

Kafka REST API SwaggerUI

Table of Contents

1. Overview	1
1.1. Version information	1
1.2. Contact information.....	1
1.3. URI scheme.....	1
1.4. Tags	1
2. Chapter of manual content 1	2
2.1. Sub chapter	2
3. Chapter of manual content 2	3
4. Resources	4
4.1. Collector-controller	4
4.1.1. Fetch all JMX metric data	4
4.1.2. Fetch JMX metric data with query filter. You can get the query filter template through the API /jmx/v2/filters.	4
4.1.3. List the query filter templates with the filterKey. If filterKey is set to empty, it will return all the templates.	5
4.2. Kafka-controller	6
4.2.1. List brokers in this cluster	6
4.2.2. List log dirs by broker list	7
4.2.3. Describe log dirs by broker list and topic list.....	7
4.2.4. Describe replica log dirs.....	8
4.2.5. Get broker configs, including dynamic configs	9
4.2.6. Get broker dynamic configs	10
4.2.7. Update broker configs	10
4.2.8. Remove broker dynamic configs	11
4.2.9. Describe cluster, nodes, controller info.	12
4.2.10. Get the message from the offset of the partition in the topic, decoder is not supported yet	12
4.2.11. Delete Consumer Group	13
4.2.12. getLastCommitTimestamp	14
4.2.13. Reset consumer group offset, earliest/latest can be used. Support reset by time for new consumer group, pass a parameter that satisfies yyyy-MM-dd HH:mm:ss.SSS to offset.	15
4.2.14. List all consumer groups from zk and kafka	16
4.2.15. Get the meta data of the specified new consumer groups, including state, coordinator, assignmentStrategy, members	16
4.2.16. Get the meta data of the specify new consumer group, including state, coordinator, assignmentStrategy, members	17
4.2.17. Describe consumer group, showing lag and offset, may be slow if multi topics are listened	18

4.2.18. Get the topics involved of the specify consumer group	19
4.2.19. Describe consumer groups by topic, showing lag and offset	19
4.2.20. Get controller in this cluster	20
4.2.21. Check the cluster health.....	21
4.2.22. Add partitions to the topics	21
4.2.23. Check the partition reassignment process	22
4.2.24. Execute the partition reassignment.....	23
4.2.25. Generate plan for the partition reassignment.....	24
4.2.26. List topics	25
4.2.27. Delete a topic list (you should enable topic deletion	25
4.2.28. Create topics	26
4.2.29. Describe a topic by fetching the metadata and config	27
4.2.30. Get topic configs	27
4.2.31. Update topic configs.....	28
4.2.32. Get topic config by key	29
4.2.33. Update a topic config by key	30
4.2.34. Get topic dyn configs	30
4.2.35. Tell if a topic exists	31
4.2.36. List topics Brief	32
4.3. User-controller	32
4.3.1. Add user.....	32
4.3.2. Get user list.	33
4.3.3. Modify user information.	34
4.3.4. Delete user.....	34
4.4. Zookeeper-controller.....	35
4.4.1. Get the connection state of zookeeper.....	35
4.4.2. Get the environment information of zookeeper.....	36
4.4.3. Get data of a zookeeper path	36
4.4.4. List a zookeeper path.....	37
4.4.5. Get the service state of zookeeper	37
5. Definitions	39
5.1. AddPartition	39
5.2. BrokerInfo	39
5.3. ClusterInfo	39
5.4. ConsumerGroupDesc.....	40
5.5. ConsumerGroupMeta	41
5.6. CustomConfigEntry	41
5.7. CustomTopicPartitionInfo	41

