V8.KafkaClient

КЛИЕНТ СЕРВЕРА APACHE KAFKA ДЛЯ 1С:ПРЕДПРИЯТИЕ 8

Программный интерфейс

версия 1.5.0.0

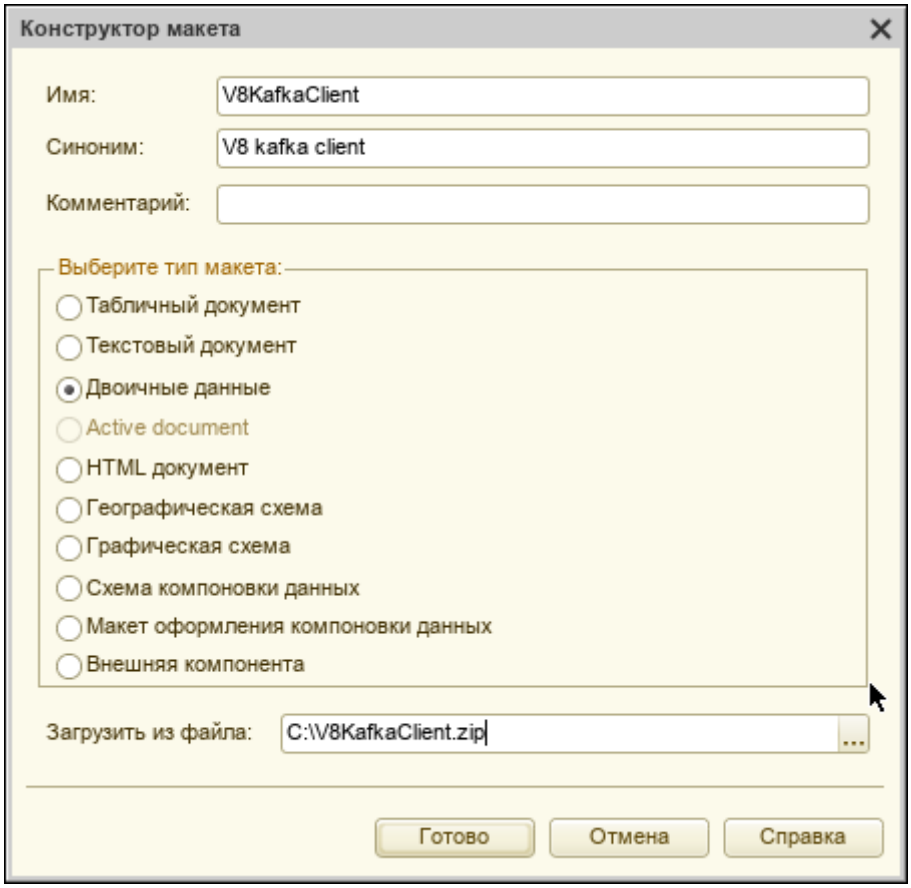
ООО "Серебряная Пуля"

Оглавление

Установка внешней компоненты

Для установки внешней компоненты в вашу конфигурацию создайте общий макет с типом "Двоичные данные".

В поле "Загрузить из файла" выберите архив с компонентой (V8KafkaClient.zip)



В дальнейшем этот макет будет использоваться для подключения внешней компоненты.

Пример:

```
КомпонентаПодключена = ПодключитьВнешнююКомпоненту("ОбщийМакет.V8KafkaClient",
"V8KafkaClient", ТипВнешнейКомпоненты.Native);
```

Код выше приведен для подключения внешней компоненты на сервере.
Если необходимо подключение на клиенте, используются процедуры `НачатьУстановкуВнешнейКомпоненты()` и `НачатьПодключениеВнешнейКомпоененты()`

Пример:

```
&НаКлиенте
Процедура ПодключитьКомпонентуНаКлиенте()
    ОбработчикУстановки = Новый
ОписаниеОповещения("ОбработчикУстановкиКомпоненты", ЭтаФорма);
    НачатьУстановкуВнешнейКомпоненты(ОбработчикУстановки,
"ОбщийМакет.V8KafkaClient");
КонецПроцедуры

&НаКлиенте
Процедура ОбработчикУстановкиКомпоненты(Знач ДополнительныеПараметры) Экспорт
```

```
    ОбработчикПодключения = Новый
ОписаниеОповещения("ОбработчикПодключенияКомпоненты", ЭтаФорма);
    НачатьПодключениеВнешнейКомпоненты(ОбработчикПодключения,
"ОбщийМакет.V8KafkaClient", "V8KafkaClient", ТипВнешнейКомпоненты.Native);
КонецПроцедуры

&НаКлиенте
Процедура ОбработчикПодключенияКомпоненты(Знач Результат, Знач
ДополнительныеПараметры) Экспорт
    КомпонентаПодключена = Результат;
КонецПроцедуры
```

Введение

Данный документ содержит описание внешней компоненты для 1С:Предприятие 8 V8.KafkaClient,
предназначенной для взаимодействия с промышленным сервером «Apache Kafka» из прикладного кода на языке
1С:Предприятие 8.

