreamingKafka поддерживает только надёжные хэш-функции SHA-256 и SHA-512 с минимальным количеством итераций 4096. Надёжные хэш-функции в сочетании с надёжными паролями и большим количеством итераций защищают от атак методом перебора, если безопасность Zookeeper скомпрометирована.
3. SCRAM следует использовать только с TLS-шифрованием, чтобы предотвратить перехват обмена SCRAM. Это защищает от атак по словарю или перебора, а также от выдачи себя за другое лицо, если Zookeeper скомпрометирован.
4. Начиная с Kafka версии 2.0, хранилище учётных данных SASL/SCRAM по умолчанию может быть переопределено с помощью кастомных обработчиков обратного вызова путём настройки **sasl.server.callback.handler.class** в установках, где Zookeeper небезопасен.
5. Более подробную информацию о вопросах безопасности можно найти в [RFC 5802](https://tools.ietf.org/html/rfc5802#section-9).

### 3.4.6 Аутентификация с использованием SASL/OAUTHBEARER

[Фреймворк авторизации OAuth 2](https://tools.ietf.org/html/rfc6749) “позволяет стороннему приложению получать ограниченный доступ к службе HTTP либо от имени владельца ресурса, организуя подтверждённое взаимодействие между владельцем ресурса и службой HTTP, либо разрешая стороннему приложению получить доступ от своего имени”. Механизм SASL OAUTHBEARER позволяет использовать фреймворк в контексте SASL (т.е. не-HTTP); он определен в [RFC 7628](https://tools.ietf.org/html/rfc7628). Имплементация OAUTHBEARER по умолчанию в RT.StreamingKafka создаёт и проверяет [незащищённые веб-токены JSON](https://tools.ietf.org/html/rfc7515#appendix-A.5) и подходит только для использования в непроизводственных установках RT.StreamingKafka.

В имплементации **principal.builder.class** по умолчанию **PrincipalName OAuthBearerToken** используется в качестве аутентифицированного **Principal**для настройки списков управления доступом (ACL) и т.д.

#### 3.4.6.1 Настройка брокеров

Добавьте соответствующим образом модифицированный файл JAAS, аналогичный приведённому ниже, в каталог конфигурации каждого брокера RT.StreamingKafka, в этом примере назовём его **kafka\_server\_jaas.conf**:

KafkaServer {

org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required

unsecuredLoginStringClaim\_sub="admin";

};

Copy

Свойство **unsecuredLoginStringClaim\_sub** в разделе **KafkaServer**используется брокером, когда он инициирует соединения с другими брокерами. В этом примере **admin**появится в утверждённом subject (sub) и будет пользователем для межброкерского взаимодействия.

2. Передайте расположение файла конфигурации JAAS в качестве параметра JVM каждому брокеру RT.StreamingKafka:

-Djava.security.auth.login.config=/etc/kafka/kafka\_server\_jaas.conf

Copy

3. Настройте порт SASL и механизмы SASL в **server.properties**. Например:

listeners=SASL\_SSL://host.name:port (or SASL\_PLAINTEXT if non-production)

security.inter.broker.protocol=SASL\_SSL (or SASL\_PLAINTEXT if non-production)

sasl.mechanism.inter.broker.protocol=OAUTHBEARER

sasl.enabled.mechanisms=OAUTHBEARER

Copy

#### 3.4.6.2 Настройка клиентов

Чтобы настроить аутентификацию SASL на клиентах:

1. Настройте свойство конфигурации JAAS для каждого клиента в **producer.properties** или **consumer.properties**. Модуль входа описывает, как клиенты, такие как поставщик и потребитель, могут подключиться к Kafka Broker. Ниже приведён пример конфигурации клиента для механизмов OAUTHBEARER:

sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \

unsecuredLoginStringClaim\_sub="alice";

Copy

Опция **unsecuredLoginStringClaim\_sub** используется клиентами для настройки утверждённого subject (sub), который определяет пользователя для клиентских подключений. В этом примере клиенты подключаются к брокеру как пользователь **alice**. Разные клиенты внутри JVM могут подключаться как разные пользователи, если указать разных subject (sub) в **sasl.jaas.config**.

Конфигурацию JAAS для клиентов альтернативно можно указать как параметр JVM, аналогично брокерам. Клиенты используют раздел входа с именем **KafkaClient**. Эта опция разрешает только одному пользователю все клиентские подключения из JVM.

2. Настройте следующие свойства в **producer.properties** или **consumer.properties**:

security.protocol=SASL\_SSL (or SASL\_PLAINTEXT if non-production)

sasl.mechanism=OAUTHBEARER

Copy

3. Имплементация **SASL/OAUTHBEARER** по умолчанию зависит от библиотеки **jackson-databind**. Поскольку это опциональная зависимость, пользователи должны настроить её как зависимость с помощью своего инструмента сборки.

#### 3.4.6.3 Варианты создания незащищённых токенов для SASL/OAUTHBEARER

1. Имплементация **SASL/OAUTHBEARER** по умолчанию в RT.StreamingKafka создаёт и проверяет [незащищённые веб-токены JSON](https://tools.ietf.org/html/rfc7515#appendix-A.5). Хотя она подходит только для непроизводственного использования, она обеспечивает гибкость для создания произвольных токенов в среде DEV или TEST.
2. Вот различные поддерживаемые параметры модуля JAAS на стороне клиента (и на стороне брокера, если OAUTHBEARER является межброкерским протоколом):

| **Опция модуля JAAS для создания незащищённых токенов** | **Описание** |
| --- | --- |
| unsecuredLoginStringClaim\_<claimname>="value" | Создаёт claim String с заданным именем и значением. Можно указать любое допустимое имя claim, кроме **iat**и **exp**(они генерируются автоматически). |
| unsecuredLoginNumberClaim\_<claimname>="value" | Создаёт claim Number с заданным именем и значением. Можно указать любое допустимое имя claim, кроме **iat**и **exp**(они генерируются автоматически). |
| unsecuredLoginListClaim\_<claimname>="value" | Создаёт claim String List с заданным именем и значениями, полученными из заданного значения, где первый символ используется в качестве разделителя. Например: **unsecuredLoginListClaim\_fubar="|value1|value2"**. Можно указать любое допустимое имя claim, кроме **iat**и **exp**(они генерируются автоматически). |
| unsecuredLoginExtension\_<extensionname>="value" | Создаёт String extension с заданным именем и значением. Например: **unsecuredLoginExtension\_traceId="123"**. Допустимым именем расширения является любая последовательность символов алфавита нижнего или верхнего регистра. Кроме того, имя расширения **auth**зарезервировано. Допустимым значением расширения является любая комбинация символов с кодами ASCII от 1 до 127. |
| unsecuredLoginPrincipalClaimName | Устанавливает имя кастомного claim, если вы хотите, чтобы имя claim String, содержащее основное имя, было отличным от **sub**. |
| unsecuredLoginLifetimeSeconds | Установите целочисленное значение, если срок действия токена должен быть установлен на значение, отличное от значения по умолчанию, равного 3600 секундам (что составляет 1 час). Claim **exp**будет установлено с учётом времени истечения срока действия. |
| unsecuredLoginScopeClaimName | Установите имя кастомного claim, если вы хотите, чтобы имя claim String или String List, содержащего любую область токена, было чем-то отличным от **scope**. |

