

# 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. Get the message from the offset of the partition in the topic, decoder is not supported yet ..	7
4.2.3. Delete old Consumer Group .....	7
4.2.4. getLastCommitTimestamp .....	8
4.2.5. Reset consumer group offset, earliest/latest can be used .....	9
4.2.6. List all consumer groups from zk and kafka .....	10
4.2.7. Describe consumer groups, showing lag and offset, may be slow if multi topic are listened	11
4.2.8. Get the topics involved of the specify consumer group .....	12
4.2.9. Describe consumer groups by topic, showing lag and offset .....	12
4.2.10. Check the cluster health.....	13
4.2.11. Add a partition to the topic .....	14
4.2.12. Check the partition reassignment process .....	14
4.2.13. Execute the partition reassignment.....	15
4.2.14. Generate plan for the partition reassignment.....	16
4.2.15. List topics .....	17
4.2.16. Create a topic.....	17
4.2.17. Describe a topic by fetching the metadata and config .....	18
4.2.18. Delete a topic (you should enable topic deletion .....	19
4.2.19. Create topic configs .....	19
4.2.20. Get topic configs .....	20

4.2.21. Update topic configs . . . . .	21
4.2.22. Delete topic configs . . . . .	22
4.2.23. Get topic config by key . . . . .	22
4.2.24. Delete a topic config by key . . . . .	23
4.2.25. Create a topic config by key . . . . .	24
4.2.26. Update a topic config by key . . . . .	25
4.2.27. Tell if a topic exists . . . . .	26
4.2.28. Write a message to the topic, for testing purpose . . . . .	26
4.2.29. List topics Brief . . . . .	27
4.3. User-controller . . . . .	27
4.3.1. Add user. . . . .	28
4.3.2. Get user list. . . . .	28
4.3.3. Modify user information. . . . .	29
4.3.4. Delete user. . . . .	29
4.4. Zookeeper-controller . . . . .	30
4.4.1. Get the connection state of zookeeper . . . . .	30
4.4.2. Get the environment information of zookeeper . . . . .	31
4.4.3. List a zookeeper path. . . . .	31
4.4.4. Get the service state of zookeeper . . . . .	32
5. Definitions . . . . .	34
5.1. AddPartition . . . . .	34
5.2. BrokerInfo . . . . .	34
5.3. ConsumerGroupDesc. . . . .	34
5.4. GeneralResponse . . . . .	35
5.5. HashMap«string,object» . . . . .	35
5.6. HealthCheckResult . . . . .	35
5.7. HostAndPort . . . . .	36
5.8. JMXConfiguration . . . . .	36
5.9. JMXFilter. . . . .	36
5.10. JMXMetricData . . . . .	37
5.11. JMXMetricDataV1. . . . .	37
5.12. JMXQuery . . . . .	37
5.13. Map«int,long» . . . . .	38
5.14. Pattern . . . . .	38
5.15. ReassignWrapper . . . . .	38
5.16. TopicAndPartition . . . . .	38
5.17. TopicBrief. . . . .	38
5.18. TopicDetail. . . . .	39

5.19. TopicMeta.....	39
5.20. TopicPartitionInfo .....	39
5.21. User .....	40
5.22. ZkServerClient .....	40
5.23. ZkServerEnvironment .....	41
5.24. ZkServerStat .....	41

# Chapter 1. Overview

Kafka REST API SwaggerUI

## 1.1. Version information

*Version* : 0.1.0

## 1.2. Contact information

*Contact* : gnuhpc

*Contact Email* : [gnuuhpc@gmail.com](mailto:gnuuhpc@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	<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. 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	<b>offset</b> <i>required</i>	offset	integer(int64)
Path	<b>partition</b> <i>required</i>	partition	integer(int32)
Path	<b>topic</b> <i>required</i>	topic	string
Query	<b>decoder</b> <i>optional</i>	decoder	string

#### Responses

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

#### Consumes

- `application/json`

#### Produces

- `/`

### 4.2.3. Delete old Consumer Group

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

## Parameters

Type	Name	Description	Schema
Path	<b>consumergroup</b> <i>required</i>	consumergroup	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

- `/`

### 4.2.4. getLastCommitTimestamp

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

## Parameters

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

- `/`

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

```
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>required</i>	offset	string
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

- `/`

### 4.2.6. 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

- `/`

## 4.2.7. Describe consumer groups, showing lag and offset, may be slow if multi topic 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

- `/`

## 4.2.8. 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

- `/`

## 4.2.9. Describe consumer groups by topic, showing lag and offset

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

### 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	< <a href="#">ConsumerGroupDesc</a> > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

## Consumes

- `application/json`

## Produces

- `/`

## 4.2.10. 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

- `/`

## 4.2.11. Add a partition to the topic

POST /kafka/partitions/add

### Parameters

Type	Name	Description	Schema
Body	<b>addPartition</b> <i>required</i>	addPartition	<a href="#">AddPartition</a>

### Responses

HTTP Code	Description	Schema
200	OK	<a href="#">TopicMeta</a>
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

### Consumes

- `application/json`

### Produces

- `/`

## 4.2.12. Check the partition reassignment process

PUT /kafka/partitions/reassign/check

## Parameters

Type	Name	Description	Schema
Body	<b>reassignStr</b> <i>required</i>	reassignStr	string

## Responses

HTTP Code	Description	Schema
-1	Reassignment Failed	No Content
0	Reassignment In Progress	No Content
1	Reassignment Completed	No Content
200	OK	< string, integer(int32) > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

## Consumes

- `application/json`

## Produces

- `/`

### 4.2.13. Execute the partition reassignment

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

## Parameters

Type	Name	Description	Schema
Body	<b>reassignStr</b> <i>required</i>	reassignStr	string

## Responses

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

## Consumes

- `application/json`

## Produces

- `/`

### 4.2.14. Generate plan for the partition reassignment

POST /kafka/partitions/reassign/generate

## Parameters

Type	Name	Description	Schema
Body	<b>reassignWrapper</b> <i>per required</i>	reassignWrapper	<a href="#">ReassignWrapper</a>

## Responses

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

### Consumes

- `application/json`

### Produces

- `/`

## 4.2.15. 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

- `/`

## 4.2.16. Create a topic

POST /kafka/topics/create

### Parameters

Type	Name	Description	Schema
Query	<b>reassignStr</b> <i>optional</i>	reassignStr	string
Body	<b>topic</b> <i>required</i>	topic	<a href="#">TopicDetail</a>

## Responses

HTTP Code	Description	Schema
201	Created	<a href="#">TopicMeta</a>
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

## Consumes

- `application/json`

## Produces

- `/`

### 4.2.17. 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

- /

### 4.2.18. Delete a topic (you should enable topic deletion)

