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 dir.. 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 12
      4.2.11. Delete Consumer Group 14
      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 all the meta data of new consumer groups, including state, coordinator, assignmentStrategy, members 17
      4.2.16. Get the meta data of the specify new consumer group, including state, coordinator, assignmentStrategy, members 18
      4.2.17. Describe consumer group, showing lag and offset, may be slow if multi topics are listened18

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 20
4.2.20. Get controller in this cluster 21
4.2.21. Check the cluster health. 21
4.2.22. Add partitions to the topics 22
4.2.23. Move partition leader to preferred replica. 22
4.2.24. Check the partition reassignment process 23
4.2.25. Execute the partition reassignment 24
4.2.26. Generate plan for the partition reassignment 25
4.2.27. Stop the partition reassignment process 26
4.2.28. List topics 26
4.2.29. Delete a topic list (you should enable topic deletion 27
4.2.30. Create topics 28
4.2.31. Create topics check 28
4.2.32. Describe a topic by fetching the metadata and config 29
4.2.33. Get topic configs 30
4.2.34. Update topic configs 30
4.2.35. Get topic config by key 31
4.2.36. Update a topic config by key 32
4.2.37. Get topic dyn configs 33
4.2.38. Tell if a topic exists 33
4.2.39. List topics Brief 34
4.3. Schema-registry-controller 35
4.3.1. Get schema by id 35
4.3.2. List all subjects 35
4.3.3. Check if a schema has already been registered under the specified subject 36
4.3.4. Get latest schema by subject 37
4.3.5. Delete the specified subject and its associated compatibility level if registered. 38
4.3.6. Register schema by subject 38
4.3.7. Get all versions for the specified subject 39
4.3.8. Get schema by subject and version 40
4.4. User-controller 40
4.4.1. Add user. 41
4.4.2. Get user list. 41
4.4.3. Modify user information. 42
4.4.4. Delete user. 42
4.5. Zookeeper-controller 43
4.5.1. Get the connection state of zookeeper 43

4.5.2. Get the environment information of zookeeper . . . 44
4.5.3. Get data of a zookeeper path . . . 44
4.5.4. List a zookeeper path . . . 45
4.5.5. Get the service state of zookeeper . . . 46
5. Definitions . . . 47
5.1. AddPartition . . . 47
5.2. BrokerInfo . . . 47
5.3. ClusterInfo . . . 47
5.4. ConsumerGroupDesc . . . 48
5.5. ConsumerGroupMeta . . . 49
5.6. CustomConfigEntry . . . 49
5.7. CustomTopicPartitionInfo . . . 49
5.8. GeneralResponse . . . 50
5.9. HashMap«string,object» . . . 50
5.10. HealthCheckResult . . . 50
5.11. HostAndPort . . . 51
5.12. JMXConfiguration . . . 51
5.13. JMXFilter . . . 51
5.14. JMXMetricData . . . 52
5.15. JMXMetricDataV1 . . . 52
5.16. JMXQuery . . . 52
5.17. LogDirInfo . . . 52
5.18. Map«int,long» . . . 55
5.19. Map«string,LogDirInfo» . . . 55
5.20. MemberDescription . . . 55
5.21. Node . . . 55
5.22. Pattern . . . 56
5.23. ReassignModel . . . 56
5.24. ReassignStatus . . . 56
5.25. ReassignWrapper . . . 57
5.26. Record . . . 57
5.27. ReplicaInfo . . . 57
5.28. ReplicaLogDirInfo . . . 58
5.29. SchemaMetadata . . . 58
5.30. SchemaRegistryMetadata . . . 58
5.31. TopicBrief . . . 58
5.32. TopicDetail . . . 59
5.33. TopicMeta . . . 59

5.34. TopicPartition 60
5.35. TopicPartitionInfo 60
5.36. TopicPartitionReplicaAssignment 60
5.37. User 60
5.38. ZkServerClient 61
5.39. ZkServerEnvironment 61
5.40. ZkServerStat 61

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

- collector-controller : Rest API for Collecting JMX Metric Data
- kafka-controller : Kafka Controller
- schema-registry-controller : Schema Registry 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** *optional*	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** *optional*	Parameter jmxurl should be a comma-separated list of {IP:Port} or set to 'default'	string
Body	**jmxQuery** *required*	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** *required*	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** *optional*	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

```
POST /kafka/brokers/logdirs/detail
```