#### 3.4.6.4 Варианты проверки незащищённого токена для SASL/OAUTHBEARER

1. Вот различные поддерживаемые варианты модуля JAAS на стороне брокера для проверки [незащищённого веб-токена JSON](https://tools.ietf.org/html/rfc7515#appendix-A.5):

| **Опция модуля JAAS для проверки незащищенных токенов** | **Описание** |
| --- | --- |
| unsecuredValidatorPrincipalClaimName="value" | Устанавливает непустое значение, если вы хотите, чтобы определённый claim String, содержащий имя принципала, проверялся на наличие; по умолчанию проверяется наличие claim **sub**. |
| unsecuredValidatorScopeClaimName="value" | Устанавливает имя кастомного claim, если вы хотите, чтобы имя claim String или String List, содержащее любую токен **scope**, было чем-то отличным от **scope**. |
| unsecuredValidatorRequiredScope="value" | Устанавливает список значений **scope**, разделённый пробелами, если вы хотите, чтобы claim String/String List, содержащий токен **scope**, было проверен, чтобы убедиться, что он содержит определённые значения. |
| unsecuredValidatorAllowableClockSkewMs="value" | Устанавливает положительное целое значение, если вы хотите допустить некоторое количество положительных миллисекунд отклонения часов (по умолчанию — **0**). |

2. Незащищенная имплементация **SASL/OAUTHBEARER** по умолчанию может быть переопределена (и должна быть переопределена в производственных средах) с использованием кастомного входа в систему и обработчиков обратного вызова сервера SASL.

3. Более подробную информацию о вопросах безопасности см. в [RFC 6749, раздел 10](https://tools.ietf.org/html/rfc6749#section-10).

#### 3.4.6.5 Обновление токена для SASL/OAUTHBEARER

RT.StreamingKafka периодически обновляет любой токен до истечения срока его действия, чтобы клиент мог продолжать подключаться к брокерам. Параметры, влияющие на работу алгоритма обновления, указываются как часть конфигурации поставщика/потребителя/брокера и заключаются в следующем. Подробности см. в разделах по этим свойствам. Значения по умолчанию обычно обоснованы, и в этом случае эти параметры конфигурации не нужно задавать явно.

* **sasl.login.refresh.window.factor**.
* **sasl.login.refresh.window.jitter**.
* **sasl.login.refresh.min.period.seconds**.
* **sasl.login.refresh.min.buffer.seconds**.

#### 3.4.6.6 Безопасное/производственное использование SASL/OAUTHBEARER

Сценарии производственного использования потребуют написания имплементации **org.apache.kafka.common.security.auth.AuthenticateCallbackHandler**, которая может обрабатывать инстанс **org.apache.kafka.common.security.oauthbearer.OAuthBearerTokenCallback**, и объявления его через параметр конфигурации sasl.login.callback.handler.class для клиента, не являющегося брокером, или через параметр конфигурации **listener.name.sasl\_ssl.oauthbearer.sasl.login.callback.handler.class** для брокеров (когда **SASL/OAUTHBEARER** является межброкерским протоколом).

Сценарии производственного использования также потребуют написания имплементации **org.apache.kafka.common.security.auth.AuthenticateCallbackHandler**, которая может обрабатывать инстанс **org.apache.kafka.common.security.oauthbearer.OAuthBearerValidatorCallback**, и объявления его через опцию конфигурации брокера **listener.name.sasl\_ssl.oauthbearer.sasl.server.callback.handler.class**.

#### 3.4.6.7 Вопросы безопасности для SASL/OAUTHBEARER

1. Имплементация **SASL/OAUTHBEARER** по умолчанию в RT.StreamingKafka создаёт и проверяет [незащищённые веб-токены JSON](https://tools.ietf.org/html/rfc7515#appendix-A.5). Подходит только для непроизводственного использования.
2. **OAUTHBEARER**следует использовать в производственных средах только с TLS-шифрованием для предотвращения перехвата токенов.
3. Незащищённая имплементация **SASL/OAUTHBEARER** по умолчанию может быть переопределена (и должна быть переопределена в производственных средах) с использованием кастомного входа в систему и обработчиков обратного вызова сервера SASL.
4. Более подробную информацию о вопросах безопасности OAuth 2 в целом можно найти в [RFC 6749, раздел 10](https://tools.ietf.org/html/rfc6749#section-10).

### 3.4.7 Включение нескольких механизмов SASL в брокере

1. Укажите конфигурацию модулей входа для всех включённых механизмов в разделе **KafkaServer**файла конфигурации JAAS. Например:

KafkaServer {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

storeKey=true

keyTab="/etc/security/keytabs/kafka\_server.keytab"

principal="kafka/kafka1.hostname.com@EXAMPLE.COM";

org.apache.kafka.common.security.plain.PlainLoginModule required

username="admin"

password="admin-secret"

user\_admin="admin-secret"

user\_alice="alice-secret";

};

Copy

2. Включите механизмы SASL в **server.properties**:

sasl.enabled.mechanisms=GSSAPI,PLAIN,SCRAM-SHA-256,SCRAM-SHA-512,OAUTHBEARER

Copy

3. При необходимости укажите протокол безопасности SASL и механизм межброкерской связи в **server.properties**:

security.inter.broker.protocol=SASL\_PLAINTEXT (or SASL\_SSL)

sasl.mechanism.inter.broker.protocol=GSSAPI (or one of the other enabled mechanisms)

Copy

4. Следуйте инструкциям, расположенным выше, для конкретного механизма в GSSAPI (Kerberos), PLAIN, SCRAM и OAUTHBEARER, чтобы настроить SASL для включённых механизмов.

### 3.4.8 Изменение механизма SASL в работающем кластере

Механизм SASL можно модифицировать в работающем кластере, используя следующую последовательность действий:

1. Включите новый механизм SASL, добавив **sasl.enabled.mechanisms** в **server.properties** для каждого брокера. Обновите файл конфигурации JAAS, включив в него оба механизма, как описано в п. 7.7. Постепенно перемещайте ноды кластера.
2. Перезапустите клиенты, используя новый механизм.
3. Чтобы изменить механизм взаимодействия между брокерами (если это необходимо), установите для **sasl.mechanism.inter.broker.protocol** в **server.properties** новый механизм и снова постепенно выполните возврат кластера.
4. Чтобы удалить старый механизм (если это необходимо), удалите старый механизм из **sasl.enabled.mechanisms** в **server.properties**и удалите записи для старого механизма из файла конфигурации JAAS. Снова постепенно выполните возврат кластер.

### 3.4.9 Аутентификация с использованием токенов делегирования

Аутентификация на основе токена делегирования — это упрощённый механизм аутентификации, дополняющий существующие методы SASL/SSL. Токены делегирования — это общие секреты брокеров и клиентов RT.StreamingKafka. Токены делегирования помогут фреймворкам обработки распределить рабочую нагрузку между доступными воркерами в безопасной среде без дополнительной нагрузки на распространение Kerberos TGT/keytabs или хранилищ ключей при использовании двустороннего SSL. Более подробную информацию см. в [KIP-48](https://cwiki.apache.org/confluence/display/KAFKA/KIP-48+Delegation+token+support+for+Kafka).