5.8. GeneralResponse	42
5.9. HashMap«string,object»	42
5.10. HealthCheckResult	42
5.11. HostAndPort	43
5.12. JMXConfiguration	43
5.13. JMXFilter	43
5.14. JMXMetricData	44
5.15. JMXMetricDataV1	44
5.16. JMXQuery	44
5.17. LogDirInfo	44
5.18. Map«int,long»	47
5.19. Map«string,LogDirInfo»	47
5.20. MemberDescription	47
5.21. Node	47
5.22. Pattern	48
5.23. ReassignModel	48
5.24. ReassignStatus	48
5.25. ReassignWrapper	49
5.26. Record	49
5.27. ReplicaInfo	49
5.28. ReplicaLogDirInfo	49
5.29. TopicBrief	50
5.30. TopicDetail	50
5.31. TopicMeta	50
5.32. TopicPartition	51
5.33. TopicPartitionInfo	51
5.34. TopicPartitionReplica	51
5.35. TopicPartitionReplicaAssignment	52
5.36. User	52
5.37. ZkServerClient	52
5.38. ZkServerEnvironment	53
5.39. ZkServerStat	53

Chapter 1. Overview

Kafka REST API SwaggerUI

1.1. Version information

Version : 0.1.0

1.2. Contact information

Contact : gnuhpc

Contact Email : gnuahpc@gmail.com

1.3. URI scheme

Host : localhost:8080

BasePath : /

1.4. Tags

- collector-controller : Rest API for Collecting JMX Metric Data
- kafka-controller : Kafka Controller
- user-controller : Security User Management 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. Collector-controller

Rest API for Collecting JMX Metric Data

4.1.1. Fetch all JMX metric data

```
GET /jmx/v1
```

Parameters

Type	Name	Description	Schema
Query	jmxurl <i>optional</i>	Parameter jmxurl should be a comma-separated list of {IP:Port} or set to 'default'	string

Responses

HTTP Code	Description	Schema
200	OK	< JMXMetricDataV1 > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.1.2. Fetch JMX metric data with query filter. You can get the query filter template through the API `/jmx/v2/filters`.

```
POST /jmx/v2
```


Parameters

Type	Name	Description	Schema
Query	jmxurl <i>optional</i>	Parameter jmxurl should be a comma-separated list of {IP:Port} or set to 'default'	string
Body	jmxQuery <i>required</i>	jmxQuery	JMXQuery

Responses

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

Consumes

- [application/json](#)

Produces

- [/](#)

4.1.3. List the query filter templates with the filterKey. If filterKey is set to empty, it will return all the templates.

```
GET /jmx/v2/filters
```

Parameters

Type	Name	Description	Schema
Query	filterKey <i>required</i>	filterKey	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

- `/`

4.2. Kafka-controller

Kafka Controller

4.2.1. 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

- `/`

4.2.2. List log dirs by broker list

```
GET /kafka/brokers/logdirs
```

Parameters

Type	Name	Description	Schema
Query	brokerList <i>optional</i>	brokerList	< integer(int32) > array(multi)

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

- /

4.2.3. Describe log dirs by broker list and topic list

```
GET /kafka/brokers/logdirs/detail
```

Parameters

Type	Name	Description	Schema
Query	brokerList <i>optional</i>	brokerList	< integer(int32) > array(multi)
Query	logDirList <i>optional</i>	logDirList	< string > array(multi)

Type	Name	Description	Schema
Query	topicList <i>optional</i>	topicList	< string > array(multi)

Responses

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

Consumes

- [application/json](#)

Produces

- [/](#)

4.2.4. Describe replica log dirs.

```
GET /kafka/brokers/replicalogdirs
```

Parameters

Type	Name	Description	Schema
Query	replicas <i>required</i>	replicas	string

Responses

HTTP Code	Description	Schema
200	OK	< string, ReplicaLogDirInfo > map
401	Unauthorized	No Content

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

Consumes

- `application/json`

Produces

- `/`

4.2.5. Get broker configs, including dynamic configs

```
GET /kafka/brokers/{brokerId}/conf
```

Parameters

Type	Name	Description	Schema
Path	brokerId <i>required</i>	brokerId	integer(int32)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.6. Get broker dynamic configs

```
GET /kafka/brokers/{brokerId}/dynconf
```

Parameters

Type	Name	Description	Schema
Path	brokerId <i>required</i>	brokerId	integer(int32)