Документ содержит описание классов и их программных интерфейсов, входящих в состав внешней компоненты.

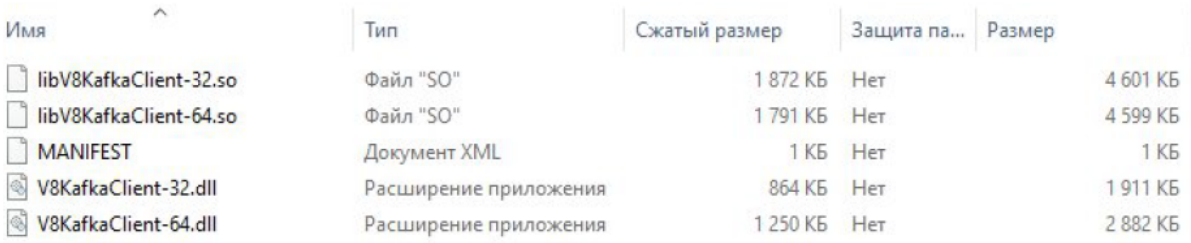
Обратная связь по работе компоненты:

- Официальный телеграмм чат Серебряной пули - SilverBulleters Community @Telegram
- Индивидуальный телеграмм чат клиента, создаваемый при регистрации
- Электронная почта поддержки b2b@silverbulleters.org

Совместимость версий

Внешняя компонента совместима со всеми версиями 1С:Предприятие 8, поддерживающими технологию
NativeAPI. Это все версии платформы начиная с 8.2 и выше.

В комплект поставки входят варианты компоненты для Windows и Linux в 32-битном и 64-битном исполнении.



Имя	Тип	Сжатый размер	Защита па...	Размер
libV8KafkaClient-32.so	Файл "SO"	1 872 КБ	Нет	4 601 КБ
libV8KafkaClient-64.so	Файл "SO"	1 791 КБ	Нет	4 599 КБ
MANIFEST	Документ XML	1 КБ	Нет	1 КБ
V8KafkaClient-32.dll	Расширение приложения	864 КБ	Нет	1 911 КБ
V8KafkaClient-64.dll	Расширение приложения	1 250 КБ	Нет	2 882 КБ

Общие положения

Программный интерфейс классов компоненты построен таким образом, чтобы разработчик мог максимально эффективно использовать уже имеющуюся в Интернет информацию по взаимодействию с Apache Kafka. Поэтому, все конфигурационные параметры, по возможности, выполнены точно так же, как они выполнены в клиентских библиотеках Kafka для других языков программирования.

Например, каждый класс работы с сервером имеет методы взаимодействия с настройками «УстановитьПараметр/ПолучитьПараметр», а сами параметры определяют поведение того или иного аспекта системы. Также, к общему принципу работы с классами можно отнести обработку ошибок. В платформе 1С исключение, брошенное компонентой и перехваченное в языке 1С, не будет содержать текста ошибки (не предусмотрено NativeAPI). Для того, чтобы получить реальный текст ошибки используется вспомогательное свойство «ТекстПоследнейОшибки». Значение последней ошибки очищается при очередном успешном вызове любого метода.

Общие методы всех классов

Активировать/ Activate

Проверяет лицензию компоненты. После подключения компоненты. При создании каждого нового объекта класса компоненты, первым необходимо вызывать этот метод (один раз) и передать содержимое файла лицензии одной строкой. После этого разблокируются остальные методы объекта класса. Файл лицензии поставляется вместе с компонентой.

Параметры:

- license – содержимое файла лицензии. Строка.

Возвращаемое значение:

- Булево. Результат проверки лицензии.

УстановитьПараметр/ SetConfigProperty

Устанавливает значение конфигурации для класса компоненты. Необходимо настраивать конфигурацию до начала отправки/получения сообщений (методы Подписаться, Отправить, Назначить)

Параметры:

- key – ключ параметра. Строка.
- value – значение параметра. Число/Строка/Булево, в зависимости от параметра.

ПолучитьПараметр/GetConfigProperty

Получает значение установленного ранее параметра.

Параметры:

- key – ключ параметра. Строка.
- value – Выходной параметр. Полученное значение параметра. Число/Строка/Булево, в зависимости от параметра.

Возвращаемое значение:

- Булево. Результат получения параметра. Истина – если параметр был задан, Ложь, если параметр отсутствует.

ТекущиеПараметры/DumpConfig

Получает значения всех установленных параметров в виде набора строк Ключ=Значение. Каждый параметр начинается с новой строки.

Примечание

Полный перечень имен параметров и их трактовку можно прочесть по адресу https://github.com/edenhill/librdkafka/blob/master/CONFIGURATION.md

Таймаут/Timeout

Интервал ожидания асинхронных операций. Число.