При имплементации **principal.builder.class** по умолчанию владелец токена делегирования используется в качестве аутентифицированного **Principal**для настройки списков управления доступом (ACL) и т.д.

Типичные шаги по использованию токена делегирования:

1. Пользователь проходит аутентификацию в кластере RT.StreamingKafka через SASL или SSL и получает токен делегирования. Это можно сделать с помощью API администратора или скрипта **kafka-delegation-tokens.sh**.
2. Пользователь безопасно передаёт токен делегирования клиентам RT.StreamingKafka для аутентификации в кластере RT.StreamingKafka.
3. Владелец/обновляющий токен может продлить/не продлить токены делегирования.

#### 3.4.9.1 Управление токенами

Секрет используется для создания и валидации токенов делегирования. Это предоставляется с использованием параметра конфигурации **delegation.token.secret.key**. Один и тот же секретный ключ должен быть настроен для всех брокеров. Если секрет не установлен или имеет пустую строку, брокеры отключат аутентификацию токена делегирования.

В текущей имплементации данные токена хранятся в Zookeeper и подходят для использования в установках RT.StreamingKafka, где Zookeeper находится в частной сети. Также в настоящее время этот секрет хранится в виде открытого текста в файле конфигурации **server.properties** (они будут сделаны настраиваемыми в будущем выпуске RT.StreamingKafka).

Токен имеет текущий срок жизни и максимальный возобновляемый срок действия. По умолчанию токены необходимо обновлять каждые 24 часа на срок до 7 дней. Их можно настроить с помощью параметров конфигурации **delegation.token.expiry.time.ms** и **delegation.token.max.lifetime.ms**.

Токены также могут быть отменены явно. Если токен не продлевается по истечении срока действия токена или если срок жизни токена превышает максимальное, он будет удалён из всех кэшей брокера, а также из Zookeeper.

#### 3.4.9.2 Создание токенов делегирования

Токены можно создавать с помощью API администратора или скрипта **kafka-delegation-tokens.sh**. Запросы токена делегирования (создание/обновление/истечение срока действия/описание) должны отправляться только по аутентифицированным каналам SASL или SSL. Токены не могут быть запрошены, если инициирующая аутентификация выполнена с помощью токена делегирования. Токен может быть создан пользователем для него самого или других пользователей с помощью параметра **--owner-principal**. Владелец/обновляющие могут продлевать токены или истекать срок их действия. Владельцы/обновляющие всегда могут описать свои токены. Для описания других токенов необходимо добавить разрешение **DESCRIBE\_TOKEN** к ресурсу пользователя, представляющему владельца токена. Примеры скриптов **kafka-delegation-tokens.sh** приведены ниже.

Создайте токен делегирования:

> bin/kafka-delegation-tokens.sh --bootstrap-server localhost:9092 --create --max-life-time-period -1 --command-config client.properties --renewer-principal User:user1

Copy

Создайте токен делегирования для другого владельца:

> bin/kafka-delegation-tokens.sh --bootstrap-server localhost:9092 --create --max-life-time-period -1 --command-config client.properties --renewer-principal User:user1 --owner-principal User:owner1

Copy

Обновите токен делегирования:

> bin/kafka-delegation-tokens.sh --bootstrap-server localhost:9092 --renew --renew-time-period -1 --command-config client.properties --hmac ABCDEFGHIJK

Copy

Истеките срок действия токена делегирования:

> bin/kafka-delegation-tokens.sh --bootstrap-server localhost:9092 --expire --expiry-time-period -1 --command-config client.properties --hmac ABCDEFGHIJK

Copy

Опишите существующие токены с помощью опции **--describe**:

> bin/kafka-delegation-tokens.sh --bootstrap-server localhost:9092 --describe --command-config client.properties --owner-principal User:user1

Copy

#### 3.4.9.3 Аутентификация токена

Аутентификация токена делегирования сочетается с текущим механизмом аутентификации SASL/SCRAM. Мы должны включить механизм SASL/SCRAM в кластере RT.StreamingKafka.

Конфигурирование клиентов RT.StreamingKafka:

1. Сконфигурируйте свойство конфигурации JAAS для каждого клиента в **producer.properties** или **consumer.properties**. Модуль входа описывает, как клиенты, такие как поставщик и потребитель, могут подключиться к RT.StreamingKafka Broker. Ниже приведён пример конфигурации клиента для аутентификации по токену:

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="tokenID123" \

password="lAYYSFmLs4bTjf+lTZ1LCHR/ZZFNA==" \

tokenauth="true";

Copy

Параметры **username**и **password**используются клиентами для настройки идентификатора и HMAC (hash-based message authentication code) токена. Опция **tokenauth**используется для сообщения серверу об аутентификации по токену. В этом примере клиенты подключаются к брокеру, используя идентификатор токена **tokenID123**. Разные клиенты внутри JVM могут подключаться с использованием разных токенов, указав разные сведения о токенах в **sasl.jaas.config**.

Конфигурацию JAAS для клиентов альтернативно можно указать как параметр JVM, аналогично брокерам. Клиенты используют раздел входа с именем **KafkaClient**. Эта опция разрешает только одного пользователя для установления всех клиентских подключений из JVM.

#### 3.4.9.4 Процедура ротации секрета вручную

Нам требуется повторное развёртывание, если необходимо сменить секрет. Во время этого процесса уже подключённые клиенты продолжат работать. Но любые новые запросы на подключение и запросы на продление/истечение срока действия со старыми токенами могут завершиться неудачей. Шаги приведены ниже.

1. Истеките срок действия всех существующих токенов.
2. Поменяйте секрет, последовательно обновив его.
3. Сгенерируйте новые токены.

Вероятно в новой версии RT.StreamingKafka эта процедура будет автоматизирована.

## 3.5 Авторизация и списки ACL

RT.StreamingKafka поставляется с подключаемым фреймворком авторизации, которой конфигурируется с помощью свойства **authorizer.class.name** в конфигурации сервера. Конфигурируемые имплементации должны расширять **org.apache.kafka.server.authorizer.Authorizer**. RT.StreamingKafka предоставляет имплементации по умолчанию, которые хранят списки ACL в метаданных кластера (либо Zookeeper, либо логи метаданных KRaft). Для кластеров на базе Zookeeper предоставленная имплементация настраивается следующим образом:

authorizer.class.name=kafka.security.authorizer.AclAuthorizer

Copy

Для кластеров KRaft используйте следующую конфигурацию на всех нодах (брокерах, контроллерах или комбинированных нодах брокера и контроллера):

authorizer.class.name=org.apache.kafka.metadata.authorizer.StandardAuthorizer

Copy

Списки ACL RT.StreamingKafka определяются в общем формате «Principal {P} is [Allowed|Denied] Operation {O} From Host {H} on any Resource {R} matching ResourcePattern {RP}». Подробнее о структуре ACL можно прочитать в [KIP-11](https://cwiki.apache.org/confluence/display/KAFKA/KIP-11+-+Authorization+Interface) и шаблонах ресурсов в [KIP-290](https://cwiki.apache.org/confluence/display/KAFKA/KIP-290%3A+Support+for+Prefixed+ACLs). Чтобы добавить, удалить или составить список ACL, вы можете использовать CLI ACL RT.StreamingKafka **kafka-acls.sh**. По умолчанию, если ни один **ResourcePatterns**не соответствует конкретному **Resource**, то у **Resource** нет связанных с ним ACL, и поэтому никому, кроме суперпользователей, не разрешён доступ к **Resource**. Если вы хотите изменить это поведение, вы можете включить следующее в **server.properties** .

