

# 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 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 .....	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* : [gnuahpc@gmail.com](mailto: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
- 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	<b>jmxurl</b> <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	< <a href="#">JMXMetricDataV1</a> > 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	<b>jmxurl</b> <i>optional</i>	Parameter jmxurl should be a comma-separated list of {IP:Port} or set to 'default'	string
Body	<b>jmxQuery</b> <i>required</i>	jmxQuery	<a href="#">JMXQuery</a>

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">JMXMetricData</a> > 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	<b>filterKey</b> <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	< <a href="#">BrokerInfo</a> > 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	<b>brokerList</b> <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

POST /kafka/brokers/logdirs/detail

Parameters

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

Type	Name	Description	Schema
Body	<b>topicPartitionMap</b> <i>optional</i>	topicPartitionMap	< string, < integer(int32) > array > map

## Responses

HTTP Code	Description	Schema
200	OK	< string, < string, <a href="#">LogDirInfo</a> > 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	<b>brokerId</b> <i>required</i>	brokerId	integer(int32)
Path	<b>partition</b> <i>required</i>	partition	integer(int32)
Path	<b>topic</b> <i>required</i>	topic	string

## Responses

HTTP Code	Description	Schema
200	OK	<a href="#">ReplicaLogDirInfo</a>
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	<b>brokerId</b> <i>required</i>	brokerId	integer(int32)

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">CustomConfigEntry</a> > 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	<b>brokerId</b> <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	<b>brokerId</b> <i>required</i>	brokerId	integer(int32)

Type	Name	Description	Schema
Body	<b>props</b> <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	<b>brokerId</b> <i>required</i>	brokerId	integer(int32)
Query	<b>configKeysToBeRemoved</b> <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	<a href="#">ClusterInfo</a>
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	<b>offset</b> <i>optional</i>	[long/yyyy-MM-dd HH:mm:ss.SSS] can be supported.	string	
Path	<b>partition</b> <i>required</i>	partition	integer(int32)	
Path	<b>topic</b> <i>required</i>	topic	string	
Query	<b>avroSchema</b> <i>optional</i>	avroSchema	string	
Query	<b>fetchTimeoutMs</b> <i>optional</i>	fetchTimeoutMs	integer(int64)	"30000"
Query	<b>keyDecoder</b> <i>optional</i>	keyDecoder	string	"StringDeserializer"
Query	<b>maxRecords</b> <i>optional</i>	maxRecords	integer(int32)	"10"
Query	<b>valueDecoder</b> <i>optional</i>	valueDecoder	string	"StringDeserializer"

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">Record</a> > 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	<b>consumergroup</b> <i>required</i>	consumergroup	string
Path	<b>type</b> <i>required</i>	type	enum (NEW, OLD)

#### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">GeneralResponse</a>
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	<b>consumergroup</b> <i>required</i>	consumergroup	string

Type	Name	Description	Schema
Path	<b>topic</b> <i>required</i>	topic	string
Path	<b>type</b> <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	<b>consumergroup</b> <i>required</i>	consumergroup	string
Path	<b>offset</b> <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

Type	Name	Description	Schema
Path	<b>partition</b> <i>required</i>	partition	integer(int32)
Path	<b>topic</b> <i>required</i>	topic	string
Path	<b>type</b> <i>required</i>	type	enum (NEW, OLD)

## Responses

HTTP Code	Description	Schema
200	OK	<a href="#">GeneralResponse</a>
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	<b>topic</b> <i>optional</i>	topic	string
Query	<b>type</b> <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 all the meta data of new consumer groups, including state, coordinator, assignmentStrategy, members

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

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">ConsumerGroupMeta</a> > 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	<a href="#">ConsumerGroupMeta</a>
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	<b>consumerGroup</b> <i>required</i>	consumerGroup	string
Path	<b>type</b> <i>required</i>	type	enum (NEW, OLD)

## Responses

HTTP Code	Description	Schema
200	OK	< string, < <a href="#">ConsumerGroupDescription</a> > 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	<b>consumerGroup</b> <i>required</i>	consumerGroup	string
Path	<b>type</b> <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	<b>topic</b> <i>required</i>	topic	string
Path	<b>type</b> <i>required</i>	type	enum (NEW, OLD)
Query	<b>consumerGroup</b> <i>optional</i>	consumerGroup	string

#### Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">ConsumerGroupDesc</a> > 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	<a href="#">Node</a>
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	<a href="#">HealthCheckResult</a>
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	<b>addPartitions</b> <i>required</i>	addPartitions	< <a href="#">AddPartition</a> > array

### Responses

HTTP Code	Description	Schema
200	OK	< string, <a href="#">GeneralResponse</a> > 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	<b>partitionList</b> <i>required</i>	partitionList	< <a href="#">TopicPartition</a> > 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	<b>reassign</b> <i>required</i>	reassign	<a href="#">ReassignModel</a>