Большинство операций API Kafka являются асинхронными, а язык 1С, в свою очередь, не имеет средств асинхронного программирования на сервере. В связи с этим, все операции, требующие асинхронного отклика имеют Таймаут, в течение которого этот отклик будет ожидаться.

ТекстПоследнейОшибки/LastErrorDesc

Свойство Позволяет обойти невозможность получения текстов исключений в NativeAPI.

Чтение свойства «ТекстПоследнейОшибки» позволяет получить текст последней известной ошибки.

ПоследнееСообщениеВЛоге/LastMessageLog

Метод Позволяет получить текст последней записи в логе. Это может быть как текст последней ошибки, так и успешное завершение операции

ТекущийЖурнал/DumpLog

Метод Позволяет получить список всех сообщений записанных в лог. Любая операция результатом выполнения которой может быть ложь или истина будет записана в лог.

УстановитьРазмерЛога/SetLogSize

Метод Позволяет установить максимальное количество записей в логе. После переполнения заданного размера будут удалятся самые ранние записи. Для каждого объекта компоненты ведется отдельный лог.

ВерсияБиблиотекиRDKafka/LibRDKafkaVersion

Версия библиотеки от компании Confluent включенной в поставку. Позволяет программисту реализовывать универсальные алгоритмы базирующиеся на информации опубликованной на сайте библиотеки https://github.com/edenhill/librdkafka/releases, а также реализовывать алгоритмы для разных версий сервера Apache Kafka.

Например: `1.7.0`.

ВерсияБиблиотекиV8Kafka/LibV8KafkaVersion

Свойство содержит текущую версию компоненты.
Например:

- `1.4.1.0-commercial` - полнофункциональная компонента

ДвоичныйКанал/BinaryChannel

Свойство позволяет использовать режим работы с двоичными данными. По умолчанию режим отключен. Подробнее в соответствующей главе.

Класс KafkaProducer. Отправитель сообщений

Класс предназначен для отправки сообщений в разделы журнала Kafka.

Свойства класса KafkaProducer

Свойство	Доступ	Тип значения	Описание
МеткаВремениПоследнегоСообщения/LastMessageTimestamp	Чтение	Число	Отметка времени последнего доставленного на сервер сообщения
РазделПоследнегоСообщения/LastMessagePartition	Чтение	Число	Раздел (partition) в который было помещено сообщение
СмещениеПоследнегоСообщения/LastMessageOffset	Чтение	Число	Смещение (offset) по которому было записано сообщение в раздел.
СтатусПоследнегоСообщения/LastMessageStatus	Чтение	Строка	Статус доставки последнего сообщения
ДвоичныйКанал/BinaryChannel	Чтение/ Запись	Булево	Включает/отключает режим работы с двоичными данными. По умолчанию режим выключен.
ВремяОжиданияОтправкиСообщений/MessageWaitSendTime	Чтение/ Запись	Число	Таймаут в мс ожидания отправки всех накопленных сообщений в очереди на брокер при вызове метода ОжидатьОтправкуСообщений
ОжидатьОтправкуКаждогоСообщения/WaitForMessageSent	Чтение/ Запись	Булево	Если установлен в Истина (по умолчанию), то после каждого вызова метода Опубликовать будет вызван метод ОжидатьОтправкуСообщений

Опубликовать/Produce

Метод предназначен для отправки сообщения на сервер Kafka. После вызова метода нельзя изменить конфигурацию вызовом УстановитьПараметр

Параметры:

- Payload – тело сообщения. Строка
- Topic – Тема, в которую отправляется сообщение. Строка
- Partition – Раздел, в который требуется поместить сообщение. Число, Необязательный.
- Key – Ключ сообщения. Строка, Необязательный
- Timestamp – Метка времени. Число/Строка, Необязательный.

Возвращаемое значение:

- Булево. Ложь, если не удалось опубликовать. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Пример кода отправки сообщения

```
Отправитель = Новый("AddIn.KafkaClient.KafkaProducer");
Отправитель.УстановитьПараметр("metadata.broker.list",
"kafka.mydomain.com:9094");

Попытка
    Отправитель.Опубликовать("Привет, Kafka!", "topic");
Исключение
    ВызватьИсключение Отправитель.ТекстПоследнейОшибки;
КонецПопытки;
```

ДобавитьЗаголовок/AddHeader

Метод позволяет добавлять заголовки к отправляемым сообщениям на сервер Kafka. Заголовок представляет собой пару – ключ и значение. Заголовки добавляются к следующему отправляемому сообщению (метод Опубликовать/Produce). После успешной отправки сообщения заголовки удаляются. Таким образом, заголовки необходимо добавлять каждый раз перед отправкой сообщения.

Параметры:

- Key – ключ заголовка. Строка.
- Value – значение заголовка. Число/Строка/Булево

Возвращаемое значение:

- Булево. Ложь, если не удалось добавить заголовок. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

При добавлении заголовка, его значение будет преобразовано в строку. Для типа Булево это будет “true”/”false”. Ключи заголовков могут быть не уникальными.