allow.everyone.if.no.acl.found=true

Copy

Также можно добавить суперпользователей в **server.properties**, как показано ниже (обратите внимание, что разделителем является точка с запятой, поскольку имена пользователей SSL могут содержать запятую).

super.users=User:Bob;User:Alice

Copy

### 3.5.1 Отправка принципалов KRaft

В кластерах KRaft запросы администратора, такие как **CreateTopics**и **DeleteTopics**, отправляются клиентом слушателям брокера. Затем брокер пересылает запрос активному контроллеру через первого слушателя, настроенного в **controller.listener.names**. Авторизация таких запросов выполняется на ноде контроллера. Это достигается с помощью запроса **Envelope**, который упаковывает как базовый запрос от клиента, так и принципала клиента. Когда контроллер получает перенаправленный запрос **Envelope**от брокера, он сначала авторизует запрос **Envelope**, используя аутентифицированного принципала брокера. Затем он авторизует базовый запрос с использованием перенаправленного принципала.

Все это означает, что RT.StreamingKafka должен понимать, как сериализовать и десериализовать клиентского принципала. Фреймворк аутентификации позволяет кастомизировать принципалов, переопределяя конфигурацию **principal.builder.class**. Чтобы кастомизированные принципалы могли работать с KRaft, настроенный класс должен имплементировать **org.apache.kafka.common.security.auth.KafkaPrincipalSerde**, поэтому RT.StreamingKafka будет знать, как сериализовать и десериализовать принципалов. Имплементация по умолчанию **org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalBuilder** использует формат RPC Kafka, определённый в исходном коде: **clients/src/main/resources/common/message/DefaultPrincipalData.json**. Подробнее о переадресации запросов в KRaft см. [KIP-590](https://cwiki.apache.org/confluence/display/KAFKA/KIP-590%3A+Redirect+Zookeeper+Mutation+Protocols+to+The+Controller).

### 3.5.2 Кастомизация имени пользователя SSL

По умолчанию имя пользователя SSL будет иметь вид **"CN=writeuser,OU=Unknown,O=Unknown,L=Unknown,ST=Unknown,C=Unknown"**. Его можно изменить, установив настройку **ssl.principal.mapping.rules** для индивидуального правила в **server.properties**. Эта конфигурация позволяет использовать список правил для маппинга отличительного имени X.500 с коротким именем. Правила оцениваются по порядку, и первое правило, соответствующее отличительному имени, используется для его маппинга с коротким именем. Любые последующие правила в списке игнорируются.

Формат **ssl.principal.mapping.rules** представляет собой список, в котором каждое правило начинается с **"RULE:"** и содержит выражение в следующих форматах. По умолчанию правило возвращает строковое представление отличительного имени X.500. Если отличительное имя соответствует шаблону, то для имени будет выполнена команда замены. Оно также поддерживает параметры нижнего/верхнего регистра, чтобы перевести результат в нижний/верхний регистр. Это делается путём добавления **"/L"** or **"/U"** в конец правила.

RULE:pattern/replacement/

RULE:pattern/replacement/[LU]

Copy

Примеры значений **ssl.principal.mapping.rules**:

RULE:^CN=(.\*?),OU=ServiceUsers.\*$/$1/,

RULE:^CN=(.\*?),OU=(.\*?),O=(.\*?),L=(.\*?),ST=(.\*?),C=(.\*?)$/$1@$2/L,

RULE:^.\*[Cc][Nn]=([a-zA-Z0-9.]\*).\*$/$1/L,

DEFAULT

Copy

Приведённые выше правила преобразуют отличительное имя **"CN=serviceuser,OU=ServiceUsers,O=Unknown,L=Unknown,ST=Unknown,C=Unknown"** в **"serviceuser"** и **"CN=adminUser,OU=Admin,O=Unknown,L=Unknown,ST=Unknown,C=Unknown"** на **"adminuser@admin"**.

В расширенных случаях использования можно кастомизировать имя, установив кастомизированное **PrincipalBuilder**в **server.properties**, как показано ниже.

principal.builder.class=CustomizedPrincipalBuilderClass

Copy

### 3.5.3 Кастомизация имени пользователя SASL

По умолчанию имя пользователя SASL будет основной частью принципала Kerberos. Его можно изменить, установив **sasl.kerberos.principal.to.local.rules** для индивидуального правила в **server.properties**. Формат **sasl.kerberos.principal.to.local.rules** представляет собой список, в котором каждое правило работает так же, как **auth\_to\_local** в [файле конфигурации Kerberos (krb5.conf)](http://web.mit.edu/Kerberos/krb5-latest/doc/admin/conf_files/krb5_conf.html). Также поддерживается дополнительное правило нижнего/прописного регистра, позволяющее переводить результат в нижний/верхний регистр. Это делается путём добавления **"/L"** or **"/U"** в конец правила. Изучите синтаксис форматов ниже. Каждое правило начинается с **RULE:** и содержит выражение в следующих форматах. Дополнительные сведения см. в документации Kerberos.

RULE:[n:string](regexp)s/pattern/replacement/

RULE:[n:string](regexp)s/pattern/replacement/g

RULE:[n:string](regexp)s/pattern/replacement//L

RULE:[n:string](regexp)s/pattern/replacement/g/L

RULE:[n:string](regexp)s/pattern/replacement//U

RULE:[n:string](regexp)s/pattern/replacement/g/U

Copy

Пример добавления правила для правильного перевода **user@MYDOMAIN.COM** в пользователя с сохранением правила по умолчанию:

sasl.kerberos.principal.to.local.rules=RULE:[1:$1@$0](.\*@MYDOMAIN.COM)s/@.\*//,DEFAULT

Copy

### 3.5.4 Интерфейс командной строки (CLI)

CLI управления авторизацией RT.StreamingKafka можно найти в каталоге **bin**вместе со всеми другими CLI. Скрипт CLI называется **kafka-acls.sh**. Ниже перечислены все параметры, которые поддерживает скрипт:

| **Параметр** | **Описание** | **Значение по умолчанию** | **Тип параметра** |
| --- | --- | --- | --- |
| **--add** | Указывает скрипту, что пользователь пытается добавить ACL. | -- | Action |
| **--remove** | Указывает скрипту, что пользователь пытается удалить ACL. | -- | Action |
| **--list** | Указывает скрипту, что пользователь пытается вывести листинг ACL. | -- | Action |
| **--bootstrap-server** | Список пар хост/порт, которые будут использоваться для установления соединения с кластером RT.StreamingKafka. Должен быть указан только один из параметров**--bootstrap-server** или**--authorizer**. | -- | Configuration |
| **--command-config** | Файл свойств, содержащий конфигурации, которые будут переданы в Admin Client. Этот параметр можно использовать только с параметром**--bootstrap-server**. | -- | Configuration |
| **--cluster** | Указывает скрипту, что пользователь пытается взаимодействовать с ACL на отдельном ресурсе кластера. | -- | ResourcePattern |
| **--topic *[topic-name]*** | Указывает скрипту, что пользователь пытается взаимодействовать с ACL по шаблонам ресурсов топика. | -- | ResourcePattern |
| **--group *[group-name]*** | Указывает скрипту, что пользователь пытается взаимодействовать с ACL по шаблонам ресурсов группы потребителей. | -- | ResourcePattern |
| **--transactional-id *[transactional-id]*** | Идентификатор транзакции, к которому следует добавить или удалить ACL. Значение **\*** указывает, что ACL должны применяться ко всем идентификаторам транзакций. | -- | ResourcePattern |
| **--delegation-token *[delegation-token]*** | Токен делегирования, к которому следует добавить или удалить ACL. Значение **\*** указывает, что ACL должен применяться ко всем токенам. | -- | ResourcePattern |
| **--user-principal *[user-principal]*** | Пользовательский ресурс, к которому следует добавить или удалить ACL. В настоящее время поддерживается в отношении токенов делегирования. Значение **\*** указывает, что ACL должен применяться ко всем пользователям. | -- | ResourcePattern |
| **--resource-pattern-type *[pattern-type]*** | Указывает скрипту тип шаблона ресурса (для **--add**) или фильтра шаблона ресурса (для **--list** и **--remove**), который пользователь хочет использовать.  При добавлении ACL это должен быть определённый тип шаблона, например, **literal**или **prefixed**.  При листинге или удалении ACL можно использовать фильтр определённого типа шаблона для перечисления или удаления ACL из определённого типа шаблона ресурса или можно использовать значения фильтра **any**или **match**, где **any**будет соответствовать любому типу шаблона, но будет точно соответствовать имени ресурса, а **match** выполнит сопоставление с шаблоном, чтобы перечислить или удалить все ACL, которые влияют на предоставленный ресурс(ы).  **Внимание**: ключ **match**при использовании в сочетании с ключом **--remove** следует использовать с осторожностью. | literal | Configuration |
| **--allow-principal** | Принципал имеет формат **PrincipalType:name**, который будет добавлен в ACL с разрешением **Allow**. Строка **PrincipalType**по умолчанию **User**чувствительна к регистру.  Вы можете указать несколько**--allow-principal** в одной команде. | -- | Principal |
| **--deny-principal** | Принципал имеет формат **PrincipalType:name**, который будет добавлен в ACL с разрешением **Deny**. Строка **PrincipalType**по умолчанию **User** чувствительна к регистру.  Вы можете указать несколько **--deny-principal**в одной команде. | -- | Principal |
| **--principal** | Принципал имеет формат **PrincipalType:name**, который будет использоваться вместе с параметром **--list**. Строка **PrincipalType**по умолчанию **User** чувствительна к регистру. В результате будут перечислены списки ACL для указанного принципала.  Вы можете указать несколько **--principal** в одной команде. | -- | Principal |
| **--allow-host** | IP-адрес, с которого принципалы, перечисленные в **--allow-principal**, будут иметь доступ. | Если указан **--allow-principal**, по умолчанию используется **\***, что означает “все хосты”. | Host |
| **--deny-host** | IP-адрес, с которого принципалам, перечисленным в **--deny-principal**, будет отказано в доступе. | Если указан **--deny-principal**, по умолчанию используется **\***, что означает “все хосты”. | Host |
| **--operation** | Операция, которая будет разрешена или запрещена.  Допустимые значения:   * **Read**; * **Write**; * **Create**; * **Delete**; * **Alter**; * **Describe**; * **ClusterAction**; * **DescribeConfigs**; * **AlterConfigs**; * **IdempotentWrite**; * **CreateTokens**; * **DescribeTokens**; * **All**. | All | Operation |
| **--producer** | Удобная возможность добавления/удаления ACL для роли поставщика. Будут созданы списки ACL, позволяющие **WRITE**, **DESCRIBE**и **CREATE**топики. | -- | Convenience |
| **--consumer** | Удобная возможность добавления/удаления ACL для роли потребителя. Будут созданы списки ACL, которые позволят **WRITE**, **DESCRIBE**топики и **CREATE** в группе потребителей. | -- | Convenience |
| **--idempotent** | Включает идемпотентность для поставщика. Следует использовать в сочетании с параметром **--producer**.  Обратите внимание, что идемпотентность включается автоматически, если поставщик авторизован для определённого **transactional-id**. | -- | Convenience |
| **--force** | Удобный параметр: предполагать **yes**на все запросы и не задавать вопросы. | -- | Convenience |
| **--authorizer** | [**Устарело**: не поддерживается в KRaft]. Полное имя класса авторизатора. | kafka.security.authorizer.AclAuthorizer | Configuration |
| **--authorizer-properties** | [**Устарело**: не поддерживается в KRaft]. Пары ключ=значение, которые будут переданы авторизатору для инициализации. Для авторизатора по умолчанию в кластерах ZK пример значений: **zookeeper.connect=localhost:2181**. | -- | Configuration |
| **--zk-tls-config-file** | [**Устарело**: не поддерживается в KRaft]. Идентифицирует файл, в котором определяются свойства подключения TLS клиента ZooKeeper для авторизатора. Любые свойства, кроме следующих (с префиксом **authorizer.** или без него), игнорируются: **zookeeper.clientCnxnSocket**, **zookeeper.ssl.cipher.suites**, **zookeeper.ssl.client.enable**, **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**. | -- | Configuration |

### 3.5.5 Примеры

1. **Добавление ACL**.

Предположим, вы хотите добавить ACL "Principals User:Bob and User:Alice are allowed to perform Operation Read and Write on Topic Test-Topic from IP 198.51.100.0 and IP 198.51.100.1". Вы можете сделать это, запустив CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:Bob --allow-principal User:Alice --allow-host 198.51.100.0 --allow-host 198.51.100.1 --operation Read --operation Write --topic Test-topic

Copy

По умолчанию всем принципалам, у которых нет явного ACL, разрешающего доступ операции к ресурсу, запрещается. В редких случаях, когда определён разрешённый ACL, который разрешает доступ ко всем принципалам, кроме некоторых, нам придётся использовать параметры **--deny-principal** и **--deny-host**. Например, если мы хотим разрешить всем пользователям читать (**Read**) из топика **Test-topic**, но запретить **User:BadBob** только с IP 198.51.100.3, мы можем сделать это с помощью следующих команд:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:'\*' --allow-host '\*' --deny-principal User:BadBob --deny-host 198.51.100.3 --operation Read --topic Test-topic

Copy

Обратите внимание, что **--allow-host**и **--deny-host** поддерживают только IP-адреса (имена хостов не поддерживаются). В приведённых выше примерах можно добавить ACL к топику, указав **--topic *[имя-топика]*** в качестве параметра шаблона ресурса. Аналогичным образом пользователь может добавить ACL в кластер, указав **--cluster,** и в группу потребителей, указав **--group *[имя-группы]***. Вы можете добавить ACL на любой ресурс определённого типа, например. Предположим, вы хотите добавить ACL "Principal User:Peter is allowed to produce to any Topic from IP 198.51.200.0". Вы можете сделать это, используя ресурс подстановочного знака **\***, например выполнив CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:Peter --allow-host 198.51.200.1 --producer --topic '\*'

Copy

Вы можете добавить ACL к шаблонам ресурсов с префиксами, например. Предположим, вы хотите добавить ACL "Principal User:Jane is allowed to produce to any Topic whose name starts with 'Test-' from any host". Вы можете сделать это, запустив CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:Jane --producer --topic Test- --resource-pattern-type prefixed

Copy

Обратите внимание, что **--resource-pattern-type** по умолчанию имеет значение **'literal'**, что влияет только на ресурсы с точно таким же именем или, в случае имени ресурса с подстановочным знаком **\***, на ресурс с любым именем.

