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	4
	API /jmx/v2/filters.	
	4.1.3. List the query filter templates with the filterKey. If filterKey is set to empty, it will return al	l 5
	the templates.	
	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 tonic configs	20

4.2.22. Delete topic configs       2         4.2.23. Get topic config by key       2         4.2.24. Delete a topic config by key       2         4.2.25. Create a topic config by key       2         4.2.26. Update a topic config by key       2         4.2.27. Tell if a topic exists       2         4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.4. Zookeeper-controller       3         4.4. Zookeeper-controller       3         4.4. Zookeeper-controller       3         4.4. Zookeeper topic       3         4.4. Zookeeper topic       3         4.4. Zookeeper topic       3         4.4. List a zookeeper path       3         4.4. List a zookeeper path       3         4.4. S. Get the service state of zookeeper       3         5. Definitions       3         5. RorkerInfo       3         5. GeneralResponse       3         5. HealthCheckResult       3         5. HealthCheckResult       3 <t< th=""><th>4.2.21. Update topic configs</th><th> 21</th></t<>	4.2.21. Update topic configs	21
4.2.24. Delete a topic config by key       2         4.2.25. Create a topic config by key       2         4.2.26. Update a topic config by key       2         4.2.27. Tell if a topic exists       2         4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3. User-controller       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the connection state of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap-string,object>       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration<	4.2.22. Delete topic configs	22
4.2.25. Create a topic config by key       2         4.2.26. Update a topic config by key       2         4.2.27. Tell if a topic exists       2         4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3. User-controller       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap-string, object>       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricDataV1       3	4.2.23. Get topic config by key	22
4.2.26. Update a topic config by key       2         4.2.27. Tell if a topic exists       2         4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3	4.2.24. Delete a topic config by key	23
4.2.27. Tell if a topic exists       2         4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3. User-controller       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information.       2         4.3.4. Delete user.       2         4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMapestring,object*       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Mape	4.2.25. Create a topic config by key	24
4.2.28. Write a message to the topic, for testing purpose       2         4.2.29. List topics Brief       2         4.3. User-controller       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.2. Zookeeper-controller       3         4.4. Zookeeper-controller       3         4.4. Get the connection state of zookeeper       3         4.4. Get data of a zookeeper path       3         4.4. List a zookeeper path       3         4.4. List a zookeeper path       3         4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXPilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map«int,long»       3	4.2.26. Update a topic config by key	25
4.2.29. List topics Brief       2         4.3. User-controller       2         4.3.1. Add user       2         4.3.2. Get user list       2         4.3.3. Modify user information       2         4.3.4. Delete user       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap*string,object*       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map*int,long*       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.2.27. Tell if a topic exists	26
4.3. User-controller       2         4.3.1. Add user.       2         4.3.2. Get user list       2         4.3.3. Modify user information.       2         4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.2.28. Write a message to the topic, for testing purpose	26
4.3.1. Add user.       2         4.3.2. Get user list.       2         4.3.3. Modify user information.       2         4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap*string.object*       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map*int,long*       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.2.29. List topics Brief	27
4.3.2. Get user list.       2         4.3.3. Modify user information.       2         4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap*string,object*       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map*int,long*       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.3. User-controller	27
4.3.3. Modify user information.       2         4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.5. HashMap«string,object»       3         5.5. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.3.1. Add user	28
4.3.4. Delete user.       2         4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.15. ReassignWrapper       3	4.3.2. Get user list	28
4.4. Zookeeper-controller       3         4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.3.3. Modify user information	29
4.4.1. Get the connection state of zookeeper       3         4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap*string,object*       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.12. JMXQuery       3         5.13. Map*int,long*       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.3.4. Delete user	29
4.4.2. Get the environment information of zookeeper       3         4.4.3. Get data of a zookeeper path       3         4.4.4. List a zookeeper path       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4. Zookeeper-controller	30
4.4.3. Get data of a zookeeper path.       3         4.4.4. List a zookeeper path.       3         4.4.5. Get the service state of zookeeper       3         5. Definitions.       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4.1. Get the connection state of zookeeper	30
4.4.4. List a zookeeper path.       3         4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4.2. Get the environment information of zookeeper	31
4.4.5. Get the service state of zookeeper       3         5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4.3. Get data of a zookeeper path	31
5. Definitions       3         5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4.4. List a zookeeper path	32
5.1. AddPartition       3         5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	4.4.5. Get the service state of zookeeper	33
5.2. BrokerInfo       3         5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5. Definitions	34
5.3. ConsumerGroupDesc       3         5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.1. AddPartition	34
5.4. GeneralResponse       3         5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.2. BrokerInfo	34
5.5. HashMap«string,object»       3         5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.3. ConsumerGroupDesc	34
5.6. HealthCheckResult       3         5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.4. GeneralResponse	35
5.7. HostAndPort       3         5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.5. HashMap«string,object»	35
5.8. JMXConfiguration       3         5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.6. HealthCheckResult	35
5.9. JMXFilter       3         5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.7. HostAndPort	36
5.10. JMXMetricData       3         5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.8. JMXConfiguration	36
5.11. JMXMetricDataV1       3         5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.9. JMXFilter	36
5.12. JMXQuery       3         5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.10. JMXMetricData	37
5.13. Map«int,long»       3         5.14. Pattern       3         5.15. ReassignWrapper       3	5.11. JMXMetricDataV1	37
5.14. Pattern       3         5.15. ReassignWrapper       3	5.12. JMXQuery	37
5.15. ReassignWrapper	5.13. Map«int,long»	38
	5.14. Pattern	38
E 16 Tanic And Dartition	5.15. ReassignWrapper	38
5.16. Topicandianition	5.16. TopicAndPartition	38
5.17. TopicBrief	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: 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