## Responses

HTTP Code	Description	Schema
-1	Reassignment Failed	No Content
0	Reassignment In Progress	No Content
1	Reassignment Completed	No Content
200	OK	<a href="#">ReassignStatus</a>
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	<b>interBrokerThrottle</b> <i>optional</i>	interBrokerThrottle	integer(int64)	<code>"-1"</code>
Query	<b>replicaAlterLogDirsThrottle</b> <i>optional</i>	replicaAlterLogDirsThrottle	integer(int64)	<code>"-1"</code>
Query	<b>timeoutMs</b> <i>optional</i>	timeoutMs	integer(int64)	<code>"10000"</code>
Body	<b>reassign</b> <i>required</i>	reassign	<a href="#">ReassignModel</a>	

## Responses

HTTP Code	Description	Schema
200	OK	<a href="#">ReassignStatus</a>
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	<b>reassignWrapper</b> <i>required</i>	reassignWrapper	<a href="#">ReassignWrapper</a>

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">ReassignModel</a> > 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	<a href="#">GeneralResponse</a>
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	<b>topicList</b> <i>required</i>	topicList	< string > array(multi)

#### Responses

HTTP Code	Description	Schema
200	OK	< string, <a href="#">GeneralResponse</a> > 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	<b>topicList</b> <i>required</i>	topicList	< <a href="#">TopicDetail</a> > array

### Responses

HTTP Code	Description	Schema
201	Created	< string, <a href="#">GeneralResponse</a> > 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	<b>topicList</b> <i>required</i>	topicList	< <a href="#">TopicDetail</a> > 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	<b>topic</b> <i>required</i>	topic	string

## Responses

HTTP Code	Description	Schema
200	OK	<a href="#">TopicMeta</a>
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	<b>topic</b> <i>required</i>	topic	string

#### Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">CustomConfigEntry</a> > 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	<b>topic</b> <i>required</i>	topic	string

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

## Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">CustomConfigEntry</a> > 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	<b>key</b> <i>required</i>	key	string
Path	<b>topic</b> <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

- `/`

### 4.2.36. Update a topic config by key

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

#### Parameters

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

#### Responses

HTTP Code	Description	Schema
200	OK	< <a href="#">CustomConfigEntry</a> > 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	<b>topic</b> <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

- `/`

### 4.2.38. Tell if a topic exists

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

### Parameters

Type	Name	Description	Schema
Path	<b>topic</b> <i>required</i>	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	< <a href="#">TopicBrief</a> > 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	<b>schemaId</b> <i>required</i>	schemaId	integer(int32)

**Responses**

HTTP Code	Description	Schema
200	OK	<a href="#">SchemaRegistryMetadata</a>
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	< <a href="#">SchemaRegistryMetadata</a> > 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	<b>subject</b> <i>required</i>	subject	string
Query	<b>schemaStr</b> <i>required</i>	schemaStr	string

## Responses

HTTP Code	Description	Schema
200	OK	<a href="#">SchemaRegistryMetadata</a>

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	<b>subject</b> <i>required</i>	subject	string

#### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">SchemaMetadata</a>
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	<b>subject</b> <i>required</i>	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	<b>subject</b> <i>required</i>	subject	string
Query	<b>schemaStr</b> <i>required</i>	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	<b>subject</b> <i>required</i>	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	<b>subject</b> <i>required</i>	subject	string
Path	<b>versionId</b> <i>required</i>	versionId	integer(int32)

#### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">SchemaMetadata</a>
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	<b>user</b> <i>required</i>	user	<a href="#">User</a>

#### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">GeneralResponse</a>
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	<b>user</b> <i>required</i>	user	<a href="#">User</a>

#### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">GeneralResponse</a>
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	<b>username</b> <i>required</i>	username	string

### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">GeneralResponse</a>
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, <a href="#">ZkServerEnvironment</a> > 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	<b>path</b> <i>required</i>	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	<b>path</b> <i>required</i>	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, <a href="#">ZkServerStat</a> > 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
<b>numPartitionsAdded</b> <i>optional</i>	integer(int32)
<b>replicaAssignment</b> <i>optional</i>	< < integer(int32) > array > array
<b>topic</b> <i>optional</i>	string

## 5.2. BrokerInfo

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

## 5.3. ClusterInfo

Name	Schema
<b>clusterId</b> <i>optional</i>	string
<b>controller</b> <i>optional</i>	<a href="#">Node</a>
<b>nodes</b> <i>optional</i>	< <a href="#">Node</a> > array

## 5.4. ConsumerGroupDesc

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

## 5.5. ConsumerGroupMeta

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

## 5.6. CustomConfigEntry

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

## 5.7. CustomTopicPartitionInfo

Name	Schema
<b>endOffset</b> <i>optional</i>	integer(int64)
<b>in_sync</b> <i>optional</i>	boolean
<b>messageAvailable</b> <i>optional</i>	integer(int64)
<b>startOffset</b> <i>optional</i>	integer(int64)
<b>topicPartitionInfo</b> <i>optional</i>	<a href="#">TopicPartitionInfo</a>