2. **Удаление ACL**.

Удаление ACL практически то же самое. Единственное отличие заключается в том, что вместо параметра **--add** пользователям придётся указать параметр **--remove**. Чтобы удалить ACL, добавленные в первом примере выше, мы можем запустить CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --remove --allow-principal User:Bob --allow-principal User:Alice --allow-host 198.51.100.0 --allow-host 198.51.100.1 --operation Read --operation Write --topic Test-topic

Copy

Если вы хотите удалить ACL, добавленный к шаблону ресурса с префиксом выше, мы можем запустить CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --remove --allow-principal User:Jane --producer --topic Test- --resource-pattern-type Prefixed

Copy

3. **Вывод листинга ACL**.

Мы можем вывести листинг ACL для любого ресурса, указав для ресурса параметр **--list**. Чтобы вывести список всех ACL в шаблоне ресурсов **Test-topic**, мы можем запустить CLI со следующими параметрами:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --list --topic Test-topic

Copy

Однако при этом будут возвращены только те ACL, которые были добавлены именно к этому шаблону ресурсов. Могут существовать и другие ACL, влияющие на доступ к топику, например. любые ACL для подстановочного знака топика **\*** или любые ACL для шаблонов ресурсов с префиксом. ACL для шаблона ресурса с подстановочными знаками можно запросить явно:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --list --topic '\*'

Copy

Однако не всегда возможно явно запросить ACL для шаблонов ресурсов с префиксами, соответствующих топику **Test-topic**, поскольку имена таких шаблонов могут быть неизвестны. Мы можем перечислить все ACL, влияющие на топик **Test-topic**, используя **'--resource-pattern-type match'**, например

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --list --topic Test-topic --resource-pattern-type match

Copy

При этом будет выведен листинг ACL для всех соответствующих шаблонов ресурсов с литералами, подстановочными знаками и префиксами.

4. **Добавление или удаление принципала в качестве поставщика или потребителя**.

Наиболее распространённым вариантом использования управления ACL является добавление или удаление принципала в качестве поставщика или потребителя, поэтому были добавлены удобные параметры для обработки этих случаев. Чтобы добавить **User:Bob** в качестве поставщика топика **Test-topic**, мы можем выполнить следующую команду:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:Bob --producer --topic Test-topic

Copy

Аналогично, чтобы добавить **Alice**в качестве потребителя топика **Test-topic** с группой потребителей **Group-1**, нам просто нужно передать параметр**--consumer**:

> bin/kafka-acls.sh --bootstrap-server localhost:9092 --add --allow-principal User:Bob --consumer --topic Test-topic --group Group-1

Copy

Обратите внимание, что для параметра потребителя мы также должны указать группу потребителей. Чтобы удалить принципала из роли поставщика или потребителя, нам просто нужно передать параметр **--remove**.

5. **Управление ACL на основе API Admin**.

Пользователи, имеющие разрешение на изменение в ClusterResource, могут использовать API Admin для управления ACL. Скрипт **kafka-acls.sh** поддерживает API AdminClient для управления списками управления доступом без прямого взаимодействия с Zookeeper/Authorizer. Все приведённые выше примеры можно выполнить, используя параметр **--bootstrap-server**. Например:

bin/kafka-acls.sh --bootstrap-server localhost:9092 --command-config /tmp/adminclient-configs.conf --add --allow-principal User:Bob --producer --topic Test-topic

bin/kafka-acls.sh --bootstrap-server localhost:9092 --command-config /tmp/adminclient-configs.conf --add --allow-principal User:Bob --consumer --topic Test-topic --group Group-1

bin/kafka-acls.sh --bootstrap-server localhost:9092 --command-config /tmp/adminclient-configs.conf --list --topic Test-topic

bin/kafka-acls.sh --bootstrap-server localhost:9092 --command-config /tmp/adminclient-configs.conf --add --allow-principal User:tokenRequester --operation CreateTokens --user-principal "owner1"

Copy

### 3.5.6 Примитивы авторизации

Вызовы протокола обычно выполняют некоторые операции с определёнными ресурсами в RT.StreamingKafka. Для настройки эффективной защиты необходимо знать операции и ресурсы. В этом разделе перечислены эти операции и ресурсы, а затем их комбинации с протоколами, чтобы увидеть допустимые скрипты.

#### 3.5.6.1 Операции

Существует несколько примитивов операций, которые можно использовать для создания привилегий. Их можно сопоставить с определёнными ресурсами, чтобы разрешить вызовы определённых протоколов для данного пользователя. Это:

* **Read**(чтение);
* **Write**(запись);
* **Create**(создание);
* **Delete**(удаление);
* **Alter**(изменение);
* **Describe**(описание);
* **ClusterAction**(действие над кластером);
* **DescribeConfigs**(описание конфигураций);
* **AlterConfigs**(изменение конфигураций);
* **IdempotentWrite**(запись идемпотентности);
* **CreateTokens**(создание токенов);
* **DescribeTokens**(описание токенов);
* **All**(все привилегии).

#### 3.5.6.2 Ресурсы

Вышеописанные операции можно применять к определённым ресурсам, которые описаны ниже.

1. **Topik**-- это просто представляет топик. Все вызовы протокола, которые действуют на топики (например, их чтение, запись), требуют добавления соответствующей привилегии. Если произошла ошибка авторизации с ресурсом топика, будет возвращено **TOPIC\_AUTHORIZATION\_FAILED (error code: 29)**.
2. **Group**-- представляет группы потребителей у брокеров. Все вызовы протокола, которые работают с группами потребителей, например, присоединение к группе, должны иметь привилегии для указанной группы. Если привилегия не предоставлена, в ответе протокола будет возвращено **GROUP\_AUTHORIZATION\_FAILED (error code: 30)**.
3. **Cluster**-- этот ресурс представляет кластер. Операции, влияющие на весь кластер, например управляемое завершение работы, защищены привилегиями ресурса кластера. Если на ресурсе кластера возникла проблема с авторизацией, будет возвращено сообщение **CLUSTER\_AUTHORIZATION\_FAILED (error code: 31)**.
4. **TransactionalId**-- этот ресурс представляет действия, связанные с транзакциями, например коммит. Если произойдет какая-либо ошибка, брокеры вернут **TRANSACTIONAL\_ID\_AUTHORIZATION\_FAILED (error code: 53)**.
5. **DelegationToken**-- представляет токены делегирования в кластере. Такие действия, как описание токенов делегирования, могут быть защищены привилегией ресурса **DelegationToken**. Поскольку эти объекты ведут себя в RT.StreamingKafka не так, как все, рекомендуется прочитать [KIP-48](https://cwiki.apache.org/confluence/display/KAFKA/KIP-48+Delegation+token+support+for+Kafka#KIP-48DelegationtokensupportforKafka-DescribeDelegationTokenRequest) и **п. 3.4.9**.
6. **User**-- операции **CreateToken**и **DescribeToken**могут быть предоставлены ресурсам пользователя, чтобы позволить создавать и описывать токены для других пользователей. Более подробную информацию можно найти в [KIP-373](https://cwiki.apache.org/confluence/display/KAFKA/KIP-373%3A+Allow+users+to+create+delegation+tokens+for+other+users).