УдалитьЗаголовок/RemoveHeader

Метод позволяет удалять заголовки. Будут удалены все заголовки с таким ключом.

Параметры:

- Key – ключ заголовка. Строка.

Возвращаемое значение:

- Булево. Ложь, если не удалось удалить заголовок. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Есть смысл удалять заголовки только до отправки сообщения, так как после успешной отправки сообщения все заголовки будут удалены автоматически.

ОчиститьЗаголовки/ClearHeaders

Метод работает так же как и УдалитьЗаголовок/RemoveHeader, но удаляет все имеющиеся заголовки.

Пример кода добавления/удаления заголовков

```
Отправитель = Новый("AddIn.KafkaClient.KafkaProducer");
Отправитель.УстановитьПараметр("metadata.broker.list",
"kafka.mydomain.com:9094");
Попытка
  Отправитель.ДобавитьЗаголовок("Имя", "Kafka");
  Отправитель.ОчиститьЗаголовки();
  Отправитель.ДобавитьЗаголовок("Телефон", 3233);
  Отправитель.ДобавитьЗаголовок("Телефон", 3236);
  Отправитель.ДобавитьЗаголовок("Работает", Истина);
  Отправитель.УдалитьЗаголовок("Телефон");
  Отправитель.Опубликовать("Привет, Kafka!", "demo-topic");
Исключение
  ВызватьИсключение Отправитель.ТекстПоследнейОшибки;
КонецПопытки;
```

ОжидатьОтправкуСообщений/WaitSent

Метод запускает механизм отправки всех не отправленных сообщений из внутренней очереди компоненты. Таймаут ожидания задается свойством ВремяОжиданияОтправкиСообщений в миллисекундах.

Класс KafkaConsumer. Получатель сообщений

Класс предназначен для чтения разделов журнала Kafka.

Возможно 2 варианта подключения к серверу. Первый – подписка методом Subscribe. В этом случае, за назначение раздела отвечает сам сервер Kafka. Клиенту неважно, из какого раздела будет производиться чтение, клиент указывает только Тему (Topic). Второй метод – явное ручное назначение раздела с помощью метода Assign. В этом случае за назначение раздела отвечает автор клиентского приложения.

Вариант подключения может быть установлен только один раз за время жизни экземпляра класса.

При любом варианте подключения для экземпляра класса обязательно должны быть заданы параметры group.id и client.id (см. пример кода чтения сообщений)

Свойства класса KafkaConsumer

Свойство	Доступ	Тип значения	Описание
ДвоичныйКанал/BinaryChannel	Чтение/ Завись	Булево	Включает/ отключает режим работы с двоичными данными. По умолчанию режим выключен.
СмещениеПоследнегоСообщения/LastMessageOffset	Чтение	Число	По умолчанию -1 (не было полученных сообщений)
РазделПоследнегоСообщения/LastMessagePartition	Чтение	Число	По умолчанию -1 (не было полученных сообщений)
ИдБрокераПоследнегоСообщения/LastMessageBrokerId	Чтение	Число	По умолчанию -1 (не было полученных сообщений)
ИмяТемыПоследнегоСообщения/LastMessageTopicName	Чтение	Строка	По умолчанию "" (не было полученных сообщений)

Подписаться/ Subscribe

Подписывается на Тему в Kafka и позволяет читать из нее сообщения методом ConsumeMessage. После вызова метода нельзя изменить конфигурацию вызовом УстановитьПараметр. Если на брокере нет зарегистрированного оффсета для указанного group.id и явно не задан параметр auto.offset.reset, то по умолчанию auto.offset.reset=latest. Метод является асинхронным и вернет управление мгновенно, однако на брокере в этот момент

запускается ребалансировка, которая может занять некоторое время. Рекомендуется в некоторых случаях указывать задержку после вызова данного метода.

Параметры:

- Topic – имя темы. Строка.

Возвращаемое значение:

- Булево. Ложь, если подписка не удалась. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Назначить/Assign

Позволяет явно подключиться к конкретному Разделу в Теме Kafka и читать из него сообщения методом ConsumeMessage. После вызова метода нельзя изменить конфигурацию вызовом УстановитьПараметр

Параметры:

- Topic – имя темы. Строка
- Partition – раздел темы. Число
- Offset – смещение внутри Раздела. Число.

Возвращаемое значение:

- Булево. Ложь, если операция не удалась. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

ПолучитьСообщение/ConsumeMessage

Получает очередное сообщение с сервера Kafka.

Чтение сообщений выполняется последовательным вызовом метода «ПолучитьСообщение». Метод ожидает новое сообщение в течение количества миллисекунд, заданных в свойстве «Таймаут».

Параметры:

- Payload – тело сообщения. Строка. Выходной параметр
- Key – Ключ сообщения. Строка. Выходной параметр
- Timestamp – Метка времени сообщения. Число. Выходной параметр.

Возвращаемое значение:

- Булево. Истина, если сообщение успешно получено. Ложь, если произошла ошибка или истек таймаут. См. подробности в тексте ошибки.