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

- `/`

4.2.7. Update broker configs

```
PUT /kafka/brokers/{brokerId}/dynconf
```

Parameters

Type	Name	Description	Schema
Path	brokerId <i>required</i>	brokerId	integer(int32)
Body	props <i>required</i>	props	< 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

- `/`

4.2.8. Remove broker dynamic configs

```
DELETE /kafka/brokers/{brokerId}/dynconf
```

Parameters

Type	Name	Description	Schema
Path	brokerId <i>required</i>	brokerId	integer(int32)
Query	configKeysToBeRemoved <i>required</i>	configKeysToBeRemoved	< string > array(multi)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.9. Describe cluster, nodes, controller info.

```
GET /kafka/cluster
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.10. 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 <i>required</i>	offset	integer(int64)
Path	partition <i>required</i>	partition	integer(int32)

Type	Name	Description	Schema
Path	topic <i>required</i>	topic	string
Query	decoder <i>optional</i>	decoder	string

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.11. Delete Consumer Group

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

Parameters

Type	Name	Description	Schema
Path	consumergroup <i>required</i>	consumergroup	string
Path	type <i>required</i>	type	enum (NEW, OLD)

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

- `/`

4.2.12. getLastCommitTimestamp

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

Parameters

Type	Name	Description	Schema
Path	consumergroup <i>required</i>	consumergroup	string
Path	topic <i>required</i>	topic	string
Path	type <i>required</i>	type	enum (NEW, OLD)

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

- `/`

4.2.13. Reset consumer group offset, earliest/latest can be used. Support reset by time for new consumer group, pass a parameter that satisfies yyyy-MM-dd HH:mm:ss.SSS to offset.

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

Parameters

Type	Name	Description	Schema
Path	consumergroup <i>required</i>	consumergroup	string
Path	offset <i>optional</i>	[earliest/latest/{long}]/yyyy-MM-dd HH:mm:ss.SSS] can be supported. The date type is only valid for new consumer group.	string
Path	partition <i>required</i>	partition	integer(int32)
Path	topic <i>required</i>	topic	string
Path	type <i>required</i>	type	enum (NEW, OLD)

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

- `/`

4.2.14. List all consumer groups from zk and kafka

```
GET /kafka/consumergroups
```

Parameters

Type	Name	Description	Schema
Query	topic <i>optional</i>	topic	string
Query	type <i>optional</i>	type	enum (NEW, OLD)

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

- `/`

4.2.15. Get the meta data of the specified new consumer groups, including state, coordinator, assignmentStrategy, members

```
GET /kafka/consumergroups/meta
```

Parameters

Type	Name	Description	Schema
Query	consumerGroupList <i>required</i>	consumerGroupList	< string > array(multi)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.16. Get the meta data of the specify new consumer group, including state, coordinator, assignmentStrategy, members

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

Parameters

Type	Name	Description	Schema
Path	consumerGroup <i>required</i>	consumerGroup	string

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.17. Describe consumer group, showing lag and offset, may be slow if multiple topics are listened

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

Parameters

Type	Name	Description	Schema
Path	consumerGroup <i>required</i>	consumerGroup	string
Path	type <i>required</i>	type	enum (NEW, OLD)

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

- `/`

4.2.18. Get the topics involved of the specify consumer group

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

Parameters

Type	Name	Description	Schema
Path	consumerGroup <i>required</i>	consumerGroup	string
Path	type <i>required</i>	type	enum (NEW, OLD)

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

- `/`

4.2.19. Describe consumer groups by topic, showing lag and offset

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	topic	string
Path	type <i>required</i>	type	enum (NEW, OLD)
Query	consumerGroup <i>optional</i>	consumerGroup	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

- `/`

4.2.20. Get controller in this cluster

```
GET /kafka/controller
```

Responses

HTTP Code	Description	Schema
200	OK	Node
401	Unauthorized	No Content

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

Consumes

- `application/json`

Produces

- `/`

4.2.21. Check the cluster health.