#### 3.5.6.3 Операции и ресурсы по протоколам

В таблице ниже перечислены допустимые операции с ресурсами, которые выполняются протоколами API RT.StreamingKafka.

| **Протокол (ключ API)** | **Операция** | **Ресурс** | **Примечание** |
| --- | --- | --- | --- |
| PRODUCE (0) | Write | TransactionalId | Эта привилегия требуется поставщику транзакций, у которого установлен **transactional.id**. |
| PRODUCE (0) | IdempotentWrite | Cluster | Идемпотентное действие поставщика требует этой привилегии. |
| PRODUCE (0) | Write | Topic | Применяется к обычному действию поставщика. |
| FETCH (1) | ClusterAction | Cluster | Для получения данных партиции подписчик должен иметь **ClusterAction**на ресурсе **Cluster**. |
| FETCH (1) | Read | Topic | Обычным потребителям RT.StreamingKafka необходимо разрешение **Read**для каждой партиции, которую они извлекают. |
| LIST\_OFFSETS (2) | Describe | Topic | -- |
| METADATA (3) | Describe | Topic | -- |
| METADATA (3) | Create | Cluster | Если автоматическое создание топика включено, API на стороне брокера проверит наличие привилегий уровня **Cluster**. Если такие привилегии найдены, он позволит создать топик, в противном случае он будет перебирать привилегии уровня **Topic**(см. следующий). |
| METADATA (3) | Create | Topic | Разрешает автоматическое создание топика, если оно включено, но у данного пользователя нет разрешения на уровне **Cluster**(см. выше). |
| LEADER\_AND\_ISR (4) | ClusterAction | Cluster | -- |
| STOP\_REPLICA (5) | ClusterAction | Cluster | -- |
| UPDATE\_METADATA (6) | ClusterAction | Cluster | -- |
| CONTROLLED\_SHUTDOWN (7) | ClusterAction | Cluster | -- |
| OFFSET\_COMMIT (8) | Read | Group | Коммит смещения возможен только в том случае, если оно разрешено для данной группы и топика (см. ниже). Сначала проверяется групповой доступ, затем доступ к топику. |
| OFFSET\_COMMIT (8) | Read | Topic | Поскольку коммит смещения является частью процесса потребителей, им необходимы привилегии для действия чтения. |
| OFFSET\_FETCH (9) | Describe | Group | Как и в случае с **OFFSET\_COMMIT**, приложение должно иметь привилегии на уровне группы и топика, чтобы иметь возможность выполнять выборку. Однако в этом случае требуется доступ к описанию, а не к чтению. Сначала проверяется доступ **Group**, затем доступ к **Topic**. |
| OFFSET\_FETCH (9) | Describe | Topic | -- |
| FIND\_COORDINATOR (10) | Describe | Group | Запрос **FIND\_COORDINATOR** может иметь тип **Group**, и в этом случае он ищет координаторов групп потребителей. Эта привилегия будет представлять режим **Group**. |
| FIND\_COORDINATOR (10) | Describe | TransactionalId | Применимо только к поставщикам транзакций и проверяется, когда поставщик пытается найти координатора транзакций. |
| JOIN\_GROUP (11) | Read | Group | -- |
| HEARTBEAT (12) | Read | Group | -- |
| LEAVE\_GROUP (13) | Read | Group | -- |
| SYNC\_GROUP (14) | Read | Group | -- |
| DESCRIBE\_GROUPS (15) | Describe | Group | -- |
| LIST\_GROUPS (16) | Describe | Cluster | Когда брокер проверяет авторизацию запроса **list\_groups**, он сначала проверяет авторизацию на уровне **Cluster**. Если ничего не найдено, происходит проверка групп по отдельности. Эта операция не возвращает **CLUSTER\_AUTHORIZATION\_FAILED**. |
| LIST\_GROUPS (16) | Describe | Group | Если ни одна из групп не авторизована, то вместо ошибки будет отправлен просто пустой ответ. Эта операция не возвращает **CLUSTER\_AUTHORIZATION\_FAILED**. Применимо начиная с версии 2.1. |
| SASL\_HANDSHAKE (17) | -- | -- | Подтверждение SASL является частью процесса аутентификации, поэтому здесь невозможно применить какой-либо тип авторизации. |
| API\_VERSIONS (18) | -- | -- | Запрос **API\_VERSIONS** является частью подтверждения протокола RT.StreamingKafka и происходит при подключении и перед любой аутентификацией. Поэтому контролировать это с помощью авторизации невозможно. |
| CREATE\_TOPICS (19) | Create | Cluster | Если нет авторизации на уровне кластера, он не вернёт **CLUSTER\_AUTHORIZATION\_FAILED**, а вернётся к использованию уровня топика, который находится чуть ниже. Это вызовет ошибку, если возникнет проблема. |
| CREATE\_TOPICS (19) | Create | Topic | Применимо начиная с версии 2.0. |
| DELETE\_TOPICS (20) | Delete | Topic | -- |
| DELETE\_RECORDS (21) | Delete | Topic | -- |
| INIT\_PRODUCER\_ID (22) | Write | TransactionalId | -- |
| INIT\_PRODUCER\_ID (22) | IdempotentWrite | Cluster | -- |
| OFFSET\_FOR\_LEADER\_EPOCH (23) | ClusterAction | Cluster | Если для этой операции нет привилегий уровня кластера, будет проверен первый уровень топика. |
| OFFSET\_FOR\_LEADER\_EPOCH (23) | Describe | Topic | Применимо начиная с версии 2.1. |
| ADD\_PARTITIONS\_TO\_TXN (24) | Write | TransactionalId | Этот API применим только к транзакционным запросам. Сначала он проверяет действие **Write**для ресурса **TransactionalId**, затем проверяет **Topic**в субъекте (ниже). |
| ADD\_PARTITIONS\_TO\_TXN (24) | Write | Topic | -- |
| ADD\_OFFSETS\_TO\_TXN (25) | Write | TransactionalId | Подобно **ADD\_PARTITIONS\_TO\_TXN**, это применимо только к транзакционному запросу. Сначала он проверяет действие **Write**для ресурса **TransactionalId**, затем проверяет, может ли он **Read**в данной **Group**(ниже). |
| ADD\_OFFSETS\_TO\_TXN (25) | Read | Group | -- |
| END\_TXN (26) | Write | TransactionalId | -- |
| WRITE\_TXN\_MARKERS (27) | ClusterAction | Cluster | -- |
| TXN\_OFFSET\_COMMIT (28) | Write | TransactionalId | -- |
| TXN\_OFFSET\_COMMIT (28) | Read | Group | -- |
| TXN\_OFFSET\_COMMIT (28) | Group | Topic | -- |
| DESCRIBE\_ACLS (29) | Describe | Cluster | -- |
| CREATE\_ACLS (30) | Alter | Cluster | -- |
| DELETE\_ACLS (31) | Alter | Cluster | -- |
| DESCRIBE\_CONFIGS (32) | DescribeConfigs | Cluster | Если запрашиваются конфигурации брокера, брокер проверит привилегии уровня кластера. |
| DESCRIBE\_CONFIGS (32) | DescribeConfigs | Topic | Если запрашиваются конфигурации топика, брокер проверит привилегии уровня топика. |
| ALTER\_CONFIGS (33) | AlterConfigs | Cluster | Если конфигурации брокера изменены, брокер проверит привилегии уровня кластера. |
| ALTER\_CONFIGS (33) | AlterConfigs | Topic | Если конфигурации топика изменены, брокер проверит привилегии уровня топика. |
| ALTER\_REPLICA\_LOG\_DIRS (34) | Alter | Cluster | -- |
| DESCRIBE\_LOG\_DIRS (35) | Describe | Cluster | При неудачной авторизации будет возвращен пустой ответ. |
| SASL\_AUTHENTICATE (36) | -- | -- | **SASL\_AUTHENTICATE** является частью процесса аутентификации, поэтому здесь невозможно применить какую-либо авторизацию. |
| CREATE\_PARTITIONS (37) | Alter | Topic | -- |
| CREATE\_DELEGATION\_TOKEN (38) | -- | -- | Создание токенов делегирования имеет особые правила. Подробную информацию см. в **п. 3.4.9**. |
| CREATE\_DELEGATION\_TOKEN (38) | CreateTokens | User | Позволяет создавать токены делегирования для ресурса **User**. |
| RENEW\_DELEGATION\_TOKEN (39) | -- | -- | Обновление токенов делегирования имеет особые правила. Подробную информацию см. в **п. 3.4.9**. |
| EXPIRE\_DELEGATION\_TOKEN (40) | -- | -- | Для токенов делегирования с истекающим сроком действия действуют особые правила. Подробную информацию см. в **п. 3.4.9**. |
| DESCRIBE\_DELEGATION\_TOKEN (41) | Describe | DelegationToken | Для описания токенов делегирования действуют особые правила. Подробную информацию см. в **п. 3.4.9**. |
| DESCRIBE\_DELEGATION\_TOKEN (41) | DescribeTokens | User | Позволяет описать токены делегирования ресурса **User**. |
| DELETE\_GROUPS (42) | Delete | Group | -- |
| SELECT\_PREFERRED\_LEADERS (43) | ClusterAction | Cluster | -- |
| INCREMENTAL\_ALTER\_CONFIGS (44) | AlterConfigs | Cluster | Если конфигурации брокера изменены, брокер проверит привилегии уровня кластера. |
| INCREMENTAL\_ALTER\_CONFIGS (44) | AlterConfigs | Topic | Если конфигурации топика изменены, брокер проверит привилегии уровня топика. |
| ALTER\_PARTITION\_REASSIGNMENTS (45) | Alter | Cluster | -- |
| LIST\_PARTITION\_REASSIGNMENTS (46) | Describe | Cluster | -- |
| OFFSET\_DELETE (47) | Delete | Group | -- |
| OFFSET\_DELETE (47) | Read | Topic | -- |