## 5.8. GeneralResponse

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

## 5.9. HashMap«string,object»

Type : < string, object > map

## 5.10. HealthCheckResult

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

## 5.11. HostAndPort

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

## 5.12. JMXConfiguration

Name	Schema
<b>exclude</b> <i>optional</i>	<a href="#">JMXFilter</a>
<b>include</b> <i>optional</i>	<a href="#">JMXFilter</a>

## 5.13. JMXFilter

Name	Schema
<b>attribute</b> <i>optional</i>	object
<b>beanNames</b> <i>optional</i>	< string > array
<b>beanRegexes</b> <i>optional</i>	< <a href="#">Pattern</a> > array
<b>domain</b> <i>optional</i>	string
<b>domainRegex</b> <i>optional</i>	<a href="#">Pattern</a>
<b>emptyBeanName</b> <i>optional</i>	boolean
<b>filter</b> <i>optional</i>	< string, object > map



## 5.14. JMXMetricData

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

## 5.15. JMXMetricDataV1

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

## 5.16. JMXQuery

Name	Schema
<b>filters</b> <i>optional</i>	< <a href="#">JMXConfiguration</a> > array

## 5.17. LogDirInfo



Name	Schema
<b>error</b> <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
<b>replicaInfos</b> <i>optional</i>	< string, <a href="#">ReplicaInfo</a> > 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
<b>assignment</b> <i>optional</i>	< <a href="#">TopicPartition</a> > array
<b>clientId</b> <i>optional</i>	string
<b>host</b> <i>optional</i>	string
<b>memberId</b> <i>optional</i>	string

## 5.21. Node

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

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

## 5.22. Pattern

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

## 5.23. ReassignModel

Name	Schema
<b>partitions</b> <i>optional</i>	< <a href="#">TopicPartitionReplicaAssignment</a> > array
<b>version</b> <i>optional</i>	integer(int32)

## 5.24. ReassignStatus

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

## 5.25. ReassignWrapper

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

## 5.26. Record

Name	Schema
<b>key</b> <i>optional</i>	string
<b>keyDecoder</b> <i>optional</i>	string
<b>offset</b> <i>optional</i>	integer(int64)
<b>timestamp</b> <i>optional</i>	integer(int64)
<b>topic</b> <i>optional</i>	string
<b>value</b> <i>optional</i>	string
<b>valueDecoder</b> <i>optional</i>	string

## 5.27. ReplicaInfo

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

## 5.28. ReplicaLogDirInfo

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

## 5.29. SchemaMetadata

Name	Schema
<b>id</b> <i>optional</i>	integer(int32)
<b>schema</b> <i>optional</i>	string
<b>version</b> <i>optional</i>	integer(int32)

## 5.30. SchemaRegistryMetadata

Name	Schema
<b>id</b> <i>optional</i>	integer(int32)
<b>schema</b> <i>optional</i>	string
<b>subject</b> <i>optional</i>	string
<b>version</b> <i>optional</i>	integer(int32)

## 5.31. TopicBrief

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

## 5.32. TopicDetail

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

## 5.33. TopicMeta

Name	Schema
<b>internal</b> <i>optional</i>	boolean
<b>partitionCount</b> <i>optional</i>	integer(int32)
<b>replicationFactor</b> <i>optional</i>	integer(int32)
<b>topicName</b> <i>optional</i>	string
<b>topicPartitionInfos</b> <i>optional</i>	< <a href="#">CustomTopicPartitionInfo</a> > array



## 5.34. TopicPartition

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

## 5.35. TopicPartitionInfo

Name	Schema
<b>isr</b> <i>optional</i>	< <a href="#">Node</a> > array
<b>leader</b> <i>optional</i>	<a href="#">Node</a>
<b>partition</b> <i>optional</i>	integer(int32)
<b>replicas</b> <i>optional</i>	< <a href="#">Node</a> > array

## 5.36. TopicPartitionReplicaAssignment

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

## 5.37. User

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

## 5.38. ZkServerClient

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

## 5.39. ZkServerEnvironment

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

## 5.40. ZkServerStat

Name	Schema
<b>avgLatency</b> <i>optional</i>	integer(int32)
<b>buildDate</b> <i>optional</i>	string

Name	Schema
<b>clients</b> <i>optional</i>	< <a href="#">ZkServerClient</a> > array
<b>connections</b> <i>optional</i>	integer(int32)
<b>maxLatency</b> <i>optional</i>	integer(int32)
<b>minLatency</b> <i>optional</i>	integer(int32)
<b>mode</b> <i>optional</i>	enum (Leader, Follower, Observer, Standalone, Down, Unknow)
<b>msg</b> <i>optional</i>	string
<b>nodes</b> <i>optional</i>	integer(int32)
<b>outstanding</b> <i>optional</i>	integer(int32)
<b>received</b> <i>optional</i>	integer(int32)
<b>sent</b> <i>optional</i>	integer(int32)
<b>version</b> <i>optional</i>	string
<b>zxId</b> <i>optional</i>	string