• 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**

Туре	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**

Туре	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**

Туре	Name	Description	Schema
Query	<b>filterKey</b> required	filterKey	string

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

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. Get the message from the offset of the partition in the topic, decoder is not supported yet

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

#### **Parameters**

Туре	Name	Description	Schema
Path	<b>offset</b> required	offset	integer(int64)
Path	<b>partition</b> required	partition	integer(int32)
Path	topic required	topic	string
Query	<b>decoder</b> optional	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	consumergro up required	consumergroup	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**

• /

### ${\bf 4.2.4.}~ get Last Commit Time stamp$

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

#### **Parameters**

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

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

• 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	consumergro up required	consumergroup	string
Path	<b>offset</b> required	offset	string
Path	<b>partition</b> required	partition	integer(int32)
Path	topic required	topic	string
Path	<b>type</b> required	type	enum (NEW, OLD)

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

application/json

#### **Produces**

• /

## 4.2.6. List all consumer groups from zk and kafka

GET /kafka/consumergroups

#### **Parameters**

Туре	Name	Description	Schema
Query	topic optional	topic	string
Query	type optional	type	enum (NEW, OLD)

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

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**

Туре	Name	Description	Schema
Path	consumerGro up required	consumerGroup	string
Path	type required	type	enum (NEW, OLD)

#### Responses

HTTP Code	Description	Schema
200	OK	< string, < ConsumerGroupDe sc > 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**

Туре	Name	Description	Schema
Path	consumerGro up 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**

• /

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

 ${\tt GET\ /kafka/consumergroups/\{consumerGroup\}/\{type\}/topic/\{topic\}}$ 

#### **Parameters**

Туре	Name	Description	Schema
Path	consumerGro up required	consumerGroup	string

Туре	Name	Description	Schema
Path	<b>topic</b> required	topic	string
Path	<b>type</b> required	type	enum (NEW, OLD)

#### Responses

HTTP Code	Description	Schema
200	OK	<pre></pre>
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

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

application/json

#### **Produces**

• /

### 4.2.11. Add a partition to the topic

POST /kafka/partitions/add

#### **Parameters**

Туре	Name	Description	Schema
Body	addPartition required	addPartition	AddPartition

#### Responses

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

#### Consumes

application/json

#### **Produces**

\_ /

## 4.2.12. Check the partition reassignment process

PUT /kafka/partitions/reassign/check

#### **Parameters**

Туре	Name	Description	Schema
Body	reassignStr required	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**

Туре	Name	Description	Schema
Body	reassignStr required	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**

Туре	Name	Description	Schema
Body	reassignWrap per required	reassignWrapper	ReassignWrapper

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

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**

Туре	Name	Description	Schema
Query	reassignStr optional	reassignStr	string
Body	topic required	topic	TopicDetail

#### Responses

HTTP Code	Description	Schema
201	Created	TopicMeta
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**

Туре	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**

• /

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

DELETE /kafka/topics/{topic}

#### **Parameters**

Туре	Name	Description	Schema
Path	topic required	topic	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**

• /

### 4.2.19. Create topic configs

POST /kafka/topics/{topic}/conf

#### **Parameters**

Туре	Name	Description	Schema
Path	<b>topic</b> required	topic	string

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

#### Responses

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

#### Consumes

• application/json

#### **Produces**

• /

## 4.2.20. Get topic configs

GET /kafka/topics/{topic}/conf

#### **Parameters**

Туре	Name	Description	Schema
Path	topic required	topic	string

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

application/json

#### **Produces**

• /

## 4.2.21. Update topic configs

PUT /kafka/topics/{topic}/conf

#### **Parameters**

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

#### Responses

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

#### **Consumes**

application/json

#### **Produces**

• /

### 4.2.22. Delete topic configs

DELETE /kafka/topics/{topic}/conf

#### **Parameters**

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

#### Responses

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

#### Consumes

• application/json

#### **Produces**

• /

### 4.2.23. Get topic config by key

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

#### **Parameters**

Туре	Name	Description	Schema
Path	<b>key</b> required	key	string
Path	<b>topic</b> 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**

