Kafka REST API SwaggerUI

Table of Contents

Chapter 1. Overview

Kafka REST API SwaggerUI

1.1. Version information

Version : 0.1.0

1.2. Contact information

Contact : gnuhpc
Contact Email : gnuhpc@gmail.com

1.3. URI scheme

Host : localhost:8080
BasePath : /

1.4. Tags

- kafka-controller : Kafka Controller
- zookeeper-controller : Zookeeper Controller

Chapter 2. Chapter of manual content 1

This is some dummy text

2.1. Sub chapter

Dummy text of sub chapter

Chapter 3. Chapter of manual content 2

This is some dummy text

Chapter 4. Resources

4.1. Kafka-controller

Kafka Controller

4.1.1. List brokers in this cluster

```
GET /kafka/brokers
```

Responses

HTTP Code	Description	Schema
200	OK	< BrokerInfo > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.2. Get the message from the offset of the partition in the topic, decoder is not supported yet

```
GET /kafka/consumer/{topic}/{partition}/{offset}
```

Parameters

Type	Name	Description	Schema
Path	**offset** *required*	offset	integer(int64)

Type	Name	Description	Schema
Path	**partition** *required*	partition	integer(int32)
Path	**topic** *required*	topic	string
Query	**decoder** *optional*	decoder	string

Responses

HTTP Code	Description	Schema
200	OK	string
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.3. Delete old Consumer Group

```
DELETE /kafka/consumergroup/{consumergroup}
```

Parameters

Type	Name	Description	Schema
Path	**consumergroup** *required*	consumergroup	string

Responses

HTTP Code	Description	Schema
200	OK	GeneralResponse
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.4. getLastCommitTimestamp

```
GET /kafka/consumergroup/{consumergroup}/{topic}/lastcommittime
```

Parameters

Type	Name	Description	Schema
Path	**consumergroup** *required*	consumergroup	string
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	< string, < string, integer(int64) > map > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.5. Reset consumer group offset, earliest/latest can be used

```
PUT /kafka/consumergroup/{consumergroup}/{type}/{topic}/{partition}/{offset}
```

Parameters

Type	Name	Description	Schema
Path	**consumergroup** *required*	consumergroup	string
Path	**offset** *required*	offset	string
Path	**partition** *required*	partition	integer(int32)
Path	**topic** *required*	topic	string
Path	**type** *required*	type	string

Responses

HTTP Code	Description	Schema
200	OK	GeneralResponse
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.6. List consumer groups from zk and kafka

```
GET /kafka/consumergroups
```

Parameters

Type	Name	Description	Schema
Query	**topic** *optional*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	< string, < string > array > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.7. Describe consumer groups, showing lag and offset

```
GET /kafka/consumergroups/{consumerGroup}/{type}
```

Parameters

Type	Name	Description	Schema
Path	**consumerGroup** *required*	consumerGroup	string

Type	Name	Description	Schema
Path	**type** *required*	type	string
Query	**topic** *optional*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	< string, < ConsumerGroupDesc > array > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.8. Add a partition to the topic

```
POST /kafka/partitions/add
```

Parameters

Type	Name	Description	Schema
Body	**addPartition** *required*	addPartition	AddPartition

Responses

HTTP Code	Description	Schema
200	OK	TopicMeta

HTTP Code	Description	Schema
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.9. Check the partition reassignment process

```
PUT /kafka/partitions/reassign/check
```

Parameters

Type	Name	Description	Schema
Body	**reassignStr** *required*	reassignStr	string

Responses

HTTP Code	Description	Schema
-1	Reassignment Failed	No Content
0	Reassignment In Progress	No Content
1	Reassignment Completed	No Content
200	OK	< string, integer(int32) > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.10. Execute the partition reassignment

```
PUT /kafka/partitions/reassign/execute
```

Parameters

Type	Name	Description	Schema
Body	**reassignStr** *required*	reassignStr	string

Responses

HTTP Code	Description	Schema
200	OK	< string, integer(int32) > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.11. Generate plan for the partition reassignment

```
POST /kafka/partitions/reassign/generate
```

Parameters

Type	Name	Description	Schema
Body	**reassignWrapper** *required*	reassignWrapper	ReassignWrapper

Responses

HTTP Code	Description	Schema
200	OK	< string > array
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.12. List topics

```
GET /kafka/topics
```

Responses

HTTP Code	Description	Schema
200	OK	< string > array
401	Unauthorized	No Content

HTTP Code	Description	Schema
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.13. Create a topic

```
POST /kafka/topics/create
```

Parameters

Type	Name	Description	Schema
Query	**reassignStr** *optional*	reassignStr	string
Body	**topic** *required*	topic	TopicDetail

Responses

HTTP Code	Description	Schema
201	Created	TopicMeta
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.14. Describe a topic by fetching the metadata and config

```
GET /kafka/topics/{topic}
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	TopicMeta
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.15. Delete a topic (you should enable topic deletion

```
DELETE /kafka/topics/{topic}
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	GeneralResponse
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.16. Create topic configs

```
POST /kafka/topics/{topic}/conf
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string
Body	**prop** *required*	prop	< string, object > map

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.17. Get topic configs

```
GET /kafka/topics/{topic}/conf
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.18. Update topic configs

```
PUT /kafka/topics/{topic}/conf
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string
Body	**prop** *required*	prop	< string, object > map

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.19. Delete topic configs

```
DELETE /kafka/topics/{topic}/conf
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string
Body	**delProps** *required*	delProps	< string > array

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.20. Get topic config by key

```
GET /kafka/topics/{topic}/conf/{key}
```