ЗафиксироватьСмещение/ CommitOffset

Явный вызов механизма фиксации сообщений. Блокирует поток исполнения до момента успешной фиксации или превышения таймаута.

Возвращаемое значение:

- Булево. Ложь, если операция не удалась. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Пример кода чтения сообщений

```
Получатель = Новый("AddIn.KafkaClient.KafkaConsumer");
Получатель.УстановитьПараметр("metadata.broker.list",
"kafka.mydomain.com:9094");
Получатель.УстановитьПараметр("group.id", "group1");
Получатель.УстановитьПараметр("client.id", "clietn1");
Получатель.Подписаться("my-topic");

Попытка
    Данные = "";
    СообщениеПолучено = Получатель.ПолучитьСообщение(Данные);
Исключение
    ВызватьИсключение Получатель.ТекстПоследнейОшибки;
КонецПопытки;
```

Установить смещение/SetOffset

Позволяет изменить положение смещения раздела темы для группы потребителей.

Параметры:

- Topic – тема. Строка.
- Partition – раздел темы. Число.
- Offset – смещение в разделе. Число.

Возвращаемое значение:

- Булево. Истина, если смещение успешно изменено. Ложь, если произошла ошибка или истек таймаут. См. подробности в тексте ошибки.

Для смены смещения рекомендуется использовать отдельно созданный объект. После смены смещения рекомендуется его удалить прежде чем продолжить работу с разделом в котором было изменено смещение. Это необходимо чтобы брокер корректно зафиксировал смещение.

Смещение имеет смысл изменять только для группы потребителей. Поэтому, до вызова данного метода нужно обязательно установить параметр с группой:

```
Получатель.УстановитьПараметр("group.id", "group1");
```

Для того чтобы смещение изменилось сразу после вызова данного метода, необходимо установить параметр (до вызова метода subscribe или assign):

```
Получатель.УстановитьПараметр("enable.auto.offset.store ", Ложь);
```

Иначе смещение изменится после вызова метода consume.

Пример кода для изменения смещения

```
Получатель = Новый("AddIn.KafkaClient.KafkaConsumer");
Получатель.УстановитьПараметр("metadata.broker.list",
"kafka.mydomain.com:9094");
Получатель.УстановитьПараметр("group.id", "group1");
Получатель.УстановитьПараметр("enable.auto.offset.store ", Ложь);
Попытка
  СмещениеИзменено = Получатель.ИзменитьСмещение("my-topic", 1, 45);
Исключение
  ВызватьИсключение Получатель.ТекстПоследнейОшибки;
КонецПопытки;
```

ПолучитьКлючиЗаголовков/GetHeadersKeys

Метод позволяет получить список всех ключей заголовков у последнего принятого сообщения. В случае отсутствия заголовков возвращается пустая строка. Заголовки возвращаются в формате «Заголовок1| Заголовок2| ЗаголовокN».

ПолучитьЗначенияЗаголовка/GetHeaderValues

Метод позволяет получить список всех значений заголовков по выбранному ключу у последнего принятого сообщения. В случае отсутствия значений возвращается пустая строка. Значения возвращаются в формате «Значение1| Значение 2| ЗначениеN».

Параметры:

- Key – ключ заголовка. Строка.
- Values – значения. Строка. Выходной параметр

Возвращаемое значение:

- Булево. Ложь, если не удалось получить значения заголовков. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Пример кода получения ключей и значений заголовков

```
Получатель = Новый("AddIn.KafkaClient.KafkaConsumer");
Получатель.УстановитьПараметр("metadata.broker.list",
"kafka.mydomain.com:9094");
Получатель.Подписаться("my-topic");
Попытка
  Данные = "";
  СообщениеПолучено = Получатель.ПолучитьСообщение(Данные);
  КлючиЗаголовков = "";
  КлючиЗаголовков = Получатель.ПолучитьКлючиЗаголовков();
  КлючЗаголовка = "";
  КлючЗаголовка = КлючиЗаголовков; // работает только если получен один
заголовок
  ЗначенияЗаголовков = "";
  ЗначенияПолучены = Получатель.ПолучитьЗначенияЗаголовков(КлючЗаголовка,
ЗначенияЗаголовков);
Исключение
  ВызватьИсключение Получатель.ТекстПоследнейОшибки;
КонецПопытки;
```

Класс KafkaMetadataReader. Читатель метаданных сервера

Класс предназначен для чтения текущей серверной информации о состоянии узлов брокера, его тем, разделов тем, реплик разделов и состоянии синхронизации реплик.

Позволяет программисту осуществлять более удобное написание алгоритмов подписчика в зависимости о состояния узлов брокера, их количества и состояния репликации.