```
GET /kafka/health
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.22. Add partitions to the topics

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

Parameters

Type	Name	Description	Schema
Body	addPartitions <i>required</i>	addPartitions	< AddPartition > array

Responses

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

Consumes

- [application/json](#)

Produces

- [/](#)

4.2.23. Check the partition reassignment process

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

Parameters

Type	Name	Description	Schema
Body	reassign <i>required</i>	reassign	ReassignModel

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	ReassignStatus
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.24. Execute the partition reassignment

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

Parameters

Type	Name	Description	Schema
Query	interBrokerThrottle <i>optional</i>	interBrokerThrottle	integer(int64)
Query	replicaAlterLogDirsThrottle <i>optional</i>	replicaAlterLogDirsThrottle	integer(int64)
Query	timeoutMs <i>optional</i>	timeoutMs	integer(int64)
Body	reassign <i>required</i>	reassign	ReassignModel

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.25. Generate plan for the partition reassignment

POST /kafka/partitions/reassign/generate

Parameters

Type	Name	Description	Schema
Body	reassignWrapper <i>required</i>	reassignWrapper	ReassignWrapper

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.26. 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

- `/`

4.2.27. Delete a topic list (you should enable topic deletion

```
DELETE /kafka/topics
```

Parameters

Type	Name	Description	Schema
Query	topicList <i>required</i>	topicList	< string > array(multi)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.28. Create topics

POST /kafka/topics/create

Parameters

Type	Name	Description	Schema
Body	topicList <i>required</i>	topicList	< TopicDetail > array

Responses

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

Consumes

- `application/json`

Produces

- /

4.2.29. Describe a topic by fetching the metadata and config

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	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

- /

4.2.30. Get topic configs

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	topic	string

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.31. Update topic configs

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	topic	string
Body	props <i>required</i>	props	< string, object > map

Responses

HTTP Code	Description	Schema
200	OK	< CustomConfigEntry > array
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

- `/`

4.2.32. Get topic config by key

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

Parameters

Type	Name	Description	Schema
Path	key <i>required</i>	key	string
Path	topic <i>required</i>	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

- `/`

4.2.33. Update a topic config by key

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

Parameters

Type	Name	Description	Schema
Path	key <i>required</i>	key	string
Path	topic <i>required</i>	topic	string
Path	value <i>required</i>	value	string

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.34. Get topic dyn configs

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	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

- `/`

4.2.35. Tell if a topic exists

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

Parameters

Type	Name	Description	Schema
Path	topic <i>required</i>	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

- `/`

4.2.36. 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

- `/`

4.3. User-controller

Security User Management Controller.

4.3.1. Add user.

POST /users

Parameters

Type	Name	Description	Schema
Body	user <i>required</i>	user	User

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

- `/`

4.3.2. Get user list.

GET /users

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

- /

4.3.3. Modify user information.

PUT /users

Parameters

Type	Name	Description	Schema
Body	user <i>required</i>	user	User

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

- /

4.3.4. Delete user.

DELETE /users/{username}

Parameters

Type	Name	Description	Schema
Path	username <i>required</i>	username	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

- `/`

4.4. Zookeeper-controller

Zookeeper Controller

4.4.1. 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

- `/`

4.4.2. 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

- `/`

4.4.3. Get data of a zookeeper path

GET /zk/get/path

Parameters

Type	Name	Description	Schema
Query	path <i>required</i>	path	string

Responses

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

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

Consumes

- `application/json`

Produces

- `/`

4.4.4. List a zookeeper path

```
GET /zk/ls/path
```

Parameters

Type	Name	Description	Schema
Query	path <i>required</i>	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

- `/`

4.4.5. 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 <i>optional</i>	integer(int32)
replicaAssignment <i>optional</i>	< < integer(int32) > array > array
topic <i>optional</i>	string

5.2. BrokerInfo

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

5.3. ClusterInfo

Name	Schema
clusterId <i>optional</i>	string
controller <i>optional</i>	Node
nodes <i>optional</i>	< Node > array