Parameters

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

Type	Name	Description	Schema
Body	**topicPartition Map** *optional*	topicPartitionMap	< string, < integer(int32) > array > map

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.4. Describe replica log dir.

```
GET /kafka/brokers/replicalogdir/{brokerId}/{topic}/{partition}
```

Parameters

Type	Name	Description	Schema
Path	**brokerId** *required*	brokerId	integer(int32)
Path	**partition** *required*	partition	integer(int32)
Path	**topic** *required*	topic	string

Responses

HTTP Code	Description	Schema
200	OK	ReplicaLogDirInfo
401	Unauthorized	No Content
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** *required*	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** *required*	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** *required*	brokerId	integer(int32)

Type	Name	Description	Schema
Body	**props** *required*	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** *required*	brokerId	integer(int32)
Query	**configKeysToBeRemoved** *required*	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

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

Parameters

Type	Name	Description	Schema	Default
Path	**offset** *optional*	[long/yyyy-MM-dd HH:mm:ss.SSS] can be supported.	string	
Path	**partition** *required*	partition	integer(int32)	
Path	**topic** *required*	topic	string	
Query	**avroSchema** *optional*	avroSchema	string	
Query	**fetchTimeoutMs** *optional*	fetchTimeoutMs	integer(int64)	`"30000"`
Query	**keyDecoder** *optional*	keyDecoder	string	`"StringDeserializer"`
Query	**maxRecords** *optional*	maxRecords	integer(int32)	`"10"`
Query	**valueDecoder** *optional*	valueDecoder	string	`"StringDeserializer"`

Responses

HTTP Code	Description	Schema
200	OK	< Record > array
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** *required*	consumergroup	string
Path	**type** *required*	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** *required*	consumergroup	string

Type	Name	Description	Schema
Path	**topic** *required*	topic	string
Path	**type** *required*	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** *required*	consumergroup	string
Path	**offset** *optional*	[earliest/latest/{long}/yyyy-MM-dd HH:mm:ss.SSS] can be supported. The date type is only valid for new consumer group.	string

Type	Name	Description	Schema
Path	**partition** *required*	partition	integer(int32)
Path	**topic** *required*	topic	string
Path	**type** *required*	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** *optional*	topic	string
Query	**type** *optional*	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 all the meta data of new consumer groups, including state, coordinator, assignmentStrategy, members

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

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** *required*	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 multi topics are listened

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

Parameters

Type	Name	Description	Schema
Path	**consumerGroup** *required*	consumerGroup	string
Path	**type** *required*	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** *required*	consumerGroup	string
Path	**type** *required*	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** *required*	topic	string
Path	**type** *required*	type	enum (NEW, OLD)
Query	**consumerGroup** *optional*	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
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** *required*	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. Move partition leader to preferred replica.

```
PUT /kafka/partitions/preferredreplica/elect
```

Parameters

Type	Name	Description	Schema
Body	**partitionList** *required*	partitionList	< TopicPartition > array

Responses

HTTP Code	Description	Schema
-1	Other preferred replica elect is in progress	No Content
-2	Partition doesn’t exist	No Content
0	Successfully started preferred replica election	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

- `/`

4.2.24. Check the partition reassignment process

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

Parameters

Type	Name	Description	Schema
Body	**reassign** *required*	reassign	ReassignModel

Responses

HTTP Code	Description	Schema
-1	Reassignment Failed	No Content
0	Reassignment In Progress	No Content
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

- `/`

4.2.25. Execute the partition reassignment

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

Parameters

Type	Name	Description	Schema	Default
Query	**interBrokerThrottle** *optional*	interBrokerThrottle	integer(int64)	`"-1"`
Query	**replicaAlterLogDirsThrottle** *optional*	replicaAlterLogDirsThrottle	integer(int64)	`"-1"`
Query	**timeoutMs** *optional*	timeoutMs	integer(int64)	`"10000"`
Body	**reassign** *required*	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

- `/`

4.2.26. 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	< ReassignModel > array
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.27. Stop the partition reassignment process

```
PUT /kafka/partitions/reassign/stop
```

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

- `/`

4.2.28. List topics

```
GET /kafka/topics
```

Responses

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

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

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

```
DELETE /kafka/topics
```

