webMethods Adapter for Apache Kafka Installation and User's Guide

Version 9.6

December 2016

WEBMETHODS

This document applies to webMethods Adapter for Apache Kafka 9.6 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ADAPTER-WAK-IUG-96-20240321

Table of Contents

About this Guide

This guide describes how to install, configure, and use the webMethods Adapter for Apache Kafka. This guide also describes the built-in services provided by the Adapter for Apache Kafka. It contains information for administrators and application developers who want to interact with Apache Kafka through the use of Apache Kafka client API.

To use this guide effectively, you should be familiar with:

- The basic concepts and tasks for working with Apache Kafka
- The terminology and the basic operations of your operating system
- The setup and operation of webMethods Integration Server
- The basic concepts and tasks of webMethods Deployer

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at https://documentation.softwareag.com.

In addition, you can also access the cloud product documentation via https://www.softwareag.cloud. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at https://knowledge.softwareag.com.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at https://techcommunity.softwareag.com. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at https://github.com/softwareag and https://hub.docker.com/u/softwareag and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at https://empower.softwareag.com. Many services on this portal require that you have an account. If you do not yet have one, you can request it at https://empower.softwareag.com/register. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Overview of the Adapter

About the Adapter

webMethods Adapter for Apache Kafka is an add-on to webMethods Integration Server that enables you to exchange data/messages with a distributed publish-subscribe messaging system called Apache Kafka. The adapter provides a unified, high-throughput, low-latency platform for handling real-time data feeds.

Using Adapter for Apache Kafka, Integration Server clients can create and run services to perform operations such as, retrieve data from, and insert data in the Apache Kafka cluster.

For example, you can use Adapter for Apache Kafka to extract information from an XML-based purchase order, repackage it as an order record, and deliver it to a Apache Kafka topic for processing by a back-end enterprise system.

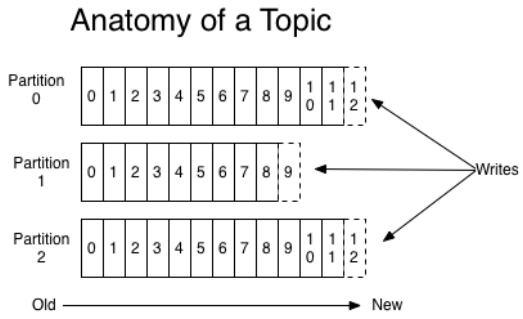
Apache Kafka Concepts

This section discusses the following concepts:

Topics and Logs

A topic is a category or feed name to which messages are published. Topics in Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it.

For each topic, the Kafka cluster maintains a partitioned log as shown below:



Each partition is an ordered, immutable sequence of records that is continually appended to a structured commit log. The records in the partitions are each assigned a sequential id number called the offset that uniquely identifies each record within the partition. The Kafka cluster retains all published records, whether or not they have been consumed using a configurable retention period. For example, if the retention policy is set to two days, then two days after a record is published, it is available for consumption, after which it will be discarded to free up space. Kafka's performance is effectively constant with respect to data size so storing data for a long time is not a problem.

The partitions in the log serve the following purposes:

- Allows the log to scale beyond a size that will fit on a single server. Each individual partition must fit on the servers that host it, but a topic may have many partitions so it can handle an arbitrary amount of data
- Act as the unit of parallelism.

Distribution

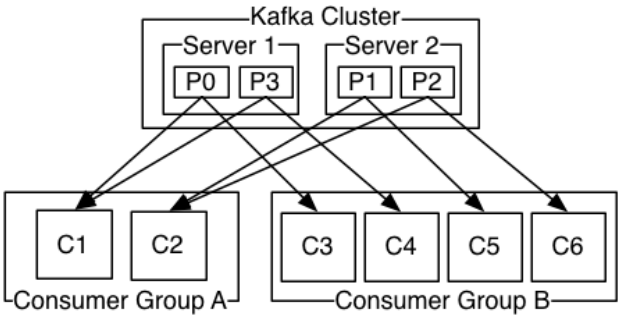
Each partition is an ordered, immutable sequence of records that is continually appended to a structured commit log. The records in the partitions are each assigned a sequential id number called the offset that uniquely identifies each record within the partition. The Kafka cluster retains all published records, whether or not they have been consumed using a configurable retention period. For example, if the retention policy is set to two days, then two days after a record is published, it is available for consumption, after which it will be discarded to free up space. Kafka's performance is effectively constant with respect to data size so storing data for a long time is not a problem. The partitions of the log are distributed over the servers in the Kafka cluster with each server handling data and requests for a share of the partitions. Each partition is replicated across a configurable number of servers for fault tolerance. Each partition has one server which acts as the "leader" and zero or more servers which act as "followers". The leader handles all read and write requests for the partition while the followers passively replicate the leader. If the leader fails, one of the followers will automatically become the new leader. Each server acts as a leader for some of its partitions and a follower for others so load is well balanced within the cluster.

Producers

Producers publish data to the topics of their choice. The producer is responsible for choosing which record to assign to which partition within the topic. This can be done in a round-robin fashion to balance the load or it can be done according to any semantic partition function, such as usage of some key in the record.

Consumers

Consumers label themselves with a consumer group name, and each record published to a topic is delivered to one consumer instance within each subscribing consumer group. Consumer instances can be in separate processes or on separate machines. If all the consumer instances have the same consumer group, then the records will effectively be load balanced over the consumer instances. If all the consumer instances have different consumer groups, then each record will be broadcasted to all the consumer processes.



The below diagram shows how a two server Kafka cluster hosting four partitions (P0-P3) with two consumer groups. Consumer group A has two consumer instances and group B has four.

More commonly, however, we have found that topics have a small number of consumer groups, one for each "logical subscriber". Each group is composed of many consumer instances for scalability and fault tolerance. This is called publish-subscribe semantics where the subscriber is a cluster of consumers instead of a single process.

The way consumption is implemented in Kafka is by dividing up the partitions in the log over the consumer instances. The process of maintaining membership in the group is handled by the Kafka protocol dynamically. If new instances join the group they will take over some partitions from other members of the group; if an instance ends, its partitions will be distributed to the remaining instances.

Guarantees

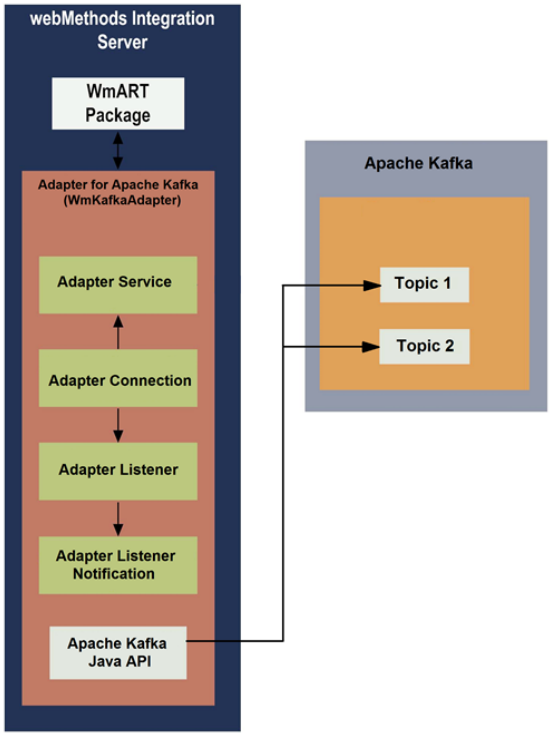
Apache Kafka provides the following guarantees:

- Messages sent by a producer to a particular topic partition will be appended in the order they are sent. For example, if a message M1 and M2 are sent by the same producer, but M1 is sent first, then M1 will have a lower offset than M2 and the message M1 appears earlier in the log.
- A consumer instance sees messages in the order they are stored in the log.
- For a topic with replication factor N, the tolerance is up to N-1 server failures without losing any records committed to the log.

Architecture and Components

Adapter for Apache Kafka provides a set of user interfaces, services that enables you to create integration with Apache Kafka. The adapter is provided as a single package that must be installed on Integration Server. For detailed installation instructions, see "Overview of installing and uninstalling Adapter for Apache Kafka" on page 24. For software requirements, see *webMethods Adapters System Requirements*.

The following diagram describes the different architectural pieces involved in the integration process:



- **webMethods Integration Server.** Adapter for Apache Kafka is installed and runs on Integration Server.
- **(WmART).** The WmART package provides a common framework for webMethods Adapter for Apache Kafka version 9.6 and later to use Integration Server's functionality, making Integration Server the run-time environment for Adapter for Apache Kafka. The WmART package is installed with Integration Server and it provides logging, error handling for the adapter, connections, notifications, and services.
- **Adapter for Apache Kafka.** The Adapter for Apache Kafka is delivered as a single package called WmKafkaAdapter. The adapter installation includes templates from which all adapter connections, notifications, and adapter services can be created.

  Adapter for Apache Kafka provides user interfaces in:

  - Integration Server Administrator that enable you to configure and manage adapter connections, listeners, and notifications.
  - webMethods Deployer and Designer user interfaces that will enable you to configure and manage adapter services. listeners, and notifications.
- **Adapter services.** Adapter services enable Adapter for Apache Kafka to produce and consume messages from Apache Kafka destinations(topics). The adapter provides service templates that enable you to configure service templates. For a detailed description of adapter services, see "Adapter Services" on page 18
- **Adapter connections.** Adapter connections enable the Integration Server to connect to Apache Kafka at runtime. You must configure an adapter connection before you can configure adapter

services. For a detailed description of adapter connections, see "Adapter Connections" on page 16

- **Adapter Listener.** Adapter listener monitors the Apache Kafka topics and passes the messages to a listener notification. You must configure adapter listeners before you configure adapter notifications. Adapter for Apache Kafka provides adapter listener templates that enable you to configure listeners, which monitors a Apache Kafka topic for messages. For a detailed description of adapter listener, see "Listeners" on page 19
- **Adapter Listener Notifications.** Adapter notifications enable the Adapter for Apache Kafka to listen to and consume messages from Apache Kafka destinations (topics). The Adapter for Apache Kafka provides an adapter notification template that enables you to configure adapter notifications.
- **Apache Kafka Java API.** Apache Kafka Java API is a client jar provided by Kafka that connects to the Kafka cluster.
- **Apache Kafka.** Apache Kafka is a distributed publish-subscribe messaging system designed to replace traditional message brokers. Message brokers are a type of middleware that translates messages of one language to another, usually more commonly-accepted language. Apache Kafka improves on traditional message brokers through advances in throughput, built-in partitioning, replication, latency and reliability.

Package Management

Adapter for Apache Kafka is provided as a package called WmKafkaAdapter that you manage like any package on Integration Server.

There are several considerations regarding how you set up and effectively manage your packages on Integration Server:

You can set up and manage the packages on Integration Server using the following considerations:

- Create user-defined packages for your connections, adapter services, and listener notifications. For details, see "Overview of Package Management" on page 28 .
- Understand how package dependencies work so that you can decide on how to manage your adapter services and notifications. For details, see "Package Dependency Requirements and Guidelines" on page 28 .
- You must have a control on which development groups have access to which adapter services and notifications. For details, see "Group Access Control" on page 31.

Adapter Connections

Adapter for Apache Kafka connects to Kafka through Java API at run time. You create one or more connections at design time to use in integrations. The number of connections you create and the type of those connections depend on the topics you are connecting to and your integration needs.

Adapter for Apache Kafka connections contain parameters that Integration Server uses to manage connections on the Apache Kafka. The adapter uses these connections to provide services. You

configure connections using Integration Server Administrator. You must have Integration Server Administrator privileges to access Adapter for Apache Kafka's administrative screens.

For instructions on configuring Adapter for Apache Kafka connections, see "Overview of Adapter for Apache Kafka Connections" on page 36 . For information about seeing user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

For a list of tasks that you must do before you can create your connections, see "Before Configuring or Managing Adapter Connection" on page 36

Connection Pools

Integration Server includes a connection management service that dynamically manages connections and connection pools based on configuration settings that you specify for the connection. All adapter services use connection pooling.

A connection pool is a collection of connections with the same set of attributes. Integration Server maintains connection pools in memory. Connection pools improve performance by enabling adapter services to re-use the open connections instead of opening new connections.

Run-Time Behavior of Connection Pools

When you enable a connection, Integration Server initializes the connection pool, creating the number of connection instances you specified in the connection's **Minimum Pool Size** field when you configured the connection. Whenever an adapter service needs a connection, Integration Server provides a connection from the pool. If no connections are available in the pool, and the maximum pool size has not been reached, the server creates one or more new connections (according to the number specified in the **Pool Increment Size** field) and adds them to the connection pool. If the pool is full (as specified in **Minimum Pool Size** field), the requesting service will wait for Integration Server to obtain a connection, up to the length of time specified in the **Block Timeout** field, until a connection becomes available. Periodically, Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period that you specified in the **Expire Timeout** field.

If initialization of the connection pool fails because of a network connection failure or some other type of exception, you can enable the system to retry the initialization any number of times, at specified intervals. For information about configuring connections, see "Overview of Adapter for Apache Kafka Connections" on page 36.

Built-In Services for Connections

Integration Server provides built-in services that enable you to programmatically control connections. You can use them to enable and disable a connection, and to return usage statistics and the current state (Enabled or Disabled) and error status for a connection. These services are located in the WmART package, in the pub.art.connection folder.

The setAdapterServiceNodeConnection built-in service enables you to change the connection associated with an adapter service respectively.

For details, see the *webMethods Integration Server Built-In Services Reference* for your release.

Adapter Services

To use Adapter for Apache Kafka, you create adapter services. Adapter services allow you to connect to the adapter's resource and initiate an operation on the resource from Integration Server

You call adapter services from flow or Java services to interact with Kafka. Adapter for Apache Kafka services perform messaging services by calling Apache Kafka Java API. Integration Server then uses the adapter connections which is defined earlier to execute the adapter services. For details, see .

Adapter services are based on templates provided with Adapter for Apache Kafka. Each template represents a specific technique for doing work on a resource, such as using the Produce Service template to deliver a message to a specified Kafka topic.

An adapter service template contains all the code necessary for interacting with the resource but without the data specifications. You provide these specifications when you create a new adapter service.

Creating a new service from an adapter service template is straightforward. Using webMethods Deployer, you assign the service a default adapter connection.

After you select the connection for the adapter service, you select the adapter service template and supply the data specifications using Designer. Some familiarity with using Designer is required. For more information, see the *webMethods Service Development Help* for your release.

Adapter for Apache Kafka provides the following adapter service templates:

Adapter Service Template	Description
Produce service	Publishes messages to a Kafka topic. For instructions about configuring the service, see "Configuring Produce Service" on page 68.
Consume service	Consumes messages from a Kafka topic. For instructions about configuring the service, see "Configuring Consume Service" on page 70.

Using Adapter Services

The tasks required to use the adapter services are as follows:

1. Create an adapter connection using Integration Server Administrator. For details, see "Overview of Adapter for Apache Kafka Connections" on page 36.

2. Select the appropriate adapter service template and configure the adapter service using Designer.

   Depending on the type of adapter service, you specify:

   - The adapter connection
   - The input fields and types as needed
   - The output fields and types as needed

   For more information about configuring adapter services, see "Overview of Adapter Services" on page 68.

3. If you plan to use an Integration Server flow or Java service to invoke the adapter service, design the flow or Java service to use this adapter service. You can use Designer to perform this task.

4. Manage the adapter service using Designer and Integration Server.

Adapter Listeners and Listener Notifications

Adapter for Apache Kafka provides listeners and listener notifications to process the messages.