5.4. ConsumerGroupDesc

Name	Schema
assignmentStrategy <i>optional</i>	string
clientId <i>optional</i>	string
consumerId <i>optional</i>	string
coordinator <i>optional</i>	Node
currentOffset <i>optional</i>	integer(int64)
groupName <i>optional</i>	string
host <i>optional</i>	string
lag <i>optional</i>	integer(int64)
logEndOffset <i>optional</i>	integer(int64)
partitionId <i>optional</i>	integer(int32)
state <i>optional</i>	enum (Unknown, PreparingRebalance, CompletingRebalance, Stable, Dead, Empty)
topic <i>optional</i>	string
type <i>optional</i>	enum (NEW, OLD)

5.5. ConsumerGroupMeta

Name	Schema
assignmentStrategy <i>optional</i>	string
coordinator <i>optional</i>	Node
groupId <i>optional</i>	string
members <i>optional</i>	< MemberDescription > array
state <i>optional</i>	enum (Unknown, PreparingRebalance, CompletingRebalance, Stable, Dead, Empty)

5.6. CustomConfigEntry

Name	Schema
isReadOnly <i>optional</i>	boolean
isSensitive <i>optional</i>	boolean
name <i>optional</i>	string
readOnly <i>optional</i>	boolean
sensitive <i>optional</i>	boolean
source <i>optional</i>	enum (DYNAMIC_TOPIC_CONFIG, DYNAMIC_BROKER_CONFIG, DYNAMIC_DEFAULT_BROKER_CONFIG, STATIC_BROKER_CONFIG, DEFAULT_CONFIG, UNKNOWN)
value <i>optional</i>	string

5.7. CustomTopicPartitionInfo

Name	Schema
endOffset <i>optional</i>	integer(int64)
in_sync <i>optional</i>	boolean
messageAvailable <i>optional</i>	integer(int64)
startOffset <i>optional</i>	integer(int64)
topicPartitionInfo <i>optional</i>	TopicPartitionInfo

5.8. GeneralResponse

Name	Schema
data <i>optional</i>	object
msg <i>optional</i>	string
state <i>optional</i>	enum (success, failure)

5.9. HashMap«string,object»

Type : < string, object > map

5.10. HealthCheckResult

Name	Description	Schema
msg <i>optional</i>		string
status <i>optional</i>		string
timestamp <i>optional</i>	Example : "yyyy-MM-dd HH:mm:ss"	string

5.11. HostAndPort

Name	Schema
hasBracketlessColons <i>optional</i>	boolean
host <i>optional</i>	string
hostText <i>optional</i>	string
port <i>optional</i>	integer(int32)

5.12. JMXConfiguration

Name	Schema
exclude <i>optional</i>	JMXFilter
include <i>optional</i>	JMXFilter

5.13. JMXFilter

Name	Schema
attribute <i>optional</i>	object
beanNames <i>optional</i>	< string > array
beanRegexes <i>optional</i>	< Pattern > array
domain <i>optional</i>	string
domainRegex <i>optional</i>	Pattern
emptyBeanName <i>optional</i>	boolean
filter <i>optional</i>	< string, object > map

5.14. JMXMetricData

Name	Description	Schema
collected <i>optional</i>		boolean
host <i>optional</i>		string
metrics <i>optional</i>		< HashMap«string,object» > array
msg <i>optional</i>		string
timestamp <i>optional</i>	Example : "yyyy-MM-dd HH:mm:ss"	string

5.15. JMXMetricDataV1

Name	Description	Schema
collected <i>optional</i>		boolean
host <i>optional</i>		string
mbeanInfo <i>optional</i>		object
msg <i>optional</i>		string
timestamp <i>optional</i>	Example : "yyyy-MM-dd HH:mm:ss"	string

