

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. Add a partition to the topic	13
4.2.11. Check the partition reassignment process	14
4.2.12. Execute the partition reassignment.....	15
4.2.13. Generate plan for the partition reassignment.....	16
4.2.14. List topics	16
4.2.15. Create a topic.....	17
4.2.16. Describe a topic by fetching the metadata and config	18
4.2.17. Delete a topic (you should enable topic deletion	18
4.2.18. Create topic configs	19
4.2.19. Get topic configs	20
4.2.20. Update topic configs.....	20

4.2.21. Delete topic configs	21
4.2.22. Get topic config by key	22
4.2.23. Delete a topic config by key	23
4.2.24. Create a topic config by key	23
4.2.25. Update a topic config by key	24
4.2.26. Tell if a topic exists	25
4.2.27. Write a message to the topic, for testing purpose	26
4.2.28. List topics Brief	27
4.3. Zookeeper-controller	27
4.3.1. Get the connection state of zookeeper	27
4.3.2. Get the environment information of zookeeper	28
4.3.3. List a zookeeper path	28
4.3.4. Get the service state of zookeeper	29
5. Definitions	30
5.1. AddPartition	30
5.2. BrokerInfo	30
5.3. ConsumerGroupDesc	30