Listeners

Adapter for Apache Kafka listener continually monitors a Kafka topic for messages. When a message appears on a topic, the listener fetches the message based on the filter criteria that you select when you configure the listener. The listener passes the message to the listener notification. Unlike the Produce Service and Consume Service, you never invoke a listener directly from a service or client. Instead, a listener is a real-time process that you configure, enable, and disable using the Integration Server Administrator.

- All the listeners will stop functioning when the package containing the listener node is disabled or when Integration Server shuts down.
- All the enabled listeners will start functioning when the package containing the listener node is enabled or when Integration Server restarts.

For information about how to configure listeners, see "Configuring a New Listener" on page 93.

Listener Notifications

A listener notification works in conjunction with a listener to process messages in Adapter for Apache Kafka. When a listener receives a message from a Kafka topic, then it passes the message to an enabled listener notification that is associated with the listener.

Important:

The message is lost if you do not configure any notifications with a listener, or if you do not enable any listener notifications which are already configured.

You can configure one or more listener notifications for each listener. If you configure multiple listener notifications with the same listener, the listener the listener passes the data to the listener notifications created first. For more information, see "Overview of Listener and Listener Notification" on page 92.

Adapter for Apache Kafka provides the following listener notification types:

- Synchronous Apache Kafka Listener Notifications
- Asynchronous Apache Kafka Listener Notifications
- Synchronous Confluent Kafka Listener Notifications
- Asynchronous Confluent Kafka Listener Notifications

Asynchronous Apache Kafka Listener Notifications

When you create an asynchronous listener notification (Apache Kafka or Confluent Kafka), Adapter for Apache Kafka creates a publishable document type. The publishable document type defines the structure of the document. At run time, after the listener retrieves a message from the Kafka topic, the listener invokes the asynchronous listener notification. The asynchronous listener notification publishes the document that has the structure defined by the publishable document type created with the asynchronous listener notification.

An asynchronous listener notification can publish a document in either of the following ways:

- Publish to a Universal Messaging when Integration Server is connected to a Universal Messaging.
- Publish locally to Integration Server when Integration Server is not connected to a Universal Messaging.
- Publish to a JMS Provider configured using Integration Server Administrator.

To process the message from the queue, you must create an Integration Server trigger that subscribes to the document type that Adapter for Apache Kafka created with the asynchronous listener notification. For more information about using triggers with services, see the *Publish-Subscribe Developer's Guide* for your release.

To select a publishing destination for asynchronous listener notification messages, see "Configuring an Apache Kafka Asynchronous Listener Notification" on page 104 and "Configuring an Confluent Kafka Asynchronous Listener Notification" on page 108.

Synchronous and asynchronous listener notifications (Apache Kafka or Confluent Kafka) are applicable for Apache Kafka and Confluent Kafka Consumers only.

Synchronous Apache Kafka Listener Notifications

When you create a synchronous listener notification (Apache Kafka or Confluent Kafka), Adapter for Apache Kafka creates the following document types:

- **Request Document**. Defines the structure of the document.
- **Reply Document**.

At run time the listener notification invokes the service configured in the listener notification and passes the specified Request document. The configured service returns the Reply document to Adapter for Apache Kafka. The listener notification waits until the service has finished processing the message before it initiates the processing the next message. A synchronous listener notification does not publish a document.

For more information about specifying the execution modes for synchronous listener notifications, see "Configuring a Synchronous Apache Kafka Listener Notification" on page 103 and "Configuring a Synchronous Confluent Kafka Listener Notification" on page 106.

Listener Notification Templates

Adapter for Apache Kafka provides the following adapter notification templates:

Listener Notification	Description
Apache Kafka Listener Notification (synchronous)	Invokes a service to process the message before receiving the next message. For more information, see "Configuring a Synchronous Apache Kafka Listener Notification" on page 103.
Apache Kafka Listener Notification (asynchronous)	Publishes a document type that Integration Server processes asynchronously. That is, Integration Server does not wait until it has finished processing a document before processing another document. For more information, see "Configuring an Apache Kafka Asynchronous Listener Notification" on page 104.
Confluent Kafka Listener Notification (synchronous)	Invokes a service to process the message before receiving the next message. Allows the user to configure Key Schema and Value Schema files that reflects the changes in the Request Document. For more information, see "Configuring a Synchronous Confluent Kafka Listener Notification" on page 106.
Confluent Kafka Listener Notification (asynchronous)	Publishes a document type that Integration Server processes asynchronously. That is, Integration Server does not wait until it has finished processing a document before processing another document. Allows the user to configure Key Schema and Value Schema files that reflects the changes in the Request Document. For more information, see "Configuring an Confluent Kafka Asynchronous Listener Notification" on page 108.

Using Version Control Systems to Manage Adapter Elements

The adapter supports the Version Control System (VCS) Integration feature provided by Designer. When you enable the feature in Integration Server, you can check adapter packages or elements into and out of your version control system from Designer. For more information about the VCS Integration feature, see the *Configuring the VCS Integration Feature*.

Beginning with Integration Server 8.2 SP3, the adapter supports the local service development feature in Designer. This feature extends the functionality of the VCS Integration feature to check package elements and their supporting files into and out of a VCS directly from Designer. For more information about local service development and how it compares to the VCS Integration feature, see the *webMethods Service Development Help*.

Viewing the Adapter's Update Level

You can view the list of updates that have been applied to the adapter **Updates** field on the adapter's About page in Integration Server Administrator.

Controlling Pagination

When using the adapter on Integration Server 9.6 and later, you can control the number of items that are displayed on the adapter Connections screen and Notifications screen. By default, 10 items are displayed per page. Click **Next** and **Previous** to move through the pages, or click a page number to go directly to a page.

To change the number of items displayed per page, set the watt.art.page.size property and specify a different number of items.

To set the number of items per page

1. From Integration Server Administrator, click **Settings > Extended**.
2. Click **Edit Extended Settings**. In the Extended Settings editor, add or update the watt.art.page.size property to specify the preferred number of items to display per page. For example, to display 50 items per page, specify:

   ```
   watt.art.page.size=50
   ```

3. Click **Save Changes**. The property appears in the Extended Settings list.

   For more information about working with extended configuration settings, see the *webMethods Integration Server Administrator's Guide* for your release.

2 Installing, and Uninstalling Adapter for Apache Kafka

Overview of installing and uninstalling Adapter for Apache Kafka

This chapter explains how to install, and uninstall webMethods Adapter 9.6 for Apache Kafka. The instructions use the Software AG Installer and Software AG Uninstaller the wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the *Installing webMethods Products On Premises* for your release.

Requirements

For a list of operating systems, and webMethods products supported by Adapter for Apache Kafka, see *webMethods Adapters System Requirements*.

Adapter for Apache Kafka has no hardware requirements beyond those of its host Integration Server.

The Integration Server Home Directory

Beginning with Integration Server 9.6, you can create and run multiple Integration Server instances under a single installation directory. Each Integration Server instance has a home directory under *Integration Server_directory* \instances*instance_name* that contains the packages, configuration files, log files, and updates for the instance.

For more information about running multiple Integration Server instances, see the *webMethods Integration Server Administrator's Guide* for your release.

This guide uses the packages_directory as the home directory in Integration Server classpaths. For Integration Server 9.6 and above, the packages_directory is *Integration Server_directory*\instances*instance_name*\packages directory.

Installing Adapter for Apache Kafka

Note:
If you are installing Adapter for Apache Kafka in a clustered environment, you must install the adapter on each Integration Server in the cluster, and each installation must be identical. For more information about working with Adapter for Apache Kafka in a clustered environment, see "Adapter for Apache Kafka in a Clustered Environment" on page 31

To install Adapter for Apache Kafka

1. Download Installer from the Empower Product Support website.

2. If you are installing the adapter on an existing Integration Server, shut down the Integration Server.

3. Start the Installer wizard.

4. Select the webMethods release that includes the Integration Server on which you want to install the adapter. For example, if you want to install the adapter on Integration Server 9.6, select the 9.6 release.

5. Specify the installation directory as follows:

   - If you are installing on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server.
   - If you are installing on an existing Integration Server and the adapter, specify the installation directory to use.

6. In the product selection list, select **Adapters** > **webMethods Adapter 9.6 for Apache Kafka**.

   If you are Integration Server using 9.6 and above, you can install the package in the default instance. In this case, Software AG Installer installs the adapter in both locations, *Integration Server_directory*`\packages` and the default instance packages directory located in *Integration Server_directory*`\instances\default\packages`.

7. To download the documentation for the adapter, go to Software AG Documentation website.

8. Copy the following JAR files from Apache Kafka's *`lib`* folder to `WmKafkaAdapter\code\jars` location:

   - zkclient-0.10.jar (Required for Apache Kafka 2.12-2.3.1 or lower versions)
   - kafka-tools-2.0.0.jar
   - kafka-clients-2.0.0.jar
   - kafka_2.12-2.0.0.jar

   Note:
   You can use the JAR files with versions specified or higher.

9. After the installation completes, close the Installer and start the host Integration Server.

Uninstalling Adapter for Apache Kafka

To uninstall Adapter for Apache Kafka

1. Shut down the host Integration Server. You do not need to shut down any other webMethods products or applications that are running on your machine.

2. Start Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server.

3. In the product selection list, select **Adapters**>**webMethods Adapter 9.6 for Apache Kafka**.

4. After Uninstaller completes, restart the host Integration Server.

Uninstaller removes all Adapter for Apache Kafka-related files that were installed. However, Uninstaller does not delete files created after you installed the adapter (for example, user-created or configuration files), nor does it delete the adapter directory structure. You can go to the *Integration Server_directory*`\packages` directory and *Integration Server_directory*`\instances\default\packages directory`. Delete the `WmKafkaAdapter` directory.

3 Package Management

Overview of Package Management

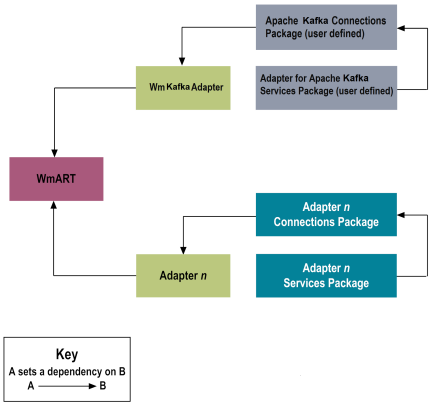
The following sections describe how to set up and manage your webMethods Adapter for Apache Kafka packages and to set up Access Control Lists (ACLs).

Adapter for Apache Kafka Package Management

Adapter for Apache Kafka is provided as a package called WmKafkaAdapter. You can manage the WmKafkaAdapter package as you would manage any package on webMethods Integration Server.

When you create connections, and adapter services, define them in user-defined packages rather than in the WmKafkaAdapter package. Doing so will allow you to manage the package more easily.

As you create user-defined packages in which to store connections, and adapter services, use the package management functionality provided in webMethods Deployer and set the user-defined packages to have a dependency on the WmKafkaAdapter package. That way, when the WmKafkaAdapter package loads or reloads, the user-defined packages load automatically. See the following diagram:



Package management tasks include:

- Setting package dependencies (see "Package Dependency Requirements and Guidelines" on page 28)
- "Enabling Packages" on page 29
- "Importing and Exporting Packages" on page 30
- "Group Access Control" on page 31

Package Dependency Requirements and Guidelines

This section contains a list of dependency requirements and guidelines for user-defined packages. For instructions for setting package dependencies, see the *webMethods Service Development Help* for your release.

- A user-defined package must have a dependency on its associated adapter package, WmKafkaAdapter. (The WmKafkaAdapter package has a dependency on the WmART package.)
- Package dependencies ensure that at start-up the Integration Server automatically loads or reloads all packages respectively: the WmART package, the adapter package, and the user-defined packages. The WmART package is automatically installed when you install Integration Server. You should not need to manually reload the WmART package.
- If the connections and adapter services of an adapter are defined in different packages, then:
  - A package that contains the connections must have a dependency on the adapter package.
  - Packages that contain adapter services must have a dependency on their associated connection package.
- Keep connections for different adapters in separate packages so that you do not create interdependencies between adapters. If a package contains connections for two different adapters, and you reload one of the adapter packages, the connections for both adapters will reload automatically.
- Integration Server will not allow you to enable a package if it has a dependency on another package which is disabled. That is, before you can enable your package, you must enable all packages on which your package depends. For information about enabling packages, see "Enabling Packages" on page 29.
- Integration Server will allow you to disable a package even if another package that is enabled has a dependency on it. Therefore, you must manually disable any user-defined packages that have a dependency on the adapter package before you disable the adapter package. For information about disabling packages, see "Disabling Packages" on page 30.
- You can provide same name for the connections, and adapter services, if they are in different folders and packages.

Enabling Packages

All packages are automatically enabled by default. Use the following procedure when you want to enable a package that was previously disabled.

To enable a package

1. Open Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the navigation area, click **Management.**
3. Click **No** in the **Enabled** column. The server displays **Yes** in the **Enabled** column.

> **Note:**
> Enabling an adapter package will not cause its associated user-defined packages to be reloaded.

Important:
Before you manually enable a user-defined package, you must first enable its associated adapter package (WmKafkaAdapter).

Disabling Packages

When you want to temporarily prohibit access to the elements in a package, disable the package. When you disable a package, the server unloads all of its elements from memory. Disabling a package prevents Integration Server from loading that package at startup.

Important:
If your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package (WmKafkaAdapter) first. Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages.

To disable a package

1. Open Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the navigation area, click **Management**.
3. Click **Yes** in the **Enabled** column for the package that you want to disable. The server issues a prompt to verify that you want to disable the package. Click **OK** to disable the package. When the package is disabled, the server displays **No** in the **Enabled** column.

A disabled adapter will:

- Remain disabled until you explicitly enable it using Integration Server Administrator.
- Not be listed in **Designer**.

Importing and Exporting Packages

You can import and export packages using Software AG Designer. Exporting allows you to export the package to a .zip file and save it to your hard drive. The .zip file can then be imported for use by another package.

Important:
Do not rename packages that you export; the rename function is comparable to moving a package, and when you import the renamed package, you can lose any triggers, and connections, associated with this package.

For details about importing and exporting packages, see the *webMethods Service Development Help* for your release.

Group Access Control

To control which groups have access to which adapter services, use access control lists (ACLs). For example, you can use ACLs to prevent one development group from inadvertently updating the work of another group, or to allow or deny access to services that are restricted to one group but not to others.

For information about assigning and managing ACLs, see the *webMethods Service Development Help* for your release.

Adapter for Apache Kafka in a Clustered Environment

Clustering is an advanced feature of the webMethods product suite that substantially extends the reliability, availability, and scalability of Integration Server. Clustering accomplishes this by providing the infrastructure and tools to deploy multiple Integration Server as if they were a single virtual server and to deliver applications that leverage that architecture. Because this activity is transparent to the client, clustering makes multiple servers look and behave as one.

Integration Server 9.0 and higher supports the caching and clustering functionality provided by Terracotta. Caching and clustering are configured at the Integration Server level and Adapter for Apache Kafka uses the caching mechanism that is enabled on Integration Server. Adapter for Apache Kafka does not explicitly implement any clustering or caching beyond what is already provided by Integration Server.

With clustering, you get the following benefits:

- **Load balancing**. This feature, provided automatically when you set up a clustered environment, allows you to spread the workload over several servers, thus improving performance and scalability.
- **Failover support**. Clustering enables you to avoid a single point of failure. If a server cannot handle a request, or becomes unavailable, the request is automatically redirected to another server in the cluster.

  Note:
  Integration Server clustering redirects HTTP and HTTPS requests, but does not redirect FTP or SMTP requests.
- **Scalability**. You can increase your capacity even further by adding new machines running Integration Server to the cluster.

