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. getMessage

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

Parameters

Type	Name	Description	Schema
Query	**decoder** *optional*	decoder	string
Query	**offset** *optional*	offset	integer(int64)

Type	Name	Description	Schema
Query	**partition** *optional*	partition	integer(int32)
Query	**topic** *optional*	topic	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. deleteOldConsumerGroup

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

Parameters

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

Responses

HTTP Code	Description	Schema
200	OK	boolean
204	No Content	No Content

HTTP Code	Description	Schema
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
Query	**consumergroup** *optional*	consumergroup	string
Query	**topic** *optional*	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. resetOffset

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

Parameters

Type	Name	Description	Schema
Query	**consumergroup** *optional*	consumergroup	string
Query	**offset** *optional*	offset	string
Query	**partition** *optional*	partition	integer(int32)
Query	**topic** *optional*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	No Content
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 new consumer groups

```
GET /kafka/consumergroups/new
```

Parameters

Type	Name	Description	Schema
Query	**t** *optional*	t	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

- `/`

4.1.7. List new consumergroups lag and offset by consumergroup and topic

```
GET /kafka/consumergroups/new/{consumerGroup}/{topic}
```

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.1.8. List old consumer groups from zk

```
GET /kafka/consumergroups/old
```

Parameters

Type	Name	Description	Schema
Query	**t** *optional*	t	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

- /

4.1.9. List old consumergroups lag and offset by consumergroup and topic

```
GET /kafka/consumergroups/old/{consumerGroup}/{topic}
```

Parameters

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

Responses

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

Consumes

- application/json

Produces

- /

4.1.10. 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
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.11. 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

HTTP Code	Description	Schema
1	Reassignment Completed	No Content
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

- `/`

4.1.12. Execute the partition reassignment

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

Parameters

Type	Name	Description	Schema	Default
Query	**throttle** *optional*	throttle	integer(int64)	"-1"
Body	**reassignStr** *required*	reassignStr	string	

Responses

HTTP Code	Description	Schema
200	OK	< string, integer(int32) > map
201	Created	No Content
401	Unauthorized	No Content

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

Consumes

- `application/json`

Produces

- `/`

4.1.13. 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

- `/`

4.1.14. List topics

```
GET /kafka/topics
```

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

- `/`

4.1.15. Create a topic

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

Parameters

Type	Name	Description	Schema
Body	**topic** *required*	topic	TopicDetail

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.1.16. 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

- `/`

4.1.17. 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	boolean
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.18. 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

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

Consumes

- `application/json`

Produces

- `/`

4.1.19. 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

- `/`

4.1.20. 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

- `/`

4.1.21. Delete topic configs

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

Parameters

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

Type	Name	Description	Schema
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

- `/`

4.1.22. 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

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

Consumes

- `application/json`

Produces

- `/`

4.1.23. 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

- /

4.1.24. 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

- /

4.1.25. 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
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

- `/`

4.1.26. Tell if a topic exists

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.1.27. 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	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `text/plain`

Produces

- `/`

4.1.28. 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. zkConnState

```
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. getEnv

```
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. ls

```
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. ping

```
GET /zk/ping
```

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

- `/`

4.2.5. getStat

```
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)
host *optional*	string
lag *optional*	integer(int64)
logEndOffset *optional*	integer(int64)
partitionId *optional*	integer(int32)
state *optional*	enum (RUNNING, PENDING)
topic *optional*	string

5.4. HostAndPort

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

5.5. Map«int,long»

Type : < string, integer(int64) > map

5.6. ReassignWrapper

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

5.7. TopicAndPartition

Type : object

5.8. TopicBrief

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

5.9. TopicDetail

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

5.10. TopicMeta

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

Name	Schema
topicName *optional*	string
topicPartitionInfos *optional*	< TopicPartitionInfo > array

5.11. 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)

5.12. ZkServerClient

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

Name	Schema
received *optional*	integer(int32)
sent *optional*	integer(int32)

5.13. ZkServerEnvironment

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

5.14. 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)
nodes *optional*	integer(int32)
outstanding *optional*	integer(int32)
received *optional*	integer(int32)
sent *optional*	integer(int32)

Name	Schema
version *optional*	string
zxId *optional*	string