• /

## 4.2.24. Delete a topic config by key

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

#### **Parameters**

Туре	Name	Description	Schema
Path	<b>key</b> required	key	string
Path	<b>topic</b> required	topic	string

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

application/json

#### **Produces**

• /

## 4.2.25. Create a topic config by key

POST /kafka/topics/{topic}/conf/{key}={value}

#### **Parameters**

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

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

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> required	key	string
Path	<b>topic</b> required	topic	string
Path	<b>value</b> required	value	string

#### Responses

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

#### Consumes

• application/json

#### **Produces**

• /

### 4.2.27. Tell if a topic exists

GET /kafka/topics/{topic}/exist

#### **Parameters**

Туре	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**

• /

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

POST /kafka/topics/{topic}/write

#### **Parameters**

Туре	Name	Description	Schema
Path	topic required	topic	string
Body	message required	message	string

#### Responses

HTTP Code	Description	Schema
201	Created	GeneralResponse
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	< TopicBrief > array
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

#### Consumes

• application/json

#### **Produces**

• /

## 4.3. User-controller

Security User Management Controller.

#### 4.3.1. Add user.

POST /users

#### **Parameters**

Туре	Name	Description	Schema
Body	<b>user</b> 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**

• /

#### 4.3.2. Get user list.

GET /users

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

application/json

#### **Produces**

• /

## 4.3.3. Modify user information.

PUT /users

#### **Parameters**

Туре	Name	Description	Schema
Body	<b>user</b> 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**

• /

#### 4.3.4. Delete user.

#### DELETE /users/{username}

#### **Parameters**

Туре	Name	Description	Schema
Path	<b>username</b> 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**

• /

## 4.4. Zookeeper-controller

Zookeeper Controller

## 4.4.1. Get the connection state of zookeeper

GET /zk/connstate

HTTP Code	Description	Schema
200	OK	string
401	Unauthorized	No Content

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

• application/json

#### **Produces**

• /

### 4.4.2. Get the environment information of zookeeper

GET /zk/env

#### Responses

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

#### **Consumes**

• application/json

#### **Produces**

• /

### 4.4.3. Get data of a zookeeper path

GET /zk/get/path

#### **Parameters**

Туре	Name	Description	Schema
Query	path required	path	string

#### Responses

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

#### Consumes

• application/json

#### **Produces**

• /

## 4.4.4. List a zookeeper path

GET /zk/ls/path

#### **Parameters**

Type	Name	Description	Schema
Query	path required	path	string

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

application/json

#### **Produces**

• /

## 4.4.5. Get the service state of zookeeper

GET /zk/stat

#### Responses

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

#### Consumes

• application/json

#### **Produces**

• /

## **Chapter 5. Definitions**

## 5.1. AddPartition

Name	Schema
numPartitionsAdded optional	integer(int32)
replicaAssignment optional	string
topic optional	string

## 5.2. BrokerInfo

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

## 5.3. ConsumerGroupDesc

Name	Schema
consumerId optional	string
currentOffset optional	integer(int64)
groupName optional	string
<b>host</b> optional	string
lag optional	integer(int64)
logEndOffset optional	integer(int64)
partitionId optional	integer(int32)
state optional	enum (RUNNING, PENDING)
topic optional	string
type optional	enum (NEW, OLD)

## 5.4. GeneralResponse

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

## 5.5. HashMap«string,object»

*Type*: < string, object > map

## 5.6. HealthCheckResult

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

## 5.7. HostAndPort

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

## 5.8. JMXConfiguration

Name	Schema
exclude optional	JMXFilter
include optional	JMXFilter

## 5.9. JMXFilter

Name	Schema
attribute optional	object
beanNames optional	< string > array
beanRegexes optional	< Pattern > array
domain optional	string
domainRegex optional	Pattern

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

## 5.10. JMXMetricData

Name	Description	Schema
<b>collected</b> optional		boolean
<b>host</b> optional		string
metrics optional		<pre> &lt;   HashMap«string,obje   ct» &gt; array</pre>
msg optional		string
timestamp optional	Example: "yyyy-MM-dd HH:mm:ss"	string

## 5.11. JMXMetricDataV1

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

## 5.12. JMXQuery

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

## 5.13. Map«int,long»

*Type*: < string, integer(int64) > map

### 5.14. Pattern

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

## 5.15. ReassignWrapper

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

## 5.16. TopicAndPartition

*Type*: object

## 5.17. TopicBrief

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

## 5.18. TopicDetail

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

## 5.19. TopicMeta

Name	Schema
partitionCount optional	integer(int32)
replicationFactor optional	integer(int32)
topicCustomConfigs optional	< string, object > map
topicName optional	string
topicPartitionInfos optional	< TopicPartitionInfo > array

## 5.20. TopicPartitionInfo

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

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

## 5.21. User

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

## 5.22. ZkServerClient

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

## 5.23. ZkServerEnvironment

Name	Schema
attributes optional	< string, string > map

## 5.24. ZkServerStat

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