For details on Integration Server clustering, see the *webMethods Integration Server Clustering Guide* for your release.

Adapter Service Support in Clusters

Adapter services are supported in a clustered environment. In order for a cluster to handle requests identically, you should be sure that the identical service is in each server in the cluster, so that if a given service is not available, the request can be redirected and handled by another server in the cluster.

For more details about adapter services in clusters, see "Clustering Considerations and Requirements" on page 32.

Replicating Packages to Integration Server

Every Integration Server in the cluster should contain an identical set of packages that you define using Adapter for Apache Kafka ; that is, you should replicate the Adapter for Apache Kafka services, and the connections they use.

To ensure consistency, we recommend that you create all packages on one server, and replicate them to the other servers. If you allow different servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides in only one server, and that server is unavailable, the request cannot be successfully redirected to another server.

For information about replicating packages, see the *webMethods Integration Server Administrator's Guide* for your release.

Clustering Considerations and Requirements

Note:
The following sections assume that you have already configured the Integration Server cluster. For details about webMethods clustering, see the *webMethods Integration Server Clustering Guide* for your release and *webMethods Integration Server Administrator's Guide*, if they are not already running.

The following considerations and requirements apply to Adapter for Apache Kafka in a clustered environment.

Requirements for Each Integration Server in a Cluster

The following table describes the requirements of each Integration Server in a given cluster:

All Integration Servers in a given cluster must have identical...	For Example
Integration Server versions	All Integration Server's in the cluster must be the same version, with the same service packs and fixes applied.
Adapter packages	All adapter packages on one Integration Server should be replicated to all other Integration Servers in the cluster.
Adapter connections	If you configure a connection to the database, this connection must appear on all servers in the cluster so that any Integration Server in the cluster can handle a given request identically.

All Integration Servers in a given cluster must have identical...	For Example
	If you plan to use connection pools in a clustered environment, see "Considerations When Configuring Connections with Connection Pooling Enabled" on page 33
Adapter services	If you configure a specific Retrieve Adapter Service, this same adapter service must appear on all servers in the cluster so that any Integration Server in the cluster can handle the request identically. If you allow different Integration Servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides on only one server, and that server is unavailable, the request cannot be successfully redirected to another server.

For information about replicating adapter packages, connections, adapter services across multiple Integration Server in a cluster, see "Replicating Packages to Integration Server" on page 32

Considerations When Installing Adapter for Apache Kafka Packages

For each Integration Server in the cluster, use the standard Adapter for Apache Kafka installation procedures for each machine, as described in "Overview of installing and uninstalling Adapter for Apache Kafka" on page 24

Considerations When Configuring Connections with Connection Pooling Enabled

When you configure a connection that uses connection pools in a clustered environment, be sure that you do not exceed the total number of connections that can be opened simultaneously for that database.

For example, if you have a cluster of two Integration Servers with a connection configured to a database that supports a maximum of 100 connections opened simultaneously, the total number of connections possible at one time must not exceed 100. This means that you cannot configure a connection with an initial pool size of 100 and replicate the connection to both servers, because there could be possibly a total of 200 connections opened simultaneously to this database.

In another example, consider a connection configured with an initial pool size of 10 and a maximum pool size of 100. If you replicate this connection across a cluster with two Integration Servers, it is possible for the connection pool size on both servers to exceed the maximum number of database connections that can be open at one time.

For information about configuring connections forAdapter for Apache Kafka, see "Overview of Adapter for Apache Kafka Connections" on page 36.

For more general information about connection pools, see the *webMethods Integration Server Administrator's Guide* for your release.

Note:
To avoid receiving duplicate messages, Consumer Connection in the cluster must be configured with same **groupId**.

4 Adapter for Apache Kafka Connections

Overview of Adapter for Apache Kafka Connections

This chapter describes how to configure and manage Adapter for Apache Kafka connections. For more information about how adapter connections work, see "Adapter Connections" on page 16

Before Configuring or Managing Adapter Connection

Perform the following steps before configuring or managing adapter connections.

To prepare to configure or manage adapter connections

1. Install webMethods Integration Server and Adapter for Apache Kafka on the same machine. For details, see "Overview of installing and uninstalling Adapter for Apache Kafka" on page 24.

2. Make sure you have Integration Server administrator privileges so that you can access Adapter for Apache Kafka's administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Start your Integration Server and Integration Server Administrator, if they are not already running.

4. Using Integration Server Administrator, make sure the WmKafkaAdapter package is enabled. For instructions, see "Enabling Packages" on page 29.

5. Using Software AG Designer, create a user-defined package to contain the connection, if you have not done so. For more information about managing packages for the adapter, see "Overview of Package Management" on page 28.

Configuring an Adapter for Apache Kafka Producer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Apache Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure Producer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Apache Kafka Producer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Bootstrap Servers	The list of servers with the combination of host and port used for establishing the initial connection to the Apache Kafka cluster. This list should be in the form, `host1:port1,host2:port2` and so on.
Acknowledgments	The number of acknowledgments the producer requires the leader to have received before considering a request as complete. This controls the durability of messages that are sent. The values allowed and their descriptions are as follows: ■ If set to `0`, the producer will not wait for any acknowledgment from the server. The message will be immediately added to the socket buffer and considered as sent. No guarantee can be made that the server has received the message. The configuration will not take effect if you retry. The offset given back for each message will always be set to `-1`. ■ If set to `1`, the leader will write the message to its local log, but will respond without awaiting for the full acknowledgement from all followers. In this case should the leader fail immediately after acknowledging the record but before the followers have replicated it, then the message is lost. ■ If set to `all`, the leader will wait for the full set of *in-sync* replicas to acknowledge the message. This guarantees that the message will not

Field	Description/Action
	be lost as long as at least one *in-sync* replica remains alive. This is the strongest available guarantee.
Request Timeout(ms)	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses, the client will resend the request if necessary or fail the request if retries are exhausted
Value Serializer Class	Defines the serializer class for value that implements the `org.apache.kafka.common.serialization.Serializer` interface. By default, `com.wm.adapter.wmkafka.idata.IDataSerializer` is used.
Key Serializer Class	Defines the serializer class for key that implements the `org.apache.kafka.common.serialization.Serializer` interface.
Partitioner Class	Defines the partitioner class that implements the Partitioner interface.
Compression Type	Defines the compression type for all data generated by the producer.
Message Send Retries	Causes the client to resend any message whose send fails with a potentially transient error, if the value greater than zero.
Batch Size	The producer will attempt to batch messages together into fewer requests whenever multiple messages are being sent to the same partition. This helps performance on both the client and the server. This configuration controls the default batch size in bytes.
Send Buffer Bytes	The size of the TCP send buffer `SO_SNDBUF` to use when sending data. If the value is set to `-1`, then the default OS will be used.
Client ID	Defines the ID string to pass to the Kafka server when you make requests.
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value`

6. In the **Connection Management Properties** section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. For more information about connection pooling, see "Adapter Connections" on page 16 . **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.

Field	Description/Action
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection with the database before it times out and returns an error. For example, you have a pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the **Block Timeout** to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing.

Field	Description/Action
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default value is `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

   The connection you created appears on the adapter's Connections screen and in Designer.

   You can enable a connection only if the parameters for the connection are valid.

Configuring a Secured Adapter for Apache Kafka SASL Producer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Apache Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure SASL Producer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Apache Kafka SASL Producer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28

Field	Description/Action
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Bootstrap Servers	The list of servers with the combination of host and port used for establishing the initial connection to the Apache Kafka cluster. This list should be in the form, `host1:port1,host2:port2` and so on.
Acknowledgments	The number of acknowledgments the producer requires the leader to have received before considering a request as complete. This controls the durability of messages that are sent. The values allowed and their descriptions are as follows: ■ If set to `0`, the producer will not wait for any acknowledgment from the server. The message will be immediately added to the socket buffer and considered as sent. No guarantee can be made that the server has received the message. The configuration will not take effect if you retry. The offset given back for each message will always be set to `-1`. ■ If set to `1`, the leader will write the message to its local log, but will respond without awaiting for the full acknowledgement from all followers. In this case should the leader fail immediately after acknowledging the record but before the followers have replicated it, then the message is lost. ■ If set to `all`, the leader will wait for the full set of *in-sync* replicas to acknowledge the message. This guarantees that the message will not be lost as long as at least one *in-sync* replica remains alive. This is the strongest available guarantee.
Request Timeout(ms)	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses, the client will resend the request if necessary or fail the request if retries are exhausted
Value Serializer Class	Defines the serializer class for value that implements the `org.apache.kafka.common.serialization.Serializer` interface. By default, `com.wm.adapter.wmkafka.idata.IDataSerializer` is used.

Field	Description/Action
Key Serializer Class	Defines the serializer class for key that implements the `org.apache.kafka.common.serialization.Serializer` interface.
Partitioner Class	Defines the partitioner class that implements the Partitioner interface.
Security Protocol	Defines the protocol which is used to communicate with brokers. The valid values are: SSL, SASL_PLAINTEXT, SASL_SSL.
Kerberos Service Name	Defines the Kerberos principal name that Apache Kafka runs as. The name can be defined either in Apache Kafka's JAAS Config or in Apache Kafka's configuration.
JAAS Config	Specify the JAAS login context parameters for SASL connections in the format used by JAAS configuration files.
Retype JAAS Config	Retype the JAAS login context parameters for SASL connections in the format used by JAAS configuration files.
Truststore Alias	Specify the alias name created for truststore using IS/security/keystore/Create Truststore Alias.
Keystore Alias	Specify the alias name created for keystore using IS/security/keystore/Create Keystore Alias.
Compression Type	Defines the compression type for all data generated by the producer.
Message Send Retries	Causes the client to resend any message whose send fails with a potentially transient error, if the value greater than zero.
Batch Size	The producer will attempt to batch messages together into fewer requests whenever multiple messages are being sent to the same partition. This helps performance on both the client and the server. This configuration controls the default batch size in bytes.
Send Buffer Bytes	The size of the TCP send buffer `SO_SNDBUF` to use when sending data. If the value is set to `-1`, then the default OS will be used.
Client ID	Defines the ID string to pass to the Kafka server when you make requests.
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value`
Kafka Version	Specify the Apache Kafka version that the producer needs to connect to. The following are the supported values: ■ `v9` This is equivalent to Apache Kafka `0.9.0.x` version.

Field	Description/Action
	■ `v10` This is equivalent to Apache Kafka `0.10.x.x` version. ■ `v11+` This is equivalent to Apache Kafka `0.11.x.x` version and later.

Note:

- Be sure to add the JAAS entry with name KafkaClient for Kerberos authentication, if JAAS configuration is not defined at the connection level and the Security Protocol is SASL_PLAINTEXT/SASL_SSL. An example to add the JAAS entry in the *Integration Server_directory*`/instances/`*`IS_Instance`*`/config/is_jaas.cnf` file is as follows:

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
useTicketCache=true
renewTicket=true
keyTab="/opt/webMethods9/wm5511/kerberos/ta2coab.keytab"
serviceName="kafka"
principal="xxxxxxxxxxxxxxxxxxxxx";
};
```

- If JAAS Config is configured both at connection level and at is_jaas.cnf file, then the connection level JAAS Config will be considered. Example, `keyTab="`*`directory1/directory2`*`.keytab"`.

For more information on about configuring keystore and truststore aliases and securing communication with Integration Server, see the *webMethods Integration Server Administrator's Guide* of your release.

6. In the **Connection Management Properties** section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. For more information about connection pooling, see "Adapter Connections" on page 16 . **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.

Field	Description/Action
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection with the database before it times out and returns an error. For example, you have a pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the **Block Timeout** to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default value is `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

Note:
The created connection will appear in the adapter's Connections screen and in Designer.

You can enable a connection only if the parameters for the connection are valid.

This connection type will be available from Fix 2 and later.

Configuring an Adapter for Apache Kafka Consumer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Apache Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure Consumer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Apache Kafka Consumer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Bootstrap Servers	The list of servers with the combination of host and port used for establishing the initial connection to the Apache Kafka cluster. This list should be in the form, `host1:port1,host2:port2` and so on.
groupId	Defines a unique string that identifies the consumer group that this consumer connection belongs to.
Client ID	Defines the ID string to pass to the Kafka server when a request is made.
Key Deserializer Class	Defines the deserializer class for key that implements `org.apache.kafka.common.serialization.Deserializer` interface.
Value Deserializer Class	Defines the deserializer class for value that implements the `org.apache.kafka.common.serialization.Deserializer` interface. The default value is, `com.wm.adapter.wmkafka.idata.IDataDeserializer`
Enable Auto Commit	Select one of the following values for this field to enable the auto commit. ■ If the value is set to `true`, the consumer's offset is periodically committed. ■ If the value is set to `false`, the Adapter for Apache Kafka commits the consumer's offsets.
Auto Commit Interval(ms)	Specifies the frequency in milliseconds that the consumer offsets are auto-committed to Kafka, if `enable.auto.commit` is set to `true`.
Session TimeOut(ms)	Specifies the timeout used to detect consumer failures while using the Kafka's group management facility.
Auto Offset Reset	Select one of the following values if no initial offset value is set in Apache Kafka: ■ **earliest** Automatically reset the offset to the earliest offset ■ **latest** Automatically reset the offset to the latest offset ■ **none** Throws exceptions to the consumer if no previous offset is found for the consumer's group
Kafka Version	Specify the Apache Kafka version that the consumer needs to connect to. The following are the supported values: ■ `v9` This is equivalent to Apache Kafka `0.9.0.x` version. ■ `v10` This is equivalent to Apache Kafka `0.10.x.x` version.
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value`

6. In the **Connection Management Properties** section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. For more information about connection pooling, see "Adapter Connections" on page 16 . **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection with the database before it times out and returns an error. For example, you have a pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the **Block Timeout** to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter.

Field	Description/Action
	If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default value is `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

   The connection you created appears on the adapter's Connections screen and in Designer.

   You can enable a connection only if the parameters for the connection are valid.

Configuring a Secured Adapter for Apache Kafka SASL Consumer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Apache Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure SASL Consumer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Apache Kafka SASL Consumer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Bootstrap Servers	The list of servers with the combination of host and port used for establishing the initial connection to the Apache Kafka cluster. This list should be in the form, `host1:port1,host2:port2` and so on.
groupId	Defines a unique string that identifies the consumer group that this consumer connection belongs to.
Client ID	Defines the ID string to pass to the Kafka server when a request is made.
Key Deserializer Class	Defines the deserializer class for key that implements `org.apache.kafka.common.serialization.Deserializer` interface.
Value Deserializer Class	Defines the deserializer class for value that implements the `org.apache.kafka.common.serialization.Deserializer` interface. The default value is, `com.wm.adapter.wmkafka.idata.IDataDeserializer`
Enable Auto Commit	Select one of the following values for this field to enable the auto commit. ■ If the value is set to `true`, the consumer's offset is periodically committed. ■ If the value is set to `false`, the Adapter for Apache Kafka commits the consumer's offsets.

Field	Description/Action
Auto Commit Interval(ms)	Specifies the frequency in milliseconds that the consumer offsets are auto-committed to Kafka, if `enable.auto.commit` is set to `true`.
Security Protocol	Defines the protocol which is used to communicate with brokers. The valid values are: SSL, SASL_PLAINTEXT, SASL_SSL.
Kerberos Service Name	Defines the Kerberos principal name that Apache Kafka runs as. The name can be defined either in Apache Kafka's JAAS Config or in Apache Kafka's configuration.
JAAS Config	Specify the JAAS login context parameters for SASL connections in the format used by JAAS configuration files.
Retype JAAS Config	Retype the JAAS login context parameters for SASL connections in the format used by JAAS configuration files.
Truststore Alias	Specify the alias name created for truststore using the path, `IS/security/keystore/Create Truststore Alias`.
Keystore Alias	Specify the alias name created for keystore using the path, `IS/security/keystore/Create Keystore Alias`.
Session TimeOut(ms)	Specifies the timeout used to detect consumer failures while using the Kafka's group management facility.
Auto Offset Reset	Select one of the following values if no initial offset value is set in Apache Kafka: ■ **earliest** Automatically reset the offset to the earliest offset ■ **latest** Automatically reset the offset to the latest offset ■ **none** Throws exceptions to the consumer if no previous offset is found for the consumer's group
Kafka Version	Specify the Apache Kafka version that the producer needs to connect to. The following are the supported values: ■ `v9` This is equivalent to Apache Kafka `0.9.0.x` version. ■ `v10` This is equivalent to Apache Kafka `0.10.x.x` version. ■ `v11+` This is equivalent to Apache Kafka `0.11.x.x` version and later.
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value`

Note:

- Be sure to add the JAAS entry with name KafkaClient for Kerberos authentication, if JAAS configuration is not defined at the connection level and the Security Protocol is SASL_PLAINTEXT/SASL_SSL. An example to add the JAAS entry in the *Integration Server_directory*/instances/*IS_Instance*/config/is_jaas.cnf file is as follows:

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
useTicketCache=true
renewTicket=true
keyTab="/opt/webMethods9/wm5511/kerberos/ta2coab.keytab"
serviceName="kafka"
principal="xxxxxxxxxxxxxxxxxxxxx";
};
```

- If JAAS Config is configured both at connection level and at is_jaas.cnf file, then the connection level JAAS Config will be considered. Example, keyTab="*directory1/directory2*.keytab".