5.16. JMXQuery

Name	Schema
filters <i>optional</i>	< JMXConfiguration > array

5.17. LogDirInfo

Name	Schema
error <i>optional</i>	enum (UNKNOWN_SERVER_ERROR, NONE, OFFSET_OUT_OF_RANGE, CORRUPT_MESSAGE, UNKNOWN_TOPIC_OR_PARTITION, INVALID_FETCH_SIZE, LEADER_NOT_AVAILABLE, NOT_LEADER_FOR_PARTITION, REQUEST_TIMED_OUT, BROKER_NOT_AVAILABLE, REPLICA_NOT_AVAILABLE, MESSAGE_TOO_LARGE, STALE_CONTROLLER_EPOCH, OFFSET_METADATA_TOO_LARGE, NETWORK_EXCEPTION, COORDINATOR_LOAD_IN_PROGRESS, COORDINATOR_NOT_AVAILABLE, NOT_COORDINATOR, INVALID_TOPIC_EXCEPTION, RECORD_LIST_TOO_LARGE, NOT_ENOUGH_REPLICAS, NOT_ENOUGH_REPLICAS_AFTER_APPEND, INVALID_REQUIRED_ACKS, ILLEGAL_GENERATION, INCONSISTENT_GROUP_PROTOCOL, INVALID_GROUP_ID, UNKNOWN_MEMBER_ID, INVALID_SESSION_TIMEOUT, REBALANCE_IN_PROGRESS, INVALID_COMMIT_OFFSET_SIZE, TOPIC_AUTHORIZATION_FAILED, GROUP_AUTHORIZATION_FAILED, CLUSTER_AUTHORIZATION_FAILED, INVALID_TIMESTAMP, UNSUPPORTED_SASL_MECHANISM, ILLEGAL_SASL_STATE, UNSUPPORTED_VERSION, TOPIC_ALREADY_EXISTS, INVALID_PARTITIONS, INVALID_REPLICATION_FACTOR, INVALID_REPLICA_ASSIGNMENT, INVALID_CONFIG, NOT_CONTROLLER, INVALID_REQUEST, UNSUPPORTED_FOR_MESSAGE_FORMAT, POLICY_VIOLATION, OUT_OF_ORDER_SEQUENCE_NUMBER, DUPLICATE_SEQUENCE_NUMBER, INVALID_PRODUCER_EPOCH, INVALID_TXN_STATE, INVALID_PRODUCER_ID_MAPPING, INVALID_TRANSACTION_TIMEOUT, CONCURRENT_TRANSACTIONS, TRANSACTION_COORDINATOR_FENCED, TRANSACTIONAL_ID_AUTHORIZATION_FAILED, SECURITY_DISABLED, OPERATION_NOT_ATTEMPTED, KAFKA_STORAGE_ERROR, LOG_DIR_NOT_FOUND, SASL_AUTHENTICATION_FAILED, UNKNOWN_PRODUCER_ID, REASSIGNMENT_IN_PROGRESS, DELEGATION_TOKEN_AUTH_DISABLED, DELEGATION_TOKEN_NOT_FOUND, DELEGATION_TOKEN_OWNER_MISMATCH, DELEGATION_TOKEN_REQUEST_NOT_ALLOWED, DELEGATION_TOKEN_AUTHORIZATION_FAILED,

Name	Schema
replicaInfos <i>optional</i>	< string, ReplicaInfo > map

5.18. Map«int,long»

Type : < string, integer(int64) > map

5.19. Map«string,LogDirInfo»

Type : < string, [LogDirInfo](#) > map

5.20. MemberDescription

Name	Schema
assignment <i>optional</i>	< TopicPartition > array
clientId <i>optional</i>	string
host <i>optional</i>	string
memberId <i>optional</i>	string

5.21. Node

Name	Schema
empty <i>optional</i>	boolean
hash <i>optional</i>	integer(int32)
host <i>optional</i>	string
id <i>optional</i>	integer(int32)
idString <i>optional</i>	string

Name	Schema
port <i>optional</i>	integer(int32)
rack <i>optional</i>	string

5.22. Pattern

Name	Schema
cursor <i>optional</i>	integer(int32)
flags <i>optional</i>	integer(int32)
pattern <i>optional</i>	string

5.23. ReassignModel

Name	Schema
partitions <i>optional</i>	< TopicPartitionReplicaAssignment > array
version <i>optional</i>	integer(int32)