Parameters

Type	Name	Description	Schema
Path	**key** *required*	key	string
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.21. Delete a topic config by key

```
DELETE /kafka/topics/{topic}/conf/{key}
```

Parameters

Type	Name	Description	Schema
Path	**key** *required*	key	string
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	boolean
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.22. Create a topic config by key

```
POST /kafka/topics/{topic}/conf/{key}={value}
```

Parameters

Type	Name	Description	Schema
Path	**key** *required*	key	string
Path	**topic** *required*	topic	string
Path	**value** *required*	value	string

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.23. Update a topic config by key

```
PUT /kafka/topics/{topic}/conf/{key}={value}
```

Parameters

Type	Name	Description	Schema
Path	**key** *required*	key	string
Path	**topic** *required*	topic	string

Type	Name	Description	Schema
Path	**value** *required*	value	string

Responses

HTTP Code	Description	Schema
200	OK	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.24. Tell if a topic exists

```
GET /kafka/topics/{topic}/exist
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	boolean
401	Unauthorized	No Content
403	Forbidden	No Content

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.25. Write a message to the topic, for testing purpose

```
POST /kafka/topics/{topic}/write
```

Parameters

Type	Name	Description	Schema
Path	**topic** *required*	topic	string
Body	**message** *required*	message	string

Responses

HTTP Code	Description	Schema
201	Created	GeneralResponse
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `text/plain`

Produces

- `/`

4.1.26. List topics Brief

```
GET /kafka/topicsbrief
```

Responses

HTTP Code	Description	Schema
200	OK	< TopicBrief > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2. Zookeeper-controller

Zookeeper Controller

4.2.1. Get the connection state of zookeeper

```
GET /zk/connstate
```

Responses

HTTP Code	Description	Schema
200	OK	string
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.2. Get the environment information of zookeeper

```
GET /zk/env
```

Responses

HTTP Code	Description	Schema
200	OK	< string, ZkServerEnvironment > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.3. List a zookeeper path

```
GET /zk/ls/{path}
```

Parameters

Type	Name	Description	Schema
Path	**path** *required*	path	string

Responses

HTTP Code	Description	Schema
200	OK	< string > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.4. Get the service state of zookeeper

```
GET /zk/stat
```

Responses

HTTP Code	Description	Schema
200	OK	< string, ZkServerStat > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

Chapter 5. Definitions

5.1. AddPartition

Name	Schema
numPartitionsAdded *optional*	integer(int32)
replicaAssignment *optional*	string
topic *optional*	string

5.2. BrokerInfo

Name	Schema
endPoints *optional*	< string > array
host *optional*	string
id *optional*	integer(int32)
jmxPort *optional*	integer(int32)
port *optional*	integer(int32)
rack *optional*	string
securityProtocol *optional*	object
startTime *optional*	string(date-time)
version *optional*	integer(int32)

5.3. ConsumerGroupDesc

Name	Schema
consumerId *optional*	string
currentOffset *optional*	integer(int64)
groupName *optional*	string
host *optional*	string
lag *optional*	integer(int64)
logEndOffset *optional*	integer(int64)
partitionId *optional*	integer(int32)
state *optional*	enum (RUNNING, PENDING)
topic *optional*	string
type *optional*	enum (NEW, OLD)

5.4. GeneralResponse

Name	Schema
msg *optional*	string
state *optional*	enum (success, failure)

5.5. HostAndPort

Name	Schema
hostText *optional*	string
port *optional*	integer(int32)

5.6. Map«int,long»

Type : < string, integer(int64) > map

5.7. ReassignWrapper

Name	Schema
brokers *optional*	< integer(int32) > array
topics *optional*	< string > array

5.8. TopicAndPartition

Type : object

5.9. TopicBrief

Name	Schema
isrRate *optional*	integer(int64)
numPartition *optional*	integer(int32)
topic *optional*	string

5.10. TopicDetail

Name	Schema
factor *optional*	integer(int32)
name *optional*	string
partitions *optional*	integer(int32)
prop *optional*	< string, object > map

5.11. TopicMeta

Name	Schema
partitionCount *optional*	integer(int32)
replicationFactor *optional*	integer(int32)
topicCustomConfigs *optional*	< string, object > map
topicName *optional*	string
topicPartitionInfos *optional*	< TopicPartitionInfo > array

5.12. TopicPartitionInfo

Name	Schema
endOffset *optional*	integer(int64)
in_sync *optional*	boolean
isr *optional*	< string > array
leader *optional*	string
messageAvailable *optional*	integer(int64)
partitionId *optional*	integer(int32)
replicas *optional*	< string > array
startOffset *optional*	integer(int64)

5.13. ZkServerClient

Name	Schema
host *optional*	string
ops *optional*	integer(int32)
port *optional*	integer(int32)
queued *optional*	integer(int32)
received *optional*	integer(int32)
sent *optional*	integer(int32)

5.14. ZkServerEnvironment

Name	Schema
attributes *optional*	< string, string > map

5.15. ZkServerStat

Name	Schema
avgLatency *optional*	integer(int32)
buildDate *optional*	string
clients *optional*	< ZkServerClient > array
connections *optional*	integer(int32)
maxLatency *optional*	integer(int32)
minLatency *optional*	integer(int32)
mode *optional*	enum (Leader, Follower, Observer)

Name	Schema
nodes *optional*	integer(int32)
outstanding *optional*	integer(int32)
received *optional*	integer(int32)
sent *optional*	integer(int32)
version *optional*	string
zxId *optional*	string