Parameters

Type	Name	Description	Schema
Query	**topicList** *required*	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

- `/`

4.2.30. Create topics

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

Parameters

Type	Name	Description	Schema
Body	**topicList** *required*	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

- `/`

4.2.31. Create topics check

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

Parameters

Type	Name	Description	Schema
Body	**topicList** *required*	topicList	< TopicDetail > array

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.32. 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

- `/`

4.2.33. 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	< CustomConfigEntry > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.34. Update topic configs

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

Parameters

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

Type	Name	Description	Schema
Body	**props** *required*	props	< string, object > map

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

- `/`

4.2.35. 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

- `/`

4.2.36. 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	< CustomConfigEntry > array
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

- `/`

4.2.37. Get topic dyn configs

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

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

- `/`

4.2.38. 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
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.39. 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. Schema-registry-controller

Schema Registry Controller

4.3.1. Get schema by id

```
GET /schemaregistry/schemas/ids/{schemaId}
```

Parameters

Type	Name	Description	Schema
Path	**schemaId** *required*	schemaId	integer(int32)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.2. List all subjects

```
GET /schemaregistry/subjects
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.3. Check if a schema has already been registered under the specified subject

```
POST /schemaregistry/subjects/{subject}
```

Parameters

Type	Name	Description	Schema
Path	**subject** *required*	subject	string
Query	**schemaStr** *required*	schemaStr	string

Responses

HTTP Code	Description	Schema
200	OK	SchemaRegistryMetadata

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

- `/`

4.3.4. Get latest schema by subject

```
GET /schemaregistry/subjects/{subject}
```

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.5. Delete the specified subject and its associated compatibility level if registered.

```
DELETE /schemaregistry/subjects/{subject}
```

Parameters

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

Responses

HTTP Code	Description	Schema
200	OK	< integer(int32) > array
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

Consumes

- `application/json`

Produces

- `/`

4.3.6. Register schema by subject

```
POST /schemaregistry/subjects/{subject}/versions
```

Parameters

Type	Name	Description	Schema
Path	**subject** *required*	subject	string
Query	**schemaStr** *required*	schemaStr	string

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.7. Get all versions for the specified subject

```
GET /schemaregistry/subjects/{subject}/versions
```

Parameters

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

Responses

HTTP Code	Description	Schema
200	OK	< integer(int32) > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.3.8. Get schema by subject and version

```
GET /schemaregistry/subjects/{subject}/versions/{versionId}
```

Parameters

Type	Name	Description	Schema
Path	**subject** *required*	subject	string
Path	**versionId** *required*	versionId	integer(int32)

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.4. User-controller

Security User Management Controller.

4.4.1. Add user.

```
POST /users
```

Parameters

Type	Name	Description	Schema
Body	**user** *required*	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

- `/`

4.4.2. Get user list.

```
GET /users
```

Responses

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

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.4.3. Modify user information.

```
PUT /users
```

Parameters

Type	Name	Description	Schema
Body	**user** *required*	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

- `/`

4.4.4. Delete user.

```
DELETE /users/{username}
```

Parameters

Type	Name	Description	Schema
Path	**username** *required*	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

- `/`

4.5. Zookeeper-controller

Zookeeper Controller

4.5.1. Get the connection state of zookeeper

```
GET /zk/connstate
```

Responses

HTTP Code	Description	Schema
200	OK	string
401	Unauthorized	No Content

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

Consumes

- `application/json`

Produces

- `/`

4.5.2. 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

- `/`

4.5.3. Get data of a zookeeper path