5.24. ReassignStatus

Name	Schema
msg <i>optional</i>	string
partitionsReassignStatus <i>optional</i>	< string, integer(int32) > map
removeThrottle <i>optional</i>	boolean
replicasReassignStatus <i>optional</i>	< string, integer(int32) > map

5.25. ReassignWrapper

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

5.26. Record

Name	Schema
decoder <i>optional</i>	string
key <i>optional</i>	object
offset <i>optional</i>	integer(int64)
timestamp <i>optional</i>	integer(int64)
topic <i>optional</i>	string
value <i>optional</i>	string

5.27. ReplicaInfo

Name	Schema
isFuture <i>optional</i>	boolean
offsetLag <i>optional</i>	integer(int64)
size <i>optional</i>	integer(int64)

5.28. ReplicaLogDirInfo

Name	Schema
currentReplicaLogDir <i>optional</i>	string
currentReplicaOffsetLag <i>optional</i>	integer(int64)
futureReplicaLogDir <i>optional</i>	string
futureReplicaOffsetLag <i>optional</i>	integer(int64)

5.29. TopicBrief

Name	Schema
isrRate <i>optional</i>	number(double)
numPartition <i>optional</i>	integer(int32)
topic <i>optional</i>	string

5.30. TopicDetail

Name	Schema
factor <i>optional</i>	integer(int32)
name <i>optional</i>	string
partitions <i>optional</i>	integer(int32)
prop <i>optional</i>	< string, object > map
replicasAssignments <i>optional</i>	< string, < integer(int32) > array > map

5.31. TopicMeta

Name	Schema
internal <i>optional</i>	boolean
partitionCount <i>optional</i>	integer(int32)
replicationFactor <i>optional</i>	integer(int32)
topicName <i>optional</i>	string
topicPartitionInfos <i>optional</i>	< CustomTopicPartitionInfo > array

5.32. TopicPartition

Name	Schema
hash <i>optional</i>	integer(int32)
partition <i>optional</i>	integer(int32)
topic <i>optional</i>	string

5.33. TopicPartitionInfo

Name	Schema
isr <i>optional</i>	< Node > array
leader <i>optional</i>	Node
partition <i>optional</i>	integer(int32)
replicas <i>optional</i>	< Node > array

5.34. TopicPartitionReplica

Name	Schema
brokerId <i>optional</i>	integer(int32)
hash <i>optional</i>	integer(int32)
partition <i>optional</i>	integer(int32)
topic <i>optional</i>	string

5.35. TopicPartitionReplicaAssignment

Name	Schema
log_dirs <i>optional</i>	< string > array
partition <i>optional</i>	integer(int32)
replicas <i>optional</i>	< integer(int32) > array
topic <i>optional</i>	string

5.36. User

Name	Schema
password <i>optional</i>	string
role <i>optional</i>	string
username <i>optional</i>	string

5.37. ZkServerClient

Name	Schema
host <i>optional</i>	string

Name	Schema
ops <i>optional</i>	integer(int32)
port <i>optional</i>	integer(int32)
queued <i>optional</i>	integer(int32)
received <i>optional</i>	integer(int32)
sent <i>optional</i>	integer(int32)

5.38. ZkServerEnvironment

Name	Schema
attributes <i>optional</i>	< string, string > map

5.39. ZkServerStat

Name	Schema
avgLatency <i>optional</i>	integer(int32)
buildDate <i>optional</i>	string
clients <i>optional</i>	< ZkServerClient > array
connections <i>optional</i>	integer(int32)
maxLatency <i>optional</i>	integer(int32)
minLatency <i>optional</i>	integer(int32)
mode <i>optional</i>	enum (Leader, Follower, Observer, Standalone, Down, Unknow)
msg <i>optional</i>	string

Name	Schema
nodes <i>optional</i>	integer(int32)
outstanding <i>optional</i>	integer(int32)
received <i>optional</i>	integer(int32)
sent <i>optional</i>	integer(int32)
version <i>optional</i>	string
zxId <i>optional</i>	string