Свойства класса KafkaMetadataReader

Свойство	Доступ	Тип значения	Описание
ProducerName/ ИмяОтправителя	Чтение	Строка	Уникальное имя, присвоенное клиентскому соединению со стороны сервера Apache Kafka
BrokersNodeSize/ КоличествоУзловБрокера	Чтение	Число	Общее количество узлов сервера, координирующих интеграцию
OriginBrokerName / РеальноеИмяБрокера	Чтение	Строка	Уникальное имя узла брокера, с которым< осуществляется в данный момент соединение получения метаданных
BrokerId / ИдентификаторБрокера	Чтение	Строка	Уникальный идентификатор брокера, с которым осуществляется в данный момент соединение получения метаданных
TopicsSize/ КоличествоТем	Чтение	Число	Количество тем созданных на текущий момент в кластере Apache Kafka

Свойства класса KafkaMetadataReader (представления метаданных)

Свойство	Доступ	Тип значения	Описание
ПредставлениеБрокеров/ BrokersView	Чтение	Строка	Многострочная строка представления в формате **id=;host;port;** где каждая строка отражает каждый узел брокера, его идентификатор, сетевое имя и порт на котором осуществляется соединение
ПредставлениеТем/ TopicsView	Чтение	Строка	Многострочная строка представления в формате **name=;partitions_count=;leader_status=;** где каждая строка отражает текущее состояние темы – количество разделов и состояние координирующего лидера
ПредставлениеРазделов/ PartitionsView	Чтение	Строка	Многострочная строка представления в формате **topic_name=;partitions_id=;leader_id=;** где каждая строка отражает текущее состояние раздела темы – его идентификатор, имя связанной темы и т.д.
ПредставлениеРепликРазделов/ PartitionsReplicasView	Чтение	Строка	Многострочная строка представления в формате **topic_name=;partitions_id=;replica=;** где каждая строка отражает текущее состояние реплики раздела и ее нахождение на определенном брокере
ПредставлениеСинхронизированныхРепликРазделов/ ISPartitionsReplicasView	Чтение	Строка	Многострочная строка представления в формате **topic_name=;partitions_id=; irs_replica=;** отражающая состояние синхронизации реплик

ПроверитьСоединение/ CheckConnection

Проверяет соединение с указанным списком серверов, по указанному варианту аутентификации. В случае успешности проверки – заполняет все свойства.

Возвращаемое значение:

- Булево. Ложь, если операция не удалась. Текст ошибки см. в свойстве «ТекстПоследнейОшибки».

Пример получения метаданных сервера

```
ОбъектМетаданных = Новый("AddIn.V8KafkaClient.KafkaMetadataReader");
ОбъектМетаданных.УстановитьПараметр("metadata.broker.list", СписокСерверов);
ОбъектМетаданных.УстановитьПараметр("sasl.mechanisms","PLAIN");

СоединениеПроверно = ОбъектМетаданных.ПроверитьСоединение();
Если СоединениеПроверно Тогда
    ТекстовоеПредставление =
        "" + ОбъектМетаданных.ПредставлениеБрокеров + "
        |" + ОбъектМетаданных.ПредставлениеТем + "
        |" + ОбъектМетаданных.ПредставлениеРазделов + "
        |" + ОбъектМетаданных.ПредставлениеБрокеров + "
        |" + ОбъектМетаданных.ПредставлениеРепликРазделов + "
        |" + ОбъектМетаданных.ПредставлениеСинхронизированныхРепликРазделов;
    Сообщить(ТекстовоеПредставление);
Иначе
    ВызватьИсключение ОбъектМетаданных.ТекстПоследнейОшибки;
КонецЕсли;
```

Работа с двоичными данными

Для передачи сообщения с двоичными данными необходимо выставить свойство ДвоичныйКанал/BinaryChannel в Истина у экземпляра объекта ОтправительСообщений, который будет отправлять сообщения. Далее работать с классом как обычно, за исключением того, что в метод Опубликовать/Produce передавать сообщение типа ДвоичныеДанные.

Для приема сообщения с двоичными данными необходимо выставить свойство ДвоичныйКанал/BinaryChannel в Истина у экземпляра объекта ПолучательСообщений, который будет получать сообщения. Далее работать с классом как обычно, за исключением того, что метод ПолучитьСообщение/ConsumeMessage будет записывать сообщения только в переменную типа ДвоичныеДанные.

Свойство ДвоичныйКанал/BinaryChannel можно изменять в любой момент между приемом сообщений. Только необходимо учитывать что и входные/выходные данные должны так же изменить свой тип.

По умолчанию свойство установлено в Ложь.

Ручное управление доставкой сообщений на брокер

Для управления доставкой в компоненте предусмотрены следующие свойства и методы класса KafkaProducer.ОтправительСообщений:

```
- ОжидатьОтправкуКаждогоСообщения/WaitForMessageSent;
- ВремяОжиданияОтправкиСообщений/MessageWaitSendTime;
- ОжидатьОтправкуСообщений/WaitSent.
```

Существует три основных стратегии управления доставкой сообщений на брокер.