```
GET /zk/get/path
```

Parameters

Type	Name	Description	Schema
Query	**path** *required*	path	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

- `/`

4.5.4. List a zookeeper path

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

Parameters

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

Responses

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

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.5.5. 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 *optional*	integer(int32)
replicaAssignment *optional*	< < integer(int32) > array > array
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. ClusterInfo

Name	Schema
clusterId *optional*	string
controller *optional*	Node
nodes *optional*	< Node > array

5.4. ConsumerGroupDesc

Name	Schema
assignmentStrategy *optional*	string
clientId *optional*	string
consumerId *optional*	string
coordinator *optional*	Node
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 (Unknown, PreparingRebalance, CompletingRebalance, Stable, Dead, Empty)
topic *optional*	string
type *optional*	enum (NEW, OLD)

5.5. ConsumerGroupMeta

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

5.6. CustomConfigEntry

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

5.7. CustomTopicPartitionInfo

Name	Schema
endOffset *optional*	integer(int64)
in_sync *optional*	boolean
messageAvailable *optional*	integer(int64)
startOffset *optional*	integer(int64)
topicPartitionInfo *optional*	TopicPartitionInfo

5.8. GeneralResponse

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

5.9. HashMap«string,object»

Type : < string, object > map

5.10. HealthCheckResult

Name	Description	Schema
msg *optional*		string
status *optional*		string
timestamp *optional*	**Example** : `"yyyy-MM-dd HH:mm:ss"`	string

5.11. HostAndPort

Name	Schema
hasBracketlessColons *optional*	boolean
host *optional*	string
hostText *optional*	string
port *optional*	integer(int32)

5.12. JMXConfiguration

Name	Schema
exclude *optional*	JMXFilter
include *optional*	JMXFilter

5.13. JMXFilter

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

5.14. JMXMetricData

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

5.15. JMXMetricDataV1

Name	Description	Schema
collected *optional*		boolean
host *optional*		string
mbeanInfo *optional*		object
msg *optional*		string
timestamp *optional*	**Example** : `"yyyy-MM-dd HH:mm:ss"`	string

5.16. JMXQuery

Name	Schema
filters *optional*	< JMXConfiguration > array

5.17. LogDirInfo

Name	Schema
error *optional*	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 *optional*	< 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 *optional*	< TopicPartition > array
clientId *optional*	string
host *optional*	string
memberId *optional*	string

5.21. Node

Name	Schema
empty *optional*	boolean
hash *optional*	integer(int32)
host *optional*	string
id *optional*	integer(int32)
idString *optional*	string

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

5.22. Pattern

Name	Schema
cursor *optional*	integer(int32)
flags *optional*	integer(int32)
pattern *optional*	string

5.23. ReassignModel

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

5.24. ReassignStatus

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

5.25. ReassignWrapper

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

5.26. Record

Name	Schema
key *optional*	string
keyDecoder *optional*	string
offset *optional*	integer(int64)
timestamp *optional*	integer(int64)
topic *optional*	string
value *optional*	string
valueDecoder *optional*	string

5.27. ReplicaInfo

Name	Schema
isFuture *optional*	boolean
offsetLag *optional*	integer(int64)
size *optional*	integer(int64)

5.28. ReplicaLogDirInfo

Name	Schema
currentReplicaLogDir *optional*	string
currentReplicaOffsetLag *optional*	integer(int64)
futureReplicaLogDir *optional*	string
futureReplicaOffsetLag *optional*	integer(int64)

5.29. SchemaMetadata

Name	Schema
id *optional*	integer(int32)
schema *optional*	string
version *optional*	integer(int32)

5.30. SchemaRegistryMetadata

Name	Schema
id *optional*	integer(int32)
schema *optional*	string
subject *optional*	string
version *optional*	integer(int32)

5.31. TopicBrief

Name	Schema
isrRate *optional*	number(double)
numPartition *optional*	integer(int32)
replicationFactor *optional*	integer(int32)
topic *optional*	string

5.32. TopicDetail

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

5.33. TopicMeta

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

5.34. TopicPartition

Name	Schema
hash *optional*	integer(int32)
partition *optional*	integer(int32)
topic *optional*	string

5.35. TopicPartitionInfo

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

5.36. TopicPartitionReplicaAssignment

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

5.37. User

Name	Schema
password *optional*	string
role *optional*	string
username *optional*	string

5.38. 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)

5.39. ZkServerEnvironment

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

5.40. ZkServerStat

Name	Schema
avgLatency *optional*	integer(int32)
buildDate *optional*	string

Name	Schema
clients *optional*	< ZkServerClient > array
connections *optional*	integer(int32)
maxLatency *optional*	integer(int32)
minLatency *optional*	integer(int32)
mode *optional*	enum (Leader, Follower, Observer, Standalone, Down, Unknow)
msg *optional*	string
nodes *optional*	integer(int32)
outstanding *optional*	integer(int32)
received *optional*	integer(int32)
sent *optional*	integer(int32)
version *optional*	string
zxId *optional*	string