## 3.6 Включение функций безопасности в работающем кластере

Вы можете защитить работающий кластер с помощью одного или нескольких поддерживаемых протоколов, обсуждавшихся ранее. Это делается поэтапно:

1. Инкрементально перебалансируйте ноды кластера, чтобы открыть дополнительные защищённый(е) порт(ы).
2. Перезапустите клиенты, используя защищённый порт, а не порт **PLAINTEXT**(подразумевается, что вы защищаете соединение клиент-брокер).
3. Снова инкрементально перебалансируйте кластер, чтобы включить безопасность между брокерами (если это необходимо).
4. Выполните последнюю инкрементальную перебалансировку для закрытия порта **PLAINTEXT**.

Конкретные шаги по настройке SSL и SASL описаны в **п. 3.3** и **п. 3.4**. Выполните следующие действия, чтобы включить безопасность для нужных протоколов.

Имплементация безопасности позволяет настраивать различные протоколы для связи брокер-клиент и брокер-брокер. Их необходимо включать в отдельных перебалансировках. Порт **PLAINTEXT**должен оставаться открытым, чтобы брокеры и/или клиенты могли продолжать общаться.

При выполнении инкрементной перебалансировки брокеры полностью останавливаются с помощью **SIGTERM**. Также рекомендуется дождаться, пока перезапущенные реплики вернутся в список ISR, прежде чем переходить к следующей ноде.

В качестве примера предположим, что мы хотим зашифровать связь брокер-клиент и брокер-брокер с помощью SSL. При первой инкрементной перебалансировке порт SSL открывается на каждой ноде:

listeners=PLAINTEXT://broker1:9091,SSL://broker1:9092

Copy

Затем мы перезапускаем клиентов, изменяя их конфигурацию, чтобы она указывала на вновь открытый защищённый порт:

bootstrap.servers = [broker1:9092,...]

security.protocol = SSL

...etc

Copy

Во второй инкрементной перебалансировке сервера мы указываем RT.StreamingKafka использовать SSL в качестве протокола брокер-брокер (который будет использовать тот же порт SSL):

listeners=PLAINTEXT://broker1:9091,SSL://broker1:9092

security.inter.broker.protocol=SSL

Copy

В финальной перебалансировке мы защищаем кластер, закрывая порт **PLAINTEXT**:

listeners=SSL://broker1:9092

security.inter.broker.protocol=SSL

Copy

В качестве альтернативы мы можем открыть несколько портов, чтобы можно было использовать разные протоколы для связи брокер-брокер и брокер-клиент. Допустим, мы хотели бы использовать SSL-шифрование повсюду (т.е. для связи брокер-брокер и брокер-клиент), но мы также хотели бы добавить аутентификацию SASL к соединению брокер-клиент. Мы добились бы этого, открыв два дополнительных порта во время первой перебалансировки:

listeners=PLAINTEXT://broker1:9091,SSL://broker1:9092,SASL\_SSL://broker1:9093

Copy

Затем мы перезапустили бы клиенты, изменив их конфигурацию, чтобы она указывала на недавно открытый порт, защищенный SASL и SSL:

bootstrap.servers = [broker1:9093,...]

security.protocol = SASL\_SSL

...etc

Copy

Вторая перебалансировка сервера переключит кластер на использование зашифрованной связи брокер-брокер через порт SSL, который мы ранее открыли на порту 9092:

listeners=PLAINTEXT://broker1:9091,SSL://broker1:9092,SASL\_SSL://broker1:9093

security.inter.broker.protocol=SSL

Copy

Последняя перебалансировка защищает кластер, закрывая порт **PLAINTEXT**.

listeners=SSL://broker1:9092,SASL\_SSL://broker1:9093

security.inter.broker.protocol=SSL

Copy

Powered by [Wiki.js](https://wiki.js.org/)