```
DELETE /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="#">GeneralResponse</a>
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

## Consumes

- `application/json`

## Produces

- /

### 4.2.19. Create topic configs

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

#### Parameters

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

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

## Responses

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

## Consumes

- `application/json`

## Produces

- `/`

## 4.2.20. 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	< string, object > map
401	Unauthorized	No Content



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

#### Consumes

- `application/json`

#### Produces

- `/`

### 4.2.21. Update topic configs

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

#### Parameters

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

#### Responses

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

#### Consumes

- `application/json`

## Produces

- /

### 4.2.22. Delete topic configs

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

#### Parameters

Type	Name	Description	Schema
Path	<b>topic</b> <i>required</i>	topic	string
Body	<b>delProps</b> <i>required</i>	delProps	< string > array

#### Responses

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

## Consumes

- `application/json`

## Produces

- /

### 4.2.23. 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

- `/`

### 4.2.24. Delete a topic config by key

```
DELETE /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	boolean
204	No Content	No Content
401	Unauthorized	No Content
403	Forbidden	No Content

#### Consumes

- `application/json`

#### Produces

- `/`

### 4.2.25. Create a topic config by key

```
POST /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	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

### Consumes

- `application/json`

### Produces

- `/`

## 4.2.26. 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	< string, object > map
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

### Consumes

- `application/json`

### Produces

- `/`

## 4.2.27. 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

- `/`

## 4.2.28. Write a message to the topic, for testing purpose

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

### Parameters

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

## Responses

HTTP Code	Description	Schema
201	Created	<a href="#">GeneralResponse</a>
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

## Consumes

- `text/plain`

## Produces

- `/`

### 4.2.29. 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. User-controller

Security User Management Controller.

### 4.3.1. 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

- `/`

### 4.3.2. 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

- `/`

### 4.3.3. 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

- `/`

### 4.3.4. 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

- `/`

## 4.4. 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

HTTP Code	Description	Schema
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, <a href="#">ZkServerEnvironment</a> > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

#### Consumes

- `application/json`

#### Produces

- `/`

### 4.4.3. List a zookeeper path

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

## Parameters

Type	Name	Description	Schema
Path	<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
404	Not Found	No Content

## Consumes

- `application/json`

## Produces

- `/`

### 4.4.4. 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>	string
<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. ConsumerGroupDesc

Name	Schema
<b>consumerId</b> <i>optional</i>	string
<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 (RUNNING, PENDING)
<b>topic</b> <i>optional</i>	string
<b>type</b> <i>optional</i>	enum (NEW, OLD)

## 5.4. GeneralResponse

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

## 5.5. HashMap«string,object»

Type : < string, object > map

## 5.6. 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

## 5.7. HostAndPort

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

## 5.8. JMXConfiguration

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

## 5.9. 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>



Name	Schema
<b>emptyBeanName</b> <i>optional</i>	boolean
<b>filter</b> <i>optional</i>	< string, object > map

## 5.10. 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>	<b>Example :</b> "yyyy-MM-dd HH:mm:ss"	string

## 5.11. 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>	<b>Example :</b> "yyyy-MM-dd HH:mm:ss"	string

## 5.12. JMXQuery

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

## 5.13. Map«int,long»

Type : < string, integer(int64) > map

## 5.14. Pattern

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

## 5.15. ReassignWrapper

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

## 5.16. TopicAndPartition

Type : object

## 5.17. TopicBrief

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

## 5.18. 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

## 5.19. TopicMeta

Name	Schema
<b>partitionCount</b> <i>optional</i>	integer(int32)
<b>replicationFactor</b> <i>optional</i>	integer(int32)
<b>topicCustomConfigs</b> <i>optional</i>	< string, object > map
<b>topicName</b> <i>optional</i>	string
<b>topicPartitionInfos</b> <i>optional</i>	< <a href="#">TopicPartitionInfo</a> > array

## 5.20. TopicPartitionInfo

Name	Schema
<b>endOffset</b> <i>optional</i>	integer(int64)
<b>in_sync</b> <i>optional</i>	boolean
<b>isr</b> <i>optional</i>	< string > array
<b>leader</b> <i>optional</i>	string

Name	Schema
<b>messageAvailable</b> <i>optional</i>	integer(int64)
<b>partitionId</b> <i>optional</i>	integer(int32)
<b>replicas</b> <i>optional</i>	< string > array
<b>startOffset</b> <i>optional</i>	integer(int64)

## 5.21. User

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

## 5.22. 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)

## 5.23. ZkServerEnvironment

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

## 5.24. ZkServerStat

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