- Ожидание доставки после каждого вызова метода Опубликовать. Данная стратегия используется по умолчанию. Время ожидания доставки задается в свойстве класса ВремяОжиданияОтправкиСообщений/MessageWaitSendTime. По умолчанию параметр равен 10000мс. К плюсам можно отнести гарантию доставки каждого сообщения (при условии возврата методом Опубликовать значения Истина), а так же возможность отслеживать доставку сообщений по порядку. Минусом данного подхода является низкая производительность, так как требуется ожидать от брокера ответ каждый раз после отправки сообщений до того как начать отправку следующего сообщения. В среднем производительность при таком подходе составляет 1/[пинг до брокера] даже при небольших размерах сообщений. Частично это можно исправить, если используются сообщения большого размера или если сообщения запаковывать в батч перед отправкой.

  Пример использования

  ```
  Успех = ПодключитьВнешнююКомпоненту("ОбщийМакет.V8KafkaClient",
  "V8KafkaClient", ТипВнешнейКомпоненты.Native);
  Если Не Успех Тогда
      ВызватьИсключение НСтр("ru = 'Не удалось подключить внешнюю компоненту
  V8KafkaClient'");
  КонецЕсли;
  Отправитель = Новый("AddIn.V8KafkaClient.KafkaProducer");
  Отправитель.ВремяОжиданияОтправкиСообщений = 30000;
  Сообщение = "МоеСообщение";
  Для НомерСообщения = 1 По 100 Цикл
      Попытка
          УспешноДоставлено =
  Отправитель.Опубликовать(Сообщение+НомерСообщения, "НазваниеТемыНаБрокере");
  // внутри вызывается метод WaitForMessagesSent
      Исключение
          Сообщить(Отправитель.ТекстПоследнейОшибки,
  СтатусСообщения.Информация);
          ВызватьИсключение ОписаниеОшибки();
      КонецПопытки;
  КонецЦикла;
  ```

- Не ожидать доставку сообщений. Чтобы использовать такой подход необходимо перед началом отправки сообщений выставить свойство ОжидатьОтправкуКаждогоСообщения/WaitForMessageSent = Ложь. Плюсы данной стратегии это максимальная производительность, которую может выдать kafka. Так же нет

необходимости в дополнительных действиях - сообщения автоматически отправляются как только собирается батч, то он автоматически отправляется на брокер. Размером батча можно управлять через метод УстановитьПараметр/SetConfigProperty и выставив параметр "batch.size". В случае если какое либо сообщение не дошло до брокера по причине проблем с сетью, компонента самостоятельно вышлет это сообщение повторно в фоновом режиме (количество попыток задается параметром retries). К недостаткам такого подхода можно отнести невозможность последовательно контролировать доставку сообщений. Так же есть вероятность потерять не отправленные сообщения в случае, если по каким либо причинам работа компоненты будет завершена аварийно. Так как до отправки батча он хранится в оперативной памяти.

Пример использования

```
Успех = ПодключитьВнешнююКомпоненту("ОбщийМакет.V8KafkaClient",
"V8KafkaClient", ТипВнешнейКомпоненты.Native);
Если Не Успех Тогда
    ВызватьИсключение НСтр("ru = 'Не удалось подключить внешнюю компоненту
V8KafkaClient'");
КонецЕсли;
Отправитель = Новый("AddIn.V8KafkaClient.KafkaProducer");
Отправитель.УстановитьПараметр("batch.size", 1024); // Размер батча не будет
превышать 1024 байта
Отправитель.ОжидатьОтправкуКаждогоСообщения = Ложь; // метод
ОжидатьОтправкуСообщений() будет вызываться атоматически
Отправитель.ВремяОжиданияОтправкиСообщений = 30000;
Сообщение = "МоеСообщение";
Для НомерСообщения = 1 По 100 Цикл
    Попытка
        Отправитель.Опубликовать(Сообщение+НомерСообщения,
"НазваниеТемыНаБрокере");
    Исключение
        Сообщить(Отправитель.ТекстПоследнейОшибки,
СтатусСообщения.Информация);
        ВызватьИсключение ОписаниеОшибки();
    КонецПопытки;
КонецЦикла;
```

- ОжидатьОтправкуСообщений/WaitSent. Чтобы использовать этот подход необходимо как и в предыдущем подходе выставить свойство ОжидатьОтправкуКаждогоСообщения/WaitForMessageSent = Ложь и выставить свойство таймаута ВремяОжиданияОтправкиСообщений/MessageWaitSendTime. Далее вызывать метод ОжидатьОтправкуСообщений/WaitSent после одного или нескольких вызовов метода Опубликовать/Prooduce. Возвращаемый результат данного метода сообщает об успешности доставки всего пакета. Данный подход является чем то средним между первыми двумя, так как появляется возможность контролировать доставку для группы сообщений, а производительность выше чем в стратегии по умолчанию. Так же можно регулировать производительность увеличивая или уменьшая размер пакета сообщений.