For more information on about configuring keystore and truststore aliases and securing communication with Integration Server, see the *webMethods Integration Server Administrator's Guide* of your release.

6. In the **Connection Management Properties** section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. For more information about connection pooling, see "Adapter Connections" on page 16 . **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection with the database before it times out and returns an error. For example, you have a pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the **Block**

Field	Description/Action
	Timeout to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default value is `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

Note:
The created connection will appear in the adapter's Connections screen and in Designer.

You can enable a connection only if the parameters for the connection are valid.

This connection type will be available from Fix 2 and later.

Custom Serializer

Kafka stores and transports byte arrays in its topics. If you wish to use the Java Objects instead of byte arrays, Kafka provides interfaces to accomplish it.

To implement a serializer, create a class that implements the `org.apache.kafka.common.serialization.Serializer` interface.

To implement a deserializer, create a class that implements the `org.apache.kafka.common.serialization.Deserializer` interface.

To implement serializer and deserializer, you need to implement the following methods:

- configure()
- close()
- serialize() / deserialize()

Serializing MyMessage in Producer

You create a serializer class that implements `org.apache.kafka.common.serialization.Serializer`.

`serialize()` receives your object and returns a serialized version as bytes array.

For example, Serializing MyMessage object in Producer is as follows:

```
public class MyValueSerializer implements Serializer<MyMessage>
{
    private boolean isKey;
    @Override
    public void configure(Map<String, ?> configs, boolean isKey)
    {
        this.isKey = isKey;
    }
    @Override
    public byte[] serialize(String topic, MyMessage message)
    {
        if (message == null) {
            return null;
        }
        try {
            (serialize your MyMessage object into bytes)
            return bytes;
        } catch (IOException | RuntimeException e) {
            throw new SerializationException("Error serializing value", e);
        }
    }
    @Override
    public void close()
    {
    }
}
```

Deserializing MyMessage in Consumer

You create a deserializer class that implements
`org.apache.kafka.common.serialization.Deserializer`

`deserialize()` receives serialized value as bytes array and returns your object.

For example, Deserializing MyMessage object in Consumer is as follows:

```
public class MyValueDeserializer implements Deserializer<MyMessage>
{
    private boolean isKey;
    @Override
    public void configure(Map<String, ?> configs, boolean isKey)
    {
        this.isKey = isKey;
    }
    @Override
    public MyMessage deserialize(String s, byte[] value)
    {
        if (value == null) {
            return null;
        }
        try {
            (deserialize value into your MyMessage object)
            MyMessage message = new MyMessage();
            return message;
        } catch (IOException | RuntimeException e) {
            throw new SerializationException("Error deserializing value", e);
        }
    }
    @Override
    public void close()
    {
    }
}
```

Note:
Like Kafka message *value,* Kafka message *key* also follows the same process in serializing and deserializing the objects.

How to integrate custom serializer/deserializer in Adapter for Apache Kafka

1. Place the jar file containing custom serializer/deserializer classes under *Integration Server_directory*\instances*instance_name*\packages\WmKafkaAdapter\code\jars.

2. Configure the fully qualified class name of the customer serializer class for the connection parameter value serializer class/key serializer class in the Producer connection page.

3. Configure the fully qualified class name of the customer deserializer class for the connection parameter value deserializer class/key deserializer class in the Consumer connection page

IData doctypes Serialization

Adapter for Apache Kafka supports Serialization and Deserialization for the following IData doctypes:

- Serializer class:- `com.wm.adapter.wmkafka.idata.IDataSerializer`
- DeSerializer class:- `com.wm.adapter.wmkafka.idata.IDataDeserializer`

Connecting to SSL Enabled Apache Kafka Cluster

When the Kafka cluster is enabled with secured socket layer (SSL)/TLS, the following configurations have to be passed in Producer Connection and Consumer Connection configurations:

- If the client authentication is not required in the Kafka cluster, then the following configurations need to be passed under **Other Properties** in the Connection Configuration screen.

```
security.protocol=SSL
ssl.truststore.location=/var/private/ssl/kafka.client.truststore.jks
//above provided is a sample path//
ssl.truststore.password=test1234
//above provided is a sample password//
```

- If client authentication is required in the Kafka cluster, then the following configurations need be passed under **Other Properties** in the Connection Configuration screen.

```
security.protocol=SSL
ssl.truststore.location=/var/private/ssl/kafka.client.truststore.jks
//above provided is a sample path//
ssl.truststore.password=test1234
ssl.keystore.location=/var/private/ssl/Kafka.client.keystore.jks
//above provided is a sample path//
ssl.keystore.password=test1234  //provided is a sample password//
ssl.key.password=test1234   //provided is a sample password//
```

Overview of Confluent Kafka Connections

This section describes how to configure Confluent Kafka services.

Configuring Confluent Distribution of Kafka

Adapter for Apache Kafka supports the Confluent platform for Apache Kafka. For Avro Serialization and Schema Registry configuration changes that are applicable if you are using a non-confluent connection type, you must perform the following:

1. Copy the required libraries to the *Integration Server_directory*`\instances\`*instance_name*`\packages\code\jars` directory. The list of libraries to copy are as follows:
   - avro-1.10.1.jar
   - common-config-6.2.0.jar

- common-utils-6.2.0.jar
- jakarta.ws.rs-api-2.1.6.jar
- jersey-client-2.30.jar
- jersey-common-2.30.jar
- kafka_2.13-6.2.0-ccs.jar
- kafka-avro-serializer-6.2.0.jar
- kafka-clients-6.2.0-ccs.jar
- kafka-schema-registry-client-6.2.0.jar
- kafka-schema-serializer-6.2.0.jar
- kafka-tools-6.2.0-ccs.jar
- wm-kafka-v9.jar
- zookeeper-3.5.9.jar

Note:
You can use the JAR files with versions specified or higher.

2. Set the following fields for Adapter for Apache Kafka Producer Connection and SASL Producer Connection type.
   - Set the **Other Properties** to `schema.registry.url=`*`http://host:port`*. For example:
     ```
     schema.registry.url=http://vmkavp02:8091
     ```
   - Set the **Value Serializer Class** to `io.confluent.kafka.serializers.KafkaAvroSerializer`.
3. Set the following fields for Adapter for Apache Kafka Consumer Connection and SASL Consumer Connection type.
   - Set the **Other Properties** to `schema.registry.url=`*`http://host:port`*`;specific.avro.reader=true`. For example:
     ```
     schema.registry.url=http://vmkavp02:8091;specific.avro.reader=true
     ```
   - Set the **Value Deserializer Class** to `io.confluent.kafka.serializers.KafkaAvroDeserializer`.

Configuring an Adapter for Confluent Kafka Producer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Apache Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure Producer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Confluent for Kafka Producer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Confluent Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help*for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Schema Registry URL	Defines a comma-separated list of URLs, that can be used to register or look up schemas.
Schema Package	Specifies the package that consists of AVRO schema files. AVRO files should be under sub-folder named as schemas in the specified package directory.
Schema Registry Credentials	Specifies the security credentials that you use when connecting to a Schema Registry.

Field	Description/Action

Credential Source	Description
USER_INFO	Set the following fields: ■ **Schema Registry Credentials** to *API key: API secret*. Specify the pair of API key and secret value, that you use when connecting to a Schema Registry. ■ **Security Protocol** to `SASL_SSL` ■ **Other Properties** to `basic.auth.credentials.source=USER_INFO`. **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertName2=value`
SASL_INHERIT	Set the following fields: ■ **Security Protocol** to `SASL_SSL` ■ **JAAS_CONFIG** to the JAAS login context parameters for SASL connections in the format used by JAAS configuration files. **Note:** If **JAAS_CONFIG** is specified, the value specified in **Schema Registry Credentials** field is not considered when connecting to a Schema Registry. ■ **Other Properties** to `basic.auth.credentials.source=SASL_INHERIT`. **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertyName2=value`
STATIC_TOKEN	Set the following fields: ■ **Schema Registry Credentials** to *OAuth token*. Specify the OAuth token. ■ **Security Protocol** to `SASL_SSL` ■ **Other Properties** to `bearer.auth.credentials.source=STATIC_TOKEN`.

Field	Description/Action
	Credential Source — **Description** **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertName2=value`
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value` Some of the Kafka related properties that can be set in the **Other Properties** field are: ■ `schema.registry.url.conn.timeout`. Specifies the timeout (milliseconds) used to detect if the Schema Registry URL is reachable. Default value is `60000` milliseconds. The range is between `1000` and `300000` Example of **Other Properties**: `schema.registry.url.conn.timeout=75000;max.poll.records=10`

For configuring other **Connection Properties** and **Connection Management Properties** section, refer "Configuring a Secured Adapter for Apache Kafka SASL Producer Connection" on page 40

6. Click **Save Connection**.

   The connection you created appears on the adapter's Connections screen and in Designer.

   You can enable a connection only if the parameters for the connection are valid.

Configuring an Adapter for Confluent Kafka Consumer Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Confluent Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To configure Consumer connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Confluent for Kafka Consumer Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Confluent Kafka** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description/Action
Schema Registry URL	Defines a comma-separated list of URLs, that can be used to register or look up schemas.
Schema Package	Specifies the package that consists of AVRO schema files. AVRO files should be under schemas folder in the specified package directory.
Schema Registry Credentials	Specifies the security credentials that you use when connecting to a Schema Registry.

Credential Source	Description
USER_INFO	Set the following fields: ■ **Schema Registry Credentials** to *API key: API secret*. Specify the pair of API key and secret value, that you use when connecting to a Schema Registry.

Field	Description/Action

Credential Source	Description
	■ **Security Protocol** to `SASL_SSL` ■ **Other Properties** to `basic.auth.credentials.source=USER_INFO`. **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertName2=value`
SASL_INHERIT	Set the following fields: ■ **Security Protocol** to `SASL_SSL` ■ **JAAS_CONFIG** to the JAAS login context parameters for SASL connections in the format used by JAAS configuration files. **Note:** If **JAAS_CONFIG** is specified, the value specified in **Schema Registry Credentials** field is not considered when connecting to a Schema Registry. ■ **Other Properties** to `basic.auth.credentials.source=SASL_INHERIT`. **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertName2=value`
STATIC_TOKEN	Set the following fields: ■ **Schema Registry Credentials** to *OAuth token*. Specify the OAuth token. ■ **Security Protocol** to `SASL_SSL` ■ **Other Properties** to `bearer.auth.credentials.source=STATIC_TOKEN`. **Note:** You can specify additional configuration in **Other Properties**. Use the following format: `propertyName1=value;propertName2=value`

Field	Description/Action
Other Properties	Specifies the Kafka related properties for additional configuration. Use the following format: `propertyName1=value;propertName2=value` Some of the Kafka related properties that can be set in the **Other Properties** field are: ■ `schema.registry.url.conn.timeout`. Specifies the timeout (milliseconds) used to detect if the Schema Registry URL is reachable. Default value is `60000` milliseconds. The range is between `1000` and `300000` Example of **Other Properties**: `schema.registry.url.conn.timeout=75000;max.poll.records=10`

For configuring other **Connection Properties** and **Connection Management Properties** section, refer "Configuring a Secured Adapter for Apache Kafka SASL Consumer Connection" on page 48

6. Click **Save Connection**.

   The connection you created appears on the adapter's Connections screen and in Designer.

   You can enable a connection only if the parameters for the connection are valid.

Adapter for Confluent KSQL

KSQL allows to build event streaming applications from Apache Kafka® topics by using SQL statements and queries. KSQL is built on Kafka Streams, so a KSQL application communicates with a Kafka cluster like any other Kafka Streams application.

Using Adapter for Apache Kafka, Integration Server clients can create and run services to perform KSQL operations such as create streams or tables, select from streams or tables, insert values into streams or tables and so on.

Configuring an Adapter for Confluent KSQL Connection

When you configure Adapter for Apache Kafka , you specify information that Integration Server uses to connect to an Confluent Kafka system. You can configure Adapter for Apache Kafka connections manually using the Integration Server Administrator screen.

To Configure KSQL Connection

1. In the **Adapters** menu of Integration Server Administrator's navigation area, click **webMethods Adapter for Apache Kafka**.

2. On the Connections screen, click **Configure New Connection**.

3. On the Connection Types screen, click **Confluent for Kafka KSQL Connection** to display the Configure Connection Type screen.

4. In the **webMethods Adapter for Apache Kafka** section, use the following fields:

Field	Description
Package	The package in which to create the connection. You must create the package using Designer before you can specify the package using this parameter. For general information about creating packages, see *webMethods Service Development Help*. **Note:** Configure the connection in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka see, "Overview of Package Management" on page 28
Folder Name	Specifies the folder in which you create the connection.
Connection Name	Specifies the name you want to give to the connection. Connection names cannot have spaces or use special characters reserved by Integration Server and Designer. For more information about the use of special characters in package, folder, and element names, see *webMethods Service Development Help*.

5. In the **Connection Properties** section, use the following fields:

Field	Description
KSQL Server URL	Defines the KSQL server REST endpoint URL.
User Name	Defines the username if the KSQL server is configured to use Basic authentication.
Password	Defines the password If the KSQL server is configured to use Basic authentication.
Connection Timeout	Defines the time(in milliseconds) to establish the connection with Server.
Read Timeout	Defines the time(in milliseconds) waiting for reading the data - after establishing the connection; maximum time of inactivity between two data packets
Keystore Alias	Specify the alias name created for keystore using IS/security/keystore/Create Keystore Alias.

Field	Description
Truststore Alias	Specify the alias name created for truststore using IS/security/keystore/Create Truststore Alias.
Verify Hostname	Specifies whether to verify hostname. If true then `org.apache.http.conn.ssl.DefaultHostnameVerifier` is used.

For more information on about configuring keystore and truststore aliases and securing communication with Integration Server, see the *webMethods Integration Server Administrator's Guide*

6. In the Connection Management Properties section, use the following fields:

Field	Description
Enabling Connection Pooling	Enables the connection to use connection pooling. For more information see, "Adapter Connections" on page 16. **Note:** If you plan to enable connection pooling in a clustered environment, consider the connection pool size.
Minimum Pool Size	If the connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that Integration Server will wait to obtain a connection with the KSQL Server before it times out and returns an error. For example, you have a pool with Maximum Pool Size of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the Block Timeout to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set

Field	Description
	the Block Timeout value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the Maximum Pool Size to accommodate such bursts in processing.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the Minimum Pool Size. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the Expire Timeout value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the Expire Timeout value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the Minimum Pool Size to avoid excessive opening/closing of connections during normal processing.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default value is 0.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

7. Click **Save Connection**.

Note:
The created connection will appear in the adapter's Connections screen and in Designer.

5 Adapter Services

Overview of Adapter Services

This chapter describes how to configure and manage Adapter for Apache Kafka services. For detailed descriptions of the available Adapter for Apache Kafka services, see "Adapter Services" on page 18

Before Configuring or Managing Adapter Services

Perform the following steps before configuring or managing adapter services.

To prepare to configure or manage Adapter for Apache Kafka services

1. Start the Integration Server and Integration Server Administrator, if they are not already running.

2. Make sure you have administrator privileges so that you can access the Adapter for Apache Kafka administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Using Integration Server Administrator, make sure the WmKafkaAdapter package is enabled. For instructions, see "Enabling Packages" on page 29.

4. Using Integration Server Administrator, configure an adapter connection to use with the adapter service. For instructions, see "Overview of Adapter for Apache Kafka Connections" on page 36.

5. Start Designer if it is not already running.

6. Using Designer, create a user-defined package to contain the services, if you have not already done so. When you configure adapter services, you should always define them in user-defined packages rather than in the WmKafkaAdapter package. For more information about managing packages for the adapter, see "Overview of Package Management" on page 28.

Configuring Produce Service

The Produce service delivers a message to a specified Kafka topic. A Produce Service creates a message object from the input signature properties, which becomes the input document. You specify input signature properties when you configure the Produce Service. In the input signature, if a valid partition number is specified, then that partition will be used when sending the message. If no partition is specified but a key is specified, then a partition will be chosen using a hash of the key. If neither key nor partition is specified, then a partition will be assigned in a round-robin fashion. You configure Adapter for Apache Kafka services using Designer.

To configure a Produce service

1. In Designer, right-click the package in which the service should be contained and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** of type **Apache Kafka Producer Connection** and click **Next**.

5. From the list of available templates, select the **Produce** service template and click **Finish.**

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

Signature of Produce Service Template

The input signature contains the following parameters:

Parameter	Description/Action
topicName	The particular Kafka topic to which a message is sent.
Key(optional)	The messages with similar key goes to a single partition.
message	Specifies the unit of information that needs to be sent.
partition(optional)	Specifies the partition in which the message is stored.
header	Specifies the list of key/values to be added as header in the record. **Note:** This parameter appears when the Kafka version of the producer connection is v11+.

The output signature contains the following parameters:

Parameter	Description/Action
topic	Specifies particular topic in which the message is appended to.
offset	Specifies the offset of the message in the Kafka topic or Kafka partition.
partition	Specifies the partition in which the messages are sent.

Configuring Consume Service

The Consume service retrieves and removes a message from a specified Kafka topic. When you configure the Consume service, you specify the output signature properties, which become the output document. You configure Adapter for Apache Kafka services using Designer. The adapter supports the following types of consumption:

- **Subscription**: Subscription is considered when input of the service contains only topics.
- **Manual Assignment**: Manual Assignment is considered when input of the service contains topics and partitions.

To configure a Consume service

1. In Designer, right-click the package in which the service should be contained and select **New** > **Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** of type **Apache Kafka Consumer Connection** and click **Next**.
5. From the list of available templates, select the **Consume** service template and click **Finish.**

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

Signature of Consume Service Template

The input signature contains the following parameters:

Parameter	Description/Action
topic	Specifies the topic(s) from which the messages is consumed.
partition(optional)	Specifies the partition from which the messages are consumed.
pollTime	Specifies the polling interval in milliseconds. Default value is `10000`.
offset(optional)	Specifies the offset of the partition from where the consumer needs to start consuming the messages.

The output signature contains the following parameters:

Parameter	Description/Action
topic	Specifies the topic(s) from which the messages are consumed.
offset	Specifies the position of the message in the corresponding Kafka partition.
partition	Specifies the partition from which the messages are consumed.
Key	Specifies the key of the consumed message
value	Specifies the value of the consumed message
header	Specifies the list of headers **Note:** This parameter appears when the Kafka version of the consumer connection is v11+.

Configuring Bulk Produce Service for Apache Kafka

The Bulk Produce service delivers a set of messages asynchronously to a specified Kafka topic and waits for the set of messages to complete. The Bulk Produce service returns a list of metadata containing the status of each message.

The Bulk Produce service creates message objects from input signature properties, which then becomes the input document. You specify input signature properties when you configure the Bulk Produce service.

Note:
If a valid partition is specified in the input signature, then that partition will be used when sending the message. If no partition is specified but a key is specified, then a partition will be chosen using a hash of the key. If neither key nor partition is specified, then a partition will be assigned in a round-robin fashion.

You configure Adapter for Apache Kafka services using Designer.

To configure a Bulk Produce service

1. In Designer, right-click the package where you want to create the service and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** of type **Apache Kafka Producer Connection** and click **Next**.

5. From the list of available templates, select the **Bulk Produce** service template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. In the **Bulk Publish** tab, configure the following fields:

Field	Description
Topic	Specifies the topic to which the message must be published.
Partition (optional)	Specifies the partition in which the message must be stored.
Maximum Message Size	Specifies the maximum number of messages that can be published.

Signature of Bulk Produce Service Template for Apache Kafka

The input signature contains the following parameters:

Parameter	Description/Action
`$topic`	Specifies the topic name where the specified value overrides the configured value in the template.
`$partition` (optional)	Specifies the partition in which the message is stored. The specified value overrides the configured value in the template.
`messages`	Specifies a set of messages with key, value, and header to be published.
`key`	Specifies the key to denote the partition in which the messages are added. Messages with the same key are added to a single partition.
`value`	Specifies the unit of information that needs to be sent.
`header`	Specifies the list of key-value pairs to be added as header in the record. **Note:** This parameter appears when the Apache Kafka version of the producer connection is version 11 or later.
`globalHeader`	Specifies the list of key-value pairs to be added as header in the record if individual messages do not have headers.

The output signature contains the following parameters:

Parameter		Description/Action
`result`		Specifies the metadata with the following fields of the corresponding message that has been published to Apache Kafka cluster.
	`isSuccess`	Specifies whether the message is sent successfully or not.
	`topic`	Specifies the topic in which the message is appended.
	`offset`	Specifies the offset of the message in the Kafka topic or Kafka partition.
	`partition`	Specifies the partition in which the messages are sent.
	`errorMessage`	Specifies the error if the message delivery fails.

Overview of Confluent Kafka Services

This section describes how to configure Confluent Kafka services.

Configuring Produce Service for Confluent Kafka

The Produce service delivers a message to a specified Kafka topic. A Produce Service creates a message object from the input signature properties, which becomes the input document. You specify input signature properties when you configure the Produce Service. In the input signature, if a valid partition number is specified, then that partition will be used when sending the message. If no partition is specified but a key is specified, then a partition will be chosen using a hash of the key. If neither key nor partition is specified, then a partition will be assigned in a round-robin fashion. You configure Adapter for Apache Kafka services using Designer.

To configure a Produce service

1. In Designer, right-click the package in which the service should be contained and select **New** > **Adapter Service** .

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** of type **Confluent Kafka Producer Connection** and click **Next**.

5. From the list of available templates, select the **Produce** service template and click **Finish**

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. Select **Publish with Schema** tab to configure the following fields:

Field	Description
Topic	Specifies the topic to which message is published. **Note:** The topic to which the message is published must be unique across the schema.
Partition(optional)	Specified partition in which the message is stored.
Key Schema	Specifies the avro based schema files.
Value Schema	Specifies the avro based schema files.

Note:
If the key serializer class or value serializer class is io.confluent.kafka.serializers.KafkaAvroSerializer in the connection parameters and avro files have been placed under schema package/schemas directory, then avro file names would be available for key schema/value schema.

Signature of Produce Service Template for Confluent Kafka

The input signature contains the following parameters:

Parameter	Description/Action
`$topic`	Specifies the topic name where the specified value overrides the configured value in the template.
`Key`	Specifies the input type as a document, if it's an avro schema file.
`value`	Specifies the input type as a document, if it's an avro schema file.
`$partition`(optional)	Specifies the partition name which will override the one configured in the template
`header`	Specifies the list of key/values to be added as header in the record.

The output signature contains the following parameters:

Parameter	Description/Action
`topic`	Specifies particular topic in which the message is appended to.
`offset`	Specifies the offset of the message in the Kafka topic or Kafka partition.
`partition`	Specifies the partition in which the messages are sent.

Configuring Consume Service for Confluent Kafka

The Consume service retrieves and removes a message from a specified Kafka topic. When you configure the Consume service, you specify the output signature properties, which become the output document. You configure Adapter for Apache Kafka services using Designer. The adapter supports the following types of consumption:

- **Subscription**: Subscription is considered when input of the service contains only topics.
- **Manual Assignment**: Manual Assignment is considered when input of the service contains topics and partitions.

To configure a Consume service

1. In Designer, right-click the package in which the service should be contained and select **New** > **Adapter Service** .
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** of type **Confluent Kafka Consumer Connection** and click **Next**.
5. From the list of available templates, select the **Consume** service template and click **Finish**

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
6. Select **Consume with Schema** tab to configure the following fields:

Field	Description
Topic	Specifies the topic(s) from which the messages are consumed. **Note:** The topic(s) from which the messages are consumed must be unique across the schema.
Partition(optional)	Specified partition in which the message is stored.
Offset field	Specifies the field name that specifies the offset of the partition from where the consumer needs to start consuming the messages.
Value Schema	Specifies the particular avro schema file name.
Key Schema	Specifies the particular avro schema file name.

Note:
If the key deserializer class or value deserializer class is io.confluent.kafka.serializers.KafkaAvroDeserializer in the connection parameters and avro files have been placed under schema package/schemas directory, then avro file names would be available for key schema/value schema.

Signature of Consume Service Template for Confluent Kafka

The input signature contains the following parameters:

Parameter	Description/Action
`$topic`	Specifies the name of the topic(s), that overrides the configured value in the **Consume with Schema** tab.
`pollTime`	Specifies the polling interval in milliseconds. Default value is `10000`.

The output signature contains the following parameters:

Parameter	Description/Action
`topic`	Overrides the topic(s) that is configured in **Consume with Schema** tab.
`offset`	Specifies the position of the message in the corresponding Kafka partition.
`partition`	Specifies the partition from which the messages are consumed.
`Key`	Specifies the input type as a document, if it's an avro schema file.
`value`	Specifies the input type as a document, if it's an avro schema file.
`header`	Specifies the list of headers of the consumed message.

Configuring Bulk Produce Service for Confluent Kafka

The Bulk Produce service delivers a set of messages asynchronously to a specified Kafka topic and waits for the set of messages to complete. The Bulk Produce service returns a list of metadata containing the status of each message.

The Bulk Produce service creates message objects from input signature properties, which then becomes the input document. You specify input signature properties when you configure the Bulk Produce service.

Note:
If a valid partition is specified in the input signature, then that partition will be used when sending the message. If no partition is specified but a key is specified, then a partition will be chosen using a hash of the key. If neither key nor partition is specified, then a partition will be assigned in a round-robin fashion.

You configure Adapter for Apache Kafka services using Designer.

To configure a Bulk Produce service

1. In Designer, right-click the package where you want to create the service and select **New** > **Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate **Adapter Connection Name** of type **Confluent for Kafka Producer Connection** and click **Next**.

5. From the list of available templates, select the **Produce** service template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. In the **Bulk Publish with Schema** tab, configure the following fields:

Field	Description
Topic	Specifies the topic to which the message must be published.
Partition (optional)	Specifies the partition in which the message must be stored.
Key Schema	Specifies the avro based schema files.
Value Schema	Specifies the avro based schema files.
Maximum Message Size	Specifies the maximum number of messages that can be published.

Note:
If the key serializer class or value serializer class is io.confluent.kafka.serializers.KafkaAvroSerializer in the connection parameters and avro files have been placed under schema package/schemas directory, then avro file names would be available for key schema/value schema.

Signature of Bulk Produce Service Template for Confluent Kafka

The input signature contains the following parameters:

Parameter	Description/Action
`$topic`	Specifies the topic name where the specified value overrides the configured value in the template.

Parameter		Description/Action
`$partition` (optional)		Specifies the partition in which the message is stored. The specified value overrides the configured value in the template.
`messages`		Specifies a set of messages with key, value, and header to be published.
	`key`	Specifies the key to denote the partition in which the messages are added. Messages with the same key are added to a single partition. The input type is a document if it's an avro schema file.
	`value`	Specifies the unit of information that needs to be sent. The input type is a document if it's an avro schema file.
	`header`	Specifies the list of key-value pairs to be added as header in the record.
`globalHeader`		Specifies the list of key-value pairs to be added as header in the record if individual messages do not have headers.

The output signature contains the following parameters:

Parameter		Description/Action
`result`		Specifies the metadata with the following fields of the corresponding message that has been published to Confluent Kafka cluster.
	`isSuccess`	Specifies whether the message is sent successfully or not.
	`topic`	Specifies the topic in which the message is appended.
	`offset`	Specifies the offset of the message in the Kafka topic or Kafka partition.
	`partition`	Specifies the partition in which the messages are sent.
	`errorMessage`	Specifies the error if the message delivery fails.

Adapter for Confluent KSQL

KSQL allows to build event streaming applications from Apache Kafka® topics by using SQL statements and queries. KSQL is built on Kafka Streams, so a KSQL application communicates with a Kafka cluster like any other Kafka Streams application.

Using Adapter for Apache Kafka, Integration Server clients can create and run services to perform KSQL operations such as create streams or tables, select from streams or tables, insert values into streams or tables and so on.

Configuring Select Services

A Select service retrieves specified information from a stream or a table. You configure Adapter for Apache Kafka services using Designer. For more information about adapter services, see "Using Adapter Services" on page 18.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 68 before you configure adapter services.

To configure a Select service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate KSQL connection as **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Select** template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. Select the **From** tab to configure streams or tables using the following fields:

Field	Description/Action
Alias	The alias for table or stream is assigned automatically when you select more than one table in the **Name** field. The default value is `t1`. **Note:** There cannot be more than two tables. It can be a combination of one stream and one table.
Name	Select a table or a stream name.
Type	Displays the type of source you select, that is, table or stream.

7. If you are not joining tables or streams, skip this step. Select the **JOIN** tab to specify the columns for joining the tables you just configured.

   a. Select the  icon to create new left columns.

   b. Select **Left Column** and select the first table's joining column.

c. Select the appropriate **Operator**.

d. Select **Right Column** and select the next table's joining column.

e. When you join two streams, WITHIN must be specified.

Use the following fields:

Field	Description/Action
WITHIN Type	Fixed or Interval. If you select Fixed as **WITHIN Type**, then the 'within' type integer will appear under input parameters of the service. This is equivalent to WITHIN N Timeunit. If you select Interval as **WITHIN Type**, then the 'within' type document will have before and after runtime parameters. This is equivalent to WITHIN(before Timeunit, after Timeunit).
WITHIN Timeunit	The time unit for the window. Valid values like DAY, DAYS, HOUR, HOURS, SECONDS and so on.