  Пример использования

```
Успех = ПодключитьВнешнююКомпоненту("ОбщийМакет.V8KafkaClient",
"V8KafkaClient", ТипВнешнейКомпоненты.Native);
Если Не Успех Тогда
    ВызватьИсключение НСтр("ru = 'Не удалось подключить внешнюю компоненту
V8KafkaClient'");
КонецЕсли;
```

```
Отправитель = Новый("AddIn.V8KafkaClient.KafkaProducer");
Отправитель.ОжидатьОтправкуКаждогоСообщения = Ложь;
Отправитель.ВремяОжиданияОтправкиСообщений = 30000;
Сообщение = "МоеСообщение";
Для НомерСообщения = 1 По 100 Цикл
    Попытка
        Отправитель.Опубликовать(Сообщение+НомерСообщения,
"НазваниеТемыНаБрокере");
        Если НомерСообщения % 10 = 0 Тогда // размер пакета 10 сообщений
            ПакетУспешноОтправлен = Отправитель.ОжидатьОтправкуСообщений();
        КонецЕсли;
    Исключение
        Сообщить(Отправитель.ТекстПоследнейОшибки,
СтатусСообщения.Информация);
        ВызватьИсключение ОписаниеОшибки();
    КонецПопытки;
КонецЦикла;
```

Использование компрессии отправляемых сообщений

Для использования компресии на стророне продъюсера при отправке сообщений необходимо добавить параметр конфигурации 'compression.codec' и указать желаемый кодек. возможные значения zstd, gzip, snappy, lz4. Пример:

```
Producer.УстановитьПараметр("compression.codec", "zstd");
```

Отправляемые сообщения будут запакованы перед отправкой. Есть возможность дополнительно регулировать степень сжатия, для каждого типа кодека это свое значение. За это отвечает параметр 'compression.level', подробнее описано тут https://github.com/confluentinc/librdkafka/blob/master/CONFIGURATION.md#:~:text=Type%3A%20enum%20value-,compression.level,-P

Компрессия на отпрвавляемые сообщения отключается, если размер сообщения становится слишком маленьким. Для каждого типа кодека это свои пороговые значения.

При приеме сообщений дополнительных действий не требуется, сообщения распакуются автоматически.

История релизов

Релиз 1.5.0.0 от 14.06.2023

- Обновлена зависимость librdkafka с 1.9.2 до 2.1.1
- Добавлена поддержка авторизации OIDC

Релиз 1.4.1.0 от 15.02.2023

- Исправлена ошибка с подключением при использовании ssl
- Добавлена поддержка компрессии snappy и lz4

Релиз 1.4.0.0 от 26.01.2023

- Добавлена поддержка компрессии zstd. Для использования:
  - перед отправкой указать параметр конфигурации compression.type=zstd;
  - для приема сообщений с компрессией zstd дополнительных действий не требуется
- Добавлено лицензирование. Для активации лицензии необходимо вызвать первым метод "Activate/Активировать" и передать туда содержимое файла лицензии одной строкой.
- Обновлена зависимость librdkafka с 1.7.0 до 1.9.2

Релиз 1.3.0.0 от 25.02.2022

- Добавлена возможность управления доствкой сообщений на брокер
- Добавлены новые свойства класса KafkaConsumer.ПолучательСообщений:
  - `СмещениеПоследнегоСообщения/LastMessageOffset`
  - `РазделПоследнегоСообщения/LastMessagePartition`
  - `ИдБрокераПоследнегоСообщения/LastMessageBrokerId`
  - `ИмяТемыПоследнегоСообщения/LastMessageTopicName`

Релиз 1.2.0.0 от 08.06.2021

- Обновлена зависимость librdkafka с 1.6.0 до 1.7.0
- Добавлена возможность работы с двоичными данными.

Релиз 1.1.0.0 от 26.04.2021

- Обновлена зависимость librdkafka с 1.5.3 до 1.6.0
- Добавлена поддержка заполнения и чтения заголовков сообщений (HEADER) для классов KProducer и KConsumer.
- Добавлена возможность изменять смещение для группы потребителей в классе KConsumer.
- Добавлено логирование действий пользователя
- Повышена стабильность работы компоненты

  более подробно о всех изменениях описано в документации.

Релиз 1.0.2.2 от 15.12.2020

- Зависимость librdkafka обновлена с 1.5.2 до 1.5.3

Релиз 1.0.2.1 от 16.11.2020

- Добавлено два новых общих свойства:
  - `ВерсияБиблиотекиRDKafka` - версия librdkafka в компоненте
  - `ВерсияБиблиотекиV8Kafka` - версия компоненты
- Зависимость librdkafka обновлена с 1.0.1 до 1.5.2

Релиз 1.0.1.4

Добавлен новый объект

```
KafkaMetadataReader
```

– предназначен для чтения метаданных брокера сообщений.

Доступно:

- чтение узлов брокера
- чтение имеющихся тем брокера, включая служебные
- чтение имеющихся разделов и тем брокера

более подробно смотри раздел документации Класс KafkaMetadataReader. Читатель метаданных сервера