8. Use the **SELECT** tab to define the columns and fields to be selected as follows:

a. Use the icon to create new rows as needed. You can use the icon to fill in all rows to the table.

b. As you insert additional rows, the corresponding **Column**, **Column Type**, **Output Field Type**, and **Output Field** display for each column you select in the **Column** field.

Use the following fields:

Field	Description/Action
Column	The column name in the table or stream.
Column Type	The column data type defined in the table or stream.
Output Field Type	The data type of the output field. Adapter for Apache Kafka automatically converts table or stream-specific types to Java data types.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
LIMIT	Specifies the number of records to retrieve from the streams or tables. The default value is 1.

Field	Description/Action
	Note: $limit overrides the design time depending on the given limit value. The service execution continues until the specified number of records are retrieved.
GROUP BY	Selects the column in which **GROUP BY** needs to be applied.
WINDOW	The type of window for the SELECT query such as SESSION, HOPPING, and TUMBLING.
Window Time Unit	The time unit for the window. Valid values like DAY, DAYS, HOUR, HOURS, SECONDS and so on.

9. Use the **WHERE** tab to specify the conditions for selecting information:

   a. Select the icon to define new WHERE clause fields.

   b. Select a logical operator from the **AND/OR** field, an **Operator**, and separators (the left and right parentheses) as needed, and specify values for the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the **Column** and **Input Field**.
Input Field	The default value is `?`, that acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

   The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of the inserted row.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
Input Field Type	The corresponding input field's Java type.

Field	Description
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the **Parameter** and **Column** fields. However, you can also specify a custom value.

c. If necessary, use the ⇧ or ⇩ icons to change the order of the WHERE clause to ensure the parameters are parsed in the correct order.

d. Repeat this procedure until you have specified all WHERE parameters.

10. From the **File** menu, select **Save**.

Configuring Insert Values Services

Insert Values service inserts new information into a stream or table. You configure Adapter for Apache Kafka services using Designer. For more information about adapter services, see "Using Adapter Services" on page 18.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 68 before you configure adapter services.

To configure an Insert Values service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate KSQL connection as **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Insert Values** template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. Select the **Insert Values** tab to configure stream or table and set the fields as follows:

Field	Description/Action
Type	Defines the source type that you select, that is, table or stream.
Name	Name of the source is selected based on the**Type** you select.

7. Use the **Column**, **Column Type**, **Input Field Type** and **Expression** fields on the top row of the tab to define the columns and fields to be inserted as described in the following table.

   a. Use the icon to create new rows as needed. You can use the icon to fill in all rows to the table.

Field	Description/Action
Column	The name of the column for a stream or table.
Column Type	The column data type in a stream or table.
Input Field Type	The corresponding Java type for the input field.
Expression	The default value is `?`, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the stream or the table. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

Note:
MAP, ARRAY, STRUCT data types are not supported in this service.

8. From the **File** menu, select **Save**.

Configuring Custom KSQL Services

A Custom KSQL service defines and executes custom KSQL to perform streaming operations. You can execute almost any KSQL statement required for integrations, such as data management statements. You configure Adapter for Apache Kafka services using Designer. For more information about adapter services, see "Using Adapter Services" on page 18.

If you need to write custom KSQL, you can create a service that uses customized KSQL statements. This allows you the flexibility to execute almost any KSQL statements required, such as data management statements and data definition statements, including terminate, drop, create etc.

Because Custom KSQL service provides no check on KSQL syntax validations, be sure that your KSQL statement works correctly. You can verify KSQL statement accuracy using your vendor's KSQL CLI. For details, see your vendor documentation.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 68 before you configure adapter services.

To create a Custom KSQL service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate KSQL connection as **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Custom KSQL** template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. Select the **KSQL** tab to specify a KSQL statement.

   If you want to specify additional properties at runtime.

   a. Use the icon to create new rows as needed. You can use the icon to fill in all rows to the table.

7. From the **File** menu, select **Save**.

Configuring Insert Into Services

An `Insert Into` service merges specified information from a stream or a table to a selected stream or table. You configure Adapter for Apache Kafka services using Designer. For more information about adapter services, see "Using Adapter Services" on page 18.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 68 before you configure adapter services.

To configure a Select service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.

2. Select the parent namespace, type a name for the adapter service, and click **Next**.

3. Select **Adapter for Apache Kafka** as the adapter type and click **Next**.

4. Select the appropriate KSQL connection as **Adapter Connection Name** and click **Next**.

5. From the list of available templates, select the **Insert Into** template and click **Finish**.

The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

6. Select the **FROM** tab to specify streams or tables from which you want to merge the data.

   Use the following fields:

Field	Description/Action
Alias	The alias for a table or stream is assigned automatically when you select more than one table in the **Name** field. The default value is `t1`. **Note:** There cannot be more than two tables or streams. It can be a combination of one stream and one table.
Name	Select a table or a stream name.
Type	Displays the source type you select, that is, table or stream.

7. Use the **SELECT** tab to define the columns and fields to be selected as follows:

   a. Use the icon to create new rows as needed. You can use the icon to fill in all rows to the table.

   b. As you insert additional rows, the corresponding **Column**, **Column Type**, **Output Field Type**, and **Output Field** display for each column you select in the **Column** field.

      Use the following fields:

Field	Description/Action
Column	The column name in a table or stream.
Column Type	The column data type defined in a stream.
Output Field Type	The data type of the output field. Adapter for Apache Kafka automatically converts KSQL-specific types to Java data types.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
PARTITION BY	The column act as the key column for the output stream, if the PARTITION BY clause is present.

8. Use the **WHERE** tab to specify the conditions for selecting information:

a. Select the icon to define new WHERE clause fields.

b. Select a logical operator from the **AND/OR** field, an **Operator**, and separators (the left and right parentheses) as needed, and specify values for the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the **Column** and **Input Field**.
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of conditions present in WHERE clause.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
Input Field Type	The corresponding input field's Java type.
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the **Parameter** and **Column** fields. However, you can also specify a custom value.

c. If necessary, use the or icons to change the order of the WHERE clause to ensure the parameters are parsed in the correct order.

d. Repeat this procedure until you have specified all WHERE parameters.

9. In **InsertInto** tab, select the **Target Stream** name in which you want to merge the data.

10. From the **File** menu, select **Save**.

Configuring Persistent Query Services

A Persistent Query service creates a KSQL stream or table that contains results of a SELECT query from an existing stream or table. You configure Adapter for Apache Kafka services using Designer. For more information about adapter services, see "Using Adapter Services" on page 18.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 68 before you configure adapter services.

To configure a Select service

1. In Designer, right-click the package in which the service should be contained and select **New > Adapter Service**.
2. Select the parent namespace, type a name for the adapter service, and click **Next**.
3. Select **Adapter for Apache Kafka** as the adapter type and click **Next**.
4. Select the appropriate KSQL connection type as **Adapter Connection Name** and click **Next**.
5. From the list of available templates, select the **Persistent Query** template and click **Finish**.

   The adapter service editor for the adapter service appears. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
6. Select the **Create** tab to configure stream or table using the following fields:

Field	Description/Action
Type	Defines the source type, that is, Stream or Table.
Name	Name of the table or stream to be selected.
Topics	Displays the available topics in the back-end.
Value Format	Data stored in different formats such as JSON, AVRO, DELIMITED.

7. Select the **From** tab for creating a new stream or table out of the selected streams or tables using the following fields:

Field	Description/Action
Alias	The alias for table or stream is assigned automatically when you select more than one table or stream in the **Name** field. The default value is `t1`. **Note:** There cannot be more than two tables. It can be a combination of one stream and one table.
Name	Select a table or a stream name.
Type	Defines the source type you select, that is, table or stream.

8. If you are not joining tables, skip this step. Select the **JOIN** tab to specify the columns for joining the tables you just configured.

   a. Select the icon to create new left columns.

   b. Select **Left Column** and select the first table's joining column.

   c. Select the appropriate **Operator**.

   d. Select **Right Column** and select the next table's joining column.

   e. When you join two streams, WITHIN must be specified.

      Use the following fields:

Field	Description/Action
WITHIN Type	Fixed or Interval. If you select Fixed as **WITHIN Type**, then the 'within' type integer will appear under input parameters of the service. This is equivalent to WITHIN N Timeunit. If you select Interval as **WITHIN Type**, then the 'within' type document will have before and after runtime parameters. This is equivalent to WITHIN(before Timeunit, after Timeunit).
WITHIN Timeunit	The time unit for the window. Valid values like DAY, DAYS, HOUR, HOURS, SECONDS and so on.

9. Use the **SELECT** tab to define the columns and fields to be selected as follows:

   a. Use the icon to create new rows as needed. You can use the icon to fill in all rows to the table.

   b. As you insert additional rows, the corresponding **Column**, **Column Type**, **Output Field Type**, and **Output Field** display for each column you select in the **Column** field.

      Use the following fields:

Field	Description/Action
Column	The column name in table or stream.
Column Type	The column data type defined in table or stream.

Field	Description/Action
Output Field Type	The data type of the output field. Adapter for Apache Kafka automatically converts table or stream-specific types to Java data types.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
PARTITION BY	The column acts as the key column for the stream, if PARTITION BY clause is present.

10. Use the **WHERE** tab to specify the conditions for selecting information:

    a. Select the icon to define new WHERE clause fields.

    b. Select a logical operator from the **AND/OR** field, an **Operator**, and separators (the left and right parentheses) as needed, and specify values for the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the **Column** and **Input Field**.
Input Field	The default value is `?`, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

The adapter automatically generates values for the following fields:

Field	Description
Parameter	The number of conditions present in WHERE clause.
Column (second occurrence of this field)	The name of the column you want to use in the WHERE clause.
Input Field Type	The corresponding input field's Java type.
Input Field (second occurrence of this field)	The name of the input field. By default the name combines the values of the **Parameter** and **Column** fields. However, you can also specify a custom value.

c. If necessary, use the or icons to change the order of the WHERE clause to ensure the parameters are parsed in the correct order.

d. Repeat this procedure until you have specified all WHERE parameters.

11. From the **File** menu, select **Save**.

6 Adapter Listeners and Listener Notifications

Overview of Listener and Listener Notification

This chapter describes how to configure and manage listeners and listener notifications. For a detailed description of Adapter for Apache Kafka listeners, see "Listeners" on page 19. For a detailed description of the available Adapter for Apache Kafka listener notifications, see "Listener Notifications" on page 19.

Listeners

This section provide instructions for configuring and managing Adapter for Apache Kafka listeners.

Before you Configure New Listeners

Perform the following tasks before configuring new listeners:

1. Install webMethods Integration Server and Adapter for Apache Kafka on the same machine. For details, see "Overview of installing and uninstalling Adapter for Apache Kafka" on page 24.

2. Make sure that you have webMethods administrator privileges so that you can access Adapter for Apache Kafka administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Start Integration Server and Integration Server Administrator, if they are not already running.

4. Using Integration Server Administrator, make sure that the WmKafkaAdapter package is enabled. To verify the status of the WmKafkaAdapter package, see "Enabling Packages" on page 29.

5. Start Designer if it is not already running.

6. Using Designer or webMethods Deployer, create a user-defined package to contain the listener, if you have not already done so. When you configure listeners, you must always define them in user-defined packages rather than in the WmKafkaAdapter package. For more information about managing packages, see "Overview of Package Management" on page 28.

   Note:
   If you are using Designer, use the Service Development perspective. For more information, see the *webMethods Service Development Help* for your release.

7. Using Integration Server Administrator, configure a valid connection. For more information about configuring connections, see "Overview of Adapter for Apache Kafka Connections" on page 36.

Configuring a New Listener

When you configure a listener, you specify information that the Integration Server uses to listen to Kafka system messages.

Before you configure the listener, you must configure an adapter connection. You create Adapter for Apache Kafka connections using Integration Server Administrator.

Perform the following procedure to configure a new listener:

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.
2. Click **Listeners**.
3. In the Listeners page, click **Configure New Listener**.
4. In the Listener Types page, select **Kafka® Consumer Listener**.
5. In the Configure Listener Type page, provide values to the following fields:

Parameter	Description/Action
Package	Package in which to create the listener. Use Designer to create the package before specifying the value in this field. For general information about creating packages, see the *webMethods Service Development Help* for your release. **Note:** Configure the listener in a user-defined package rather than in the adapter's package. For other important considerations when creating packages for Adapter for Apache Kafka, see "Package Management" on page 16.
Folder Name	Folder in which to create the listener.
Listener Name	Name of the listener that listens for messages on a Kafka topic.
Connection Name	Name of the connection that the listener must use to connect to the Kafka system. From the list of connections, select an appropriate consumer connection.

Parameter	Description/Action
Retry Limit	Number of times the adapter tries to reconnect if the adapter fails to connect, or loses connection with the Queue Manager. Default: **5**
Retry Backoff Timeout	Time in seconds that must elapse between each of the retries specified in the **Retry Limit**. Default: **10**

Important:
The listener name is prefixed by the folder name and is separated by a colon. For example: If the folder name is "Folder1" and the listener name is "Listener1", then the listener name in the "Listeners" screen will be "Folder1:Listener1".

6. In the Listener Properties section, use the following fields:

Note:
An asterisk (*) next to the filter fields indicates that Adapter for Apache Kafka passes the messages that match the filter criteria to the notification. Messages that do not match the specified filter criteria remain on the queue.

Parameter	Description/Action
Topic Name(s)	Specify comma separated list of topic.
Partition(s)	Comma separated list of partition. **Note:** If you have multiple partitions to listen from, then you provide comma separated list. Otherwise, provide just the single value.
Seek from Offset(s)	The offset of the partition from where the consumer needs to start consuming the messages.
pollInterval	Type the polling interval time in seconds.

7. Click **Save Listener**. If the parameters are valid, the listener you created appears on the page.

8. To monitor the Kafka topic, enable the listener. For more information on enabling listeners, see "Enabling Listeners" on page 95.

Enabling Listeners

The listener must be enabled to monitor the Kafka topic. When you create a listener, it is not automatically enabled.

Before you enable a listener, you must configure one or more listener notifications to associate with the listener. If no notifications are configured when you enable the listener, then the Integration Server Administrator displays a warning message.

After you have configured your listener notifications, you must enable the listener so that the associated notifications will communicate appropriately with the listener at run time. You enable the listeners using the Integration Server. For more information on configuring listeners and notifications, see the sections on "Configuring a New Listener" on page 93 and "Configuring Listener Notifications" on page 102.

Note:

- You cannot enable a listener if the connection used by the listener is disabled.
- When you reload a package that contains enabled listeners, the listeners will automatically be enabled when the package reloads. If the package contains listeners that are disabled, they will remain disabled when the package reloads.

Perform the following steps to enable a listener:

1. In the **Adapters** menu in the navigation area of the Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.
2. In the **webMethods Adapter for Apache Kafka** menu, click **Listeners**.
3. In the Listeners page, select **Enabled** in the **State** column for the listener you want to enable.

   Integration Server Administrator enables the listener.
4. Click **Enable all suspended** to quickly enable multiple suspended listeners.

Suspending Listeners

You must be running Integration Server to suspend a listener.

Note:
A suspended listener acts in the same way as a disabled listener.

Perform the following steps to suspend a listener:

1. In the **Adapters** menu in the navigation area of the Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.

2. In the **webMethods Adapter for Apache Kafka** menu, click **Listeners**.

3. In the Listeners page, select **Suspended** in the **State** column for the listener you want to suspend.

   Integration Server suspends the listener.

   The

4. Click **Suspend all enabled** to change the state quickly for multiple enabled listeners.

When you suspend a listener, the action may not take effect right away. You may have to wait. If one or more messages appear on the queue within that time interval, the adapter may receive and process the first message.

Disabling Listeners

Listeners must be disabled before you can edit or delete them. You disable listeners using Integration Server Administrator.

Note:
When you disable a connection alias, Integration Server also disables any listener associated with the connection.

Perform the following steps to disable a listener:

1. In the **Adapters** menu in the navigation area of the Adapter for Apache Kafka, click **webMethods Adapter for Apache Kafka**.

2. Click **Listeners**.

3. In the Listeners screen, select **Disabled** in the **State** column for the listener you want to disable.

   Integration Server Administrator disables the listener.

When you disable a listener, the action may not take effect right away. You may have to wait as long as the time specified in the **Wait Interval** parameter for the listener. If one or more messages appear on the queue within that time interval, the adapter may receive and process the first message.

Editing Listeners

You use Integration Server Administrator to edit the listener in the following situations:

- If you need to select a newly configured connection, or if you need to change any listener properties, such as filter values, you can update the listener parameters.

- If you need to change the order of the notifications that are associated with the listener. For more information, see "Editing the Notification Order of a Listener" on page 97.

Editing Listener Parameters

You can redefine the parameters that a listener uses when connecting to Kafka system using the Integration Server Administrator.

Note:
You can edit a listener only if it is disabled. For instructions on disabling a listener, see "Disabling Listeners" on page 96.

Perform the following steps to edit a listener:

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.
2. Click **Listeners**.
3. In the Listeners page, click the icon for the listener that you want to edit.

   The Edit Listener page displays the current parameters for the listener. For descriptions of the listener parameters, see "Configuring a New Listener" on page 93.
4. Update the listener's parameters by typing or selecting the values you want to specify.
5. Click **Save Changes** to save the listener and return to the Listeners screen.

Editing the Notification Order of a Listener

Important:
You are allowed to change the notification order of the child listeners only if a child listener is directly associated with notifications (not inheriting the notifications from the parent). The link to change the notification order is inactive for the child listeners that do not have any notifications configured.

To edit the notification order of a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.
2. In the **webMethods Adapter for Apache Kafka** menu, click **Listeners**.
3. On the Listeners screen, make sure that the listener is disabled before editing. To disable the listener, see "Disabling Listeners" on page 96.

4. On the Listeners screen, click the icon for the listener that you want to edit.

5. On the Edit Listener screen, click **Edit Notification Order**.

6. On the Edit Notification Order screen, use the **Up** and **Down** buttons to determine the processing order in which Adapter for Apache Kafka invokes the notifications.

   Note:
   For better processing results, arrange your notifications from ascending to descending order starting with the most detailed notifications to the least detailed notifications. For more information on notifications and their filter criteria, see "Considerations for Listener Notifications" on page 102.

7. Click **Save Changes** to save the notification order of the listener.

8. Click **Return to webmethods Adapter for Apache Kafka Listeners** to return to the Edit Listener screen.

Viewing Listener Parameters

You can view a listener's parameters from Integration Server Administrator, or Designer. You can also view the notification order of a listener.

Viewing Listener Parameters Using Integration Server Administrator

Perform the following procedure to view listener parameters using Integration Server Administrator.

To view a listener's parameters using Integration Server Administrator

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.

2. In the **webMethods Adapter for Apache Kafka** menu, click **Listeners**.

   The Listener page appears, listing all the listeners. You can perform the following:

   - Control the number of listener notifications that are displayed. For more information, see "Controlling Pagination" on page 22.
   - Sort and filter the listeners. For more information, see "Sorting and Filtering Listeners" on page 99.
   - Enable and disable the listeners. For more information, see "Enabling Listeners" on page 95 and "Disabling Listeners" on page 96.
   - View listener notification order. For more information, see "Viewing the Notification Order of a Listener" on page 99.

3. In the Listeners page, click the icon for the listener that you want to see.

   The View Listener page displays the parameters for the listener. For descriptions of the listener parameters, see "Configuring a New Listener" on page 93.

4. Click **Return to webmethods Adapter for Apache Kafka Listeners** to return to the main listeners screen.

Sorting and Filtering Listeners

Perform the following procedure to view listener parameters using Integration Server Administrator.

To view a listener's parameters using Integration Server Administrator

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka** > **Listeners**.

2. Perform the following to sort the listeners:

   a. Click the **Up** and **Down** arrows at the top of the column you want to sort.

3. Perform the following to filter the listeners:

   a. Click **Filter Listeners**.

   b. Type the criterion by which you want to filter into the **Filter criteria** box. Filtering is based on the node name, not the listener name. To locate all listeners containing specific alphanumeric characters, use asterisks (*) as wildcards. For example, if you want to display all listeners containing the string "abc", type `*abc*` in the **Filter criteria** box.

   c. Click **Submit**. The Listeners page displays the connections that match the filter criteria.

   d. Click **Show All Listeners** to re-display all listeners.

Viewing the Notification Order of a Listener

Perform the following procedure to view the notification order of a listener.

To view the notification order of a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.

2. Click **Listeners**.

3. In the Listeners page, click the icon for the listener that you want to view.

4. Click **View Notification Order**.

   The View Notification Order page displays the order of the listener notifications for the listener. To change the notification order for the listener, see "Editing the Notification Order of a Listener" on page 97.

5. Click **Return to View Listener** to return to the View Listener screen.

6. Click **Return to webmethods Adapter for Apache Kafka Listeners** to return to the Listeners page.

Viewing Listener Parameters Using Designer

Perform the following procedure to view listener parameters using Designer:

1. From Designer, expand the package and folder in which the listener is located.

2. Double-click the listener you want to view.

   The parameters for the listener appear on the **Listener Information** tab. For descriptions of the listener properties, see "Configuring a New Listener" on page 93.

Copying Listeners

You can copy an existing listener to create a new listener with the same or similar properties without having to re-type all properties for the listener. You copy adapter listeners using Integration Server Administrator.

Perform the following steps to copy a listener:

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.

2. Click **Listeners**.

3. In the Listeners page, click the icon for the listener that you want to copy.

   The Copy Listener page displays the current parameters for the listener that you want to copy. Update the listener name and parameters by typing or selecting the values you want to change.

   For descriptions of the listener parameters, see "Configuring a New Listener" on page 93.

4. Click **Save Listener Copy** to save the listener and return to the Listeners screen.

Deleting Listeners

If you no longer want to use a listener, use the following instructions to delete the listener. You delete listeners using Integration Server Administrator.

Important:

You can delete a listener only if the listener is disabled. For instructions on disabling a listener, see "Disabling Listeners" on page 96.

Perform the following steps to delete a listener:

1. In the **Adapters** menu in the navigation area of the Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.
2. Click **Listeners**.
3. In the Listeners page, click ✕ for the listener you want to delete.

   Integration Server deletes the listener.

Listener Notifications

The following sections provide instructions for configuring and managing Adapter for Apache Kafka listener notifications. When a listener receives a message from a Kafka topic, then it passes the message to an enabled listener notification that is associated with the listener.

Important:
The message is lost if you do not configure any notifications with a listener, or if you do not enable any notifications which are already configured. Software AG recommends that you enable a listener notification before you enable the listener.

Ideally, you must configure only one notification per listener. If you configure multiple listener notifications with the same listener, the listener passes the data to the listener notifications created first. You can view and edit the order in which the listener notifications receive the data from the listener. For more information, see "Editing the Notification Order of a Listener" on page 97 and "Viewing the Notification Order of a Listener" on page 99. If you have configured multiple notifications with a listener, Software AG recommends that you enable only the notification that needs to be executed.

Adapter for Apache Kafka provides following types of listener notifications that you can configure:

- Synchronous Apache Kafka Listener Notifications
- Asynchronous Apache Kafka Listener Notifications
- Synchronous Confluent Kafka Listener Notifications
- Asynchronous Confluent Kafka Listener Notifications

For more information on how listener notifications work, see "Listener Notifications" on page 19.

Before you Configure Listener Notifications

Perform the following tasks before configuring listener notifications:

1. Install webMethods Integration Server and Adapter for Apache Kafka on the same machine. For details, see "Overview of installing and uninstalling Adapter for Apache Kafka" on page 24.

2. Make sure that you have webMethods administrator privileges so that you can access the Adapter for Apache Kafka's administrative screens. For information about setting user privileges, see the Integration Server for your release.

3. Start Integration Server and Integration Server Administrator, if they are not already running.

4. Using Integration Server Administrator, make sure that the WmKafkaAdapter package is enabled. To verify the status of the WmKafkaAdapter package, see "Enabling Packages" on page 29.

5. Start Designer if it is not already running.

6. Using Designer, create a user-defined package to contain the listener, if you have not already done so. For more information about managing packages, see "Overview of Package Management" on page 28.

7. Using Integration Server Administrator, configure an adapter connection to use with the listener. For more information, "Overview of Adapter for Apache Kafka Connections" on page 36.

8. Configure a listener using Integration Server Administrator. For more information on how to configure a new listener, see "Configuring a New Listener" on page 93.

Considerations for Listener Notifications

For more efficient message processing to Integration Server, make sure that you validate the listener notification filter criteria when configuring listener notifications so that it does not conflict with the listener filter criteria. For example, if both a listener and its notification filter on the same field but specify different values, the notification filter criteria can never match. In this case, the listener will remove the message off the topic because it passed the listener filter criteria. However, the notification will not be processed because the value of the field will not have a match value for the notification filter criteria.

Configuring Listener Notifications

Adapter for Apache Kafka has four types of listener notifications that you can configure:

- Asynchronous Apache Kafka Listener Notification

- Synchronous Apache Kafka Listener Notification
- Asynchronous Confluent Kafka Listener Notification
- Synchronous Confluent Kafka Listener Notification

For a description of the listener notifications, see "Listener Notifications" on page 19. You configure listener notifications using Designer.

Configuring a Synchronous Apache Kafka Listener Notification

Perform the following procedure to configure a synchronous Apache Kafka listener notification:

1. Review the steps in "Before you Configure Listener Notifications" on page 102.
2. Start Designer.
3. Perform the following:
   a. Right-click the package in which the notification should be contained and select **New > Adapter Notification**.
   b. Select the parent namespace and type a name for the adapter notification.
   c. Click **Next**.
4. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
5. Select **Apache Kafka Listener Notification (synchronous)** as the template and click **Next**.
6. Select the appropriate **Notification Listener Name** and click **Next**.
7. Select the appropriate service node and click **Next**.
8. Review the **Request Document Name** and **Reply Document Name** and click **Finish**.

   The Adapter Notification template creates the following items:

   - A listener notification
   - A ReplyDocument
   - A RequestDocument
9. In the Apache Sync Listener Notification tab, confirm the **outputParameterNames**, **outputFieldNames**, and **outputFieldTypes**. The structure will reflect in the RequestDocument.
10. In the Adapter Settings tab, in Adapter Properties section, confirm the adapter name, adapter listener name, and adapter notification template.

11. Select the **Adapter Settings** tab > **Execution Mode** section, perform one of the following procedures:

    - To invoke a flow service directly, select **Service Invoke** and specify the **Service Name** field as the name of the service that you selected when you initially configured the adapter listener notification. You can select a different service by clicking the browse icon  to the right of the **Service Name** field.

      Important:
      Before you select a different service, make sure that you have disabled the notification. When the notification is enabled, then the new service is utilized. For more information on enabling and disabling listeners, see "Enabling Listeners" on page 95 and "Disabling Listeners" on page 96.

    - To publish documents locally or to a Universal Messaging and wait for a reply, select **Publish and Wait** and specify the following fields:

      - **Local** - Select **true** to publish documents locally to Integration Server only. Select **false** to publish to the Universal Messaging connected to Integration Server. If no Universal Messaging is configured for Integration Server, documents will be published locally.
      - **Wait Time** - Specify the number of milliseconds to wait for a reply. Default value is `-1`, which means to wait indefinitely.

12. From the **File** menu, select **Save**.

Configuring an Apache Kafka Asynchronous Listener Notification

Perform the following procedure to configure an asynchronous listener notification:

1. Review the steps in "Before you Configure Listener Notifications" on page 102.
2. Start Designer.
3. Perform the following:
   a. Right-click the package in which the notification should be contained and select **New > Adapter Notification**.
   b. Select the parent namespace and type a name for the adapter notification.
   c. Click **Next**.
4. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
5. Select **Apache Kafka Listener Notification (asynchronous)** as the template and click **Next**.
6. Select the appropriate **Notification Listener Name** and click **Next**.

7. Review the **Publish Document Name** and click **Finish**.

   The following items items are created:

   - An asynchronous listener notification
   - A PublishDocument

8. In the Apache Async Listener Notification tab, confirm the **outputParameterNames**, **outputFieldNames**, and **outputFieldTypes**. The structure will reflect in the PublishDocument.

9. In the Adapter Settings tab, Adapter Properties section confirm the adapter name, adapter listener name, and adapter notification template.

10. In the Publish Document to area, perform one of the following procedures:

    - To publish document to the Universal Messaging connected to the local Integration Server:

      1. Select **webMethods Messaging Provider**.
      2. Select **IS_UM_CONNECTION**.

         This option publishes documents to Universal Messaging connected to local Integration Server, if one is configured.

      3. Create a matching **webMethods Messaging Trigger** to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Universal Messaging. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.

      For more information about configuring the trigger, see the Integration Server *Publish-Subscribe Developer's Guide* for your release and *webMethods Service Development Help*.

    - To publish document to local Integration Server:

      1. Select **webMethods Messaging Provider**.
      2. Select **IS_LOCAL_CONNECTION**.
      3. Create a matching local **webMethods Messaging Trigger** to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Universal Messaging. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.

      For more information about configuring the trigger, see the Integration Server *Publish-Subscribe Developer's Guide* for your release and *webMethods Service Development Help*.

    - To publish the document to internal/external JMS provider in the form of messages:

      1. Select **JMS Provider**.

2. Click the **Browse** button next to **Connection alias name** to Select the name of the connection alias as it is configured in Integration Server and then click **OK**.
3. Type the destination name defined in the JMS provider to specify the target of messages the client produces and the source of messages it consumes.
4. Specify whether the destination is a **Queue** or a **Topic**. The default is `Queue`.

- For more information about configuring JMS Provider, see *webMethods Integration Server Administrator's Guide* for your release.

Configuring a Synchronous Confluent Kafka Listener Notification

Perform the following procedure to configure a synchronous Confluent Kafka listener notification:

1. Review the steps in "Before you Configure Listener Notifications" on page 102.
2. Start Designer.
3. Perform the following:
   a. Right-click the package in which the notification should be contained and select **New > Adapter Notification**.
   b. Select the parent namespace and type a name for the adapter notification.
   c. Click **Next**.
4. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
5. Select **Confluent Kafka Listener Notification (synchronous)** as the template and click **Next**.
6. Select the appropriate **Notification Listener Name** and click **Next**.
7. Select the appropriate service node and click **Next**.
8. Review the **Request Document Name** and **Reply Document Name** and click **Finish**.

   The following items are created:

   - A listener notification
   - A ReplyDocument
   - A RequestDocument
9. In the Confluent Sync Listener Notification tab, provide values to the following fields:

Parameter	Description/Action
Key Schema	Schema file defining the structure for keys in messages received from Confluent Kafka. The **Key Schema** is populated if the Key Deserializer Class is set to `io.confluent.kafka.serializers.KafkaAvroDeserializer` in the connection parameters.
Value Schema	Schema file defining the structure for values in messages received from Confluent Kafka. The **Value Schema** is populated if the Value Deserializer Class is set to `io.confluent.kafka.serializers.KafkaAvroDeserializer` in the connection parameters.

The non-editable fields **outputParameterNames** and **outputFieldTypes** are populated based on the files selected in **Key Schema** and **Value Schema** fields. The structure will reflect in the RequestDocument.

10. In the Adapter Settings tab, confirm the adapter name, adapter listener name, and adapter notification template.

11. Select the **Adapter Settings** tab > **Execution Mode** section, perform one of the following procedures:

    - To invoke a flow service directly, select **Service Invoke** and specify the **Service Name** field as the name of the service that you selected when you initially configured the adapter listener notification. You can select a different service by clicking the browse icon  to the right of the **Service Name** field.

      Important:
      Before you select a different service, make sure that you have disabled the notification. When the notification is enabled, then the new service is utilized. For more information on enabling and disabling listeners, see "Enabling Listeners" on page 95 and "Disabling Listeners" on page 96.

    - To publish documents locally or to a Universal Messaging and wait for a reply, select **Publish and Wait** and specify the following fields:

      - **Local** - Select **true** to publish documents locally to Integration Server only. Select **false** to publish to the Universal Messaging connected to Integration Server. If no Universal Messaging is configured for Integration Server, documents will be published locally
      - **Wait Time** - Specify the number of milliseconds to wait for a reply. Default value is `-1`, which means to wait indefinitely.

12. From the **File** menu, select **Save**.

Configuring an Confluent Kafka Asynchronous Listener Notification

Perform the following procedure to configure an asynchronous listener notification:

1. Review the steps in "Before you Configure Listener Notifications" on page 102.
2. Start Designer.
3. Perform the following:
   a. Right-click the package in which the notification should be contained and select **New > Adapter Notification**.
   b. Select the parent namespace and type a name for the adapter notification.
   c. Click **Next**.
4. Select **webMethods Adapter for Apache Kafka** as the adapter type and click **Next**.
5. Select **Confluent Kafka Listener Notification (asynchronous)** as the template and click **Next**.
6. Select the appropriate **Notification Listener Name** and click **Next**.
7. Review the **Publish Document Name** and click **Finish**.

   The following items items are created:

   - An asynchronous listener notification
   - A PublishDocument
8. In the Confluent Async Listener Notification tab, provide values to the following fields:

Parameter	Description/Action
Key Schema	Schema file defining the structure for keys in messages received from Confluent Kafka. The **Key Schema** is populated if the Key Deserializer Class is set to `io.confluent.kafka.serializers.KafkaAvroDeserializer` in the connection parameters.
Value Schema	Schema file defining the structure for values in messages received from Confluent Kafka. The **Value Schema** is populated if the Value Deserializer Class is set to `io.confluent.kafka.serializers.KafkaAvroDeserializer` in the connection parameters.

The non-editable fields **outputParameterNames** and **outputFieldTypes** are populated based on the files selected in **Key Schema** and **Value Schema** fields. The structure will reflect in the PublishDocument.

9. In the Adapter Settings tab, Adapter Properties section confirm the adapter name, adapter listener name, and adapter notification template.

10. In the Publish Document to area, perform one of the following procedures:

- To publish document to the Universal Messaging connected to the local Integration Server:

  1. Select **webMethods Messaging Provider**.
  2. Select **IS_UM_CONNECTION**.

     This option publishes documents to Universal Messaging connected to local Integration Server, if one is configured.
  3. Create a matching **webMethods Messaging Trigger** to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Universal Messaging. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.

  For more information about configuring the trigger, see the Integration Server *Publish-Subscribe Developer's Guide* for your release and *webMethods Service Development Help*.

- To publish document to local Integration Server:

  1. Select **webMethods Messaging Provider**.
  2. Select **IS_LOCAL_CONNECTION**.
  3. Create a matching local **webMethods Messaging Trigger** to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Universal Messaging. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.

  For more information about configuring the trigger, see the Integration Server *Publish-Subscribe Developer's Guide* for your release and *webMethods Service Development Help*.

- To publish the document to internal/external JMS provider in the form of messages:

  1. Select **JMS Provider**.
  2. Click the **Browse** button next to **Connection alias name** to Select the name of the connection alias as it is configured in Integration Server and then click **OK**.
  3. Type the destination name defined in the JMS provider to specify the target of messages the client produces and the source of messages it consumes.
  4. Specify whether the destination is a **Queue** or a **Topic**. The default is `Queue`.

- For more information about configuring JMS Provider, see *webMethods Integration Server Administrator's Guide* for your release.

Viewing Listener Notifications using Designer

Perform the following steps to view the listener notification using Designer:

1. In Designer, expand the package and folder that contains the listener notification you want to view.

2. Select the listener notification you want to view.

   Designer displays the configured listener notification in the adapter's Adapter Notification Editor.

Editing Listener Notifications

You use Designer to edit listener notifications, both synchronous and asynchronous.

Note:
Because listener notifications inherently depend on listeners, you cannot change a listener for a listener notification after you configure it.

To edit a listener notification

1. In Designer, expand the package and folder that contain the listener notification you want to edit.

2. Select the listener notification you want to edit.

   Designer displays the configured listener notification in the adapter's Adapter Notification Editor.

3. Do one of the following:

   - If you have the VCS Integration feature enabled, right-click the notification and select **Check Out**.
   - If you do not have the VCS Integration feature enabled, right-click the notification and select **Lock for Edit**.
   - If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check out the notification. The options available in the **Team** menu depend on the VCS client that you use.

4. Open the listener notification and modify the notification's parameter values as needed. For detailed descriptions of the notification parameters, see "Configuring Listener Notifications" on page 102.

5. After you complete the modifications, save the notification and do one of the following:

   - If you have the VCS Integration feature enabled, right-click the notification and select **Check In**. Enter a check-in comment and click **OK**.
   - If you do not have the VCS Integration feature enabled, right-click the notification and select **Unlock**.
   - If you are using the local service development feature, from the **Team** menu in Designer, select the appropriate option to check in the notification. The options available in the **Team** menu depend on the VCS client that you use.

Deleting Listener Notifications

If you no longer want to use a particular Adapter for Apache Kafka listener notification, you can delete it by following the instructions in this section. You delete listener notifications using Designer.

Important:
If you delete a synchronous listener notification, then the associated documents RequestDocument and ReplyDocument are automatically deleted. If you delete an asynchronous listener notification, then the associated PublishDocument is automatically deleted. You cannot individually delete the document types associated with the synchronous and asynchronous listener notifications.

To delete a listener notification

1. In Designer, expand the package and folder that contain the listener notification you want to delete.

2. Right-click the listener notification and click **Delete**.

Enabling Listener Notifications

After you configure an listener notification, you need to enable it using Integration Server Administrator.

Perform the following steps to enable a listener notification:

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka** .

2. Click **Listener Notifications**.

3. In the Listener Notifications page, click **No** in the **Enabled** column for the listener notification you want to enable.

   Integration Server Administrator enables the listener notification and displays a ✔ and **Yes**in the **Enabled** column.

Disabling Listener Notifications

You disable listener notifications using Integration Server Administrator.

To disable a listener notification

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **webMethods Adapter for Apache Kafka**.

2. Click **Listener Notifications**.

3. In the Listener Notifications page, click **Yes** in the **Enabled** column for the listener notification you want to disable.

   The listener notification becomes disabled and you see a **No** in the **Enabled** column.

7 Predefined Health Indicator

Predefined Health Indicator

Microservices Runtime includes predefined health indicators for some of its basic components. The health indicator captures the connection details for all the WmART based adapters at runtime. For more information, see *webMethods Adapter Runtime User's Guide*.

8 Administrator APIs

Administrator APIs

The Administrator APIs are available for Adapter for Apache Kafka. For more information about Administrator APIs and samples, see *webMethods Adapter Runtime User's Guide*.

9 Configuration Variables Templates for Adapter Assets in Microservices Runtime

Configuration Variables Templates for Adapter Assets in Microservices Runtime

The webMethods Adapter Runtime (ART) asset properties that can be configured from Integration Server Administrator are available in the configuration variables template (`application.properties` file) generated by Microservices Runtime. For more information, see *webMethods Adapter Runtime User's Guide* and *Developing Microservices with webMethods Microservices Runtime*.

10 Built-In Services

Overview

This appendix provides information on the built-in services provided by Adapter for Apache Kafka. These services are located in the WmKafkaAdapter package.

wm.adapter.wmkafka.serde:registerSerializer

The wm.adapter.wmkafka.serde:registerSerializer allows the user to configure their own serializer for **Key Serializer Class** and **Value Serializer Class**. The new serializer appears in the key or value serializer connection parameters. This is applicable only in Confluent Connection types.

Input Parameters

className	The name of the corresponding serializer class.

Output Parameters

result	The result of the service. Displays a message *successfully added* or *Failed*.

wm.adapter.wmkafka.serde:registerDeserializer

The wm.adapter.wmkafka.serde:registerDeserializer allows the user to configure their own deserializer for **Key Deserializer Class** and **Value Deserializer Class**. The new deserializer appears in the key or value deserializer connection parameters. This is applicable only in Confluent Connection types.

Input Parameters

className	The name of the corresponding deserializer class.

Output Parameters

result	The result of the service. Displays a message *successfully added* or *Failed*.

wm.adapter.wmkafka.ksql.admin:create

The wm.adapter.wmkafka.ksql.admin:create allows the user to create a stream or a table with the specified columns from an underlying Kafka topic.

Input Parameters

`$connectionName(string)`	Name of the KSQL connection.
`type(string)`	Type of source: STREAM or TABLE.
`name(string)`	Name of the stream or table to be created.
`topic(string)`	Name of the Kafka topic that supports the source. The topic must exist in Kafka.
`messageFormat(string)`	Specifies the serialization format of the message value in the topic. Supported formats are JSON, DELIMITED (comma-separated value), and AVRO.
`windowType(string)`	By default, the topic is assumed to contain non-windowed data. If the data is windowed, that is, if the data is created using KSQL query, that contains a *`WINDOW`* clause. use the`windowType` property to provide the type of window. Valid values are SESSION, HOPPING, and TUMBLING.
`columns(document)`	List of columns(name and type) present in stream or table.
`keyColumn(string)`	Determines if repartitioning can be avoided when performing aggregations and joins. **Note:** Do not set the KEY property, unless you have validated that your stream does not need to be repartitioned for the future joins. If you set the `KEY` property and your record key does not meet partitioning requirements, you will need to repartition explicitly.
`timestampColumn(string)`	Overrides the ROWTIME with the contents of the specified field or column within the Kafka message value (similar to timestamp extractors in Kafka's Streams API). By default, the implicit ROWTIME column is the timestamp of the message in Kafka topic. Timestamps have a millisecond accuracy. Time-based operations, such as windowing, will process a record according to the timestamp in ROWTIME.
`timestampFormat(string)`	Used in conjunction with `timestampColumn`.

	If this parameter is set, the TIMESTAMP field must be of type `varchar` and have a format that can be parsed with the Java DateTimeFormatter. If your timestamp format has characters requiring single quotes, you can escape them with successive single quotes, '', for example: 'yyyy-MM-dd''T''HH:mm:ssX'. If the value is not set, `timestampFormat` will assume that the TIMESTAMP field is a BIGINT.

Output Parameter

The resulting output documents consists of the following fields:

`commandId(string)`	Identifies the requested operation. You can use this ID to poll the result of the operation using the status endpoint.
`commandStatus.status(string)`	QUEUED, PARSING, EXECUTING, TERMINATED, SUCCESS, or ERROR.
`commandStatus.message(string)`	Provides a detailed message regarding the status of the execution statement.
`commandSequenceNumber(string)`	Indicates the sequence number of the requested operation in the command queue, or -1 if the operation was unsuccessful.

wm.adapter.wmkafka.ksql.admin:drop

The wm.adapter.wmkafka.ksql.admin:drop allows the user to delete a stream or a table.

Input Parameters

`$connectionName`	Name of the KSQL connection.
`type`	Type of source STREAM or TABLE.
`name`	Name of the stream or table to be deleted.
`ifExists`	If the value is set as TRUE then service does not fail even if the table or stream doesn't exist.
`deleteTopic`	If TRUE, underlying kafka topic will be marked for deletion.

Output Parameters

The resulting output document consist of following fields:

`commandId(string)`	Identifies the requested operation. You can use this ID to poll the result of the operation using the status endpoint.
`commandStatus.status(string)`	QUEUED, PARSING, EXECUTING, TERMINATED, SUCCESS, or ERROR.
`commandStatus.message(string)`	Provides a detailed message regarding the status of the execution statement.
`commandSequenceNumber(long)`	Indicates the sequence number of the requested operation in the command queue, or -1 if the operation was unsuccessful.

wm.adapter.wmkafka.ksql.query:terminate

The wm.adapter.wmkafka.ksql.query:terminate allows the user to terminate the persistent queries(read and write) running on a stream or table.

Input Parameters

`$connectionName(String)`	Name of the KSQL connection.
`name(String)`	Name of the stream or table.

Output Parameters

The resulting output document consist of following fields:

`commandId(string)`	Identifies the requested operation. You can use this ID to poll the result of the operation using the status endpoint.
`commandStatus.status(string)`	QUEUED, PARSING, EXECUTING, TERMINATED, SUCCESS, or ERROR.
`commandStatus.message(string)`	Provides a detailed message regarding the status of the execution statement.
`commandSequenceNumber(long)`	Indicates the sequence number of the requested operation in the command queue, or -1 if the operation was unsuccessful.

wm.adapter.wmkafka.topic:create

The wm.adapter.wmkafka.topic:create allows the user to create topics with specified number of partitions and replication factor.

Input Parameters

$connectionName(String)	Name of the producer or consumer connection.
topics.name(String)	Name of the topic to be created.
topics.partitions(Integer)	Number of partitions.
topics.replicationFactor(Short)	Number of replications. The topic maintains the replications in the Kafka Cluster.

Output Parameters

topicName	Name of the topic.
isCreated	Indicates if the topic is created.

wm.adapter.wmkafka.topic:drop

The wm.adapter.wmkafka.topic:drop allows the user to delete the topics present in the Kafka cluster.

Input Parameters

$connectionName(string)	Name of the producer or consumer connection.
topics (string list)	Name(s) of the Kafka topic(s) to be dropped.

Output Parameter

topicName	Name of the Kafka topic.
isDeleted	Indicates if the Kafka topic is deleted.

wm.adapter.wmkafka.topic:list

The wm.adapter.wmkafka.topic:list allows the user to get all the Kafka topics present in the Kafka cluster.

Input Parameters

$connectionName (String)	Name of the connection.

Output parameters

name	Name of the Kafka topic.
partitions	Number of partitions for Kafka topic.

wm.adapter.wmkafka.topic:describe

The wm.adapter.wmkafka.topic:describe allows the user to describe the topics.

Input Parameters

$connectionName(string)	Name of the producer or consumer connection.
topics (string list)	Name(s) of the Kafka topic(s) to be dropped.

Output Parameters

topicName	Name of the topic.
isInternal	Indicates the topic is internal to Kafka cluster or not.
partitions	Number of partitions that Kafka topic is comprised of.

wm.adapter.wmkafka.cluster:describe

The wm.adapter.wmkafka.cluster:describe allows the user to view the Kafka cluster information such as cluster ID, controller, nodes and so on.

Input Parameters

$connectionName(string)	Name of the producer or consumer connection.

Output Parameters

clusterId	Current identity of the cluster.

controller	Document describing controller node information such as ID, IP, port and so on.
nodes	List of nodes in the Kafka cluster.

A Configuration Parameters

watt.adapter.kafka.include.internal.topics

Enables filtering internal topics displayed in the dropdown list for topic names in adapter services. Possible values are:

- `false`. Default. Internal topic names are not listed in the topic names in adapter services.
- `true`. All topic names including internal topic names are listed in the topic names in adapter services.

Note:
This parameter is applicable only if the version of producer and consumer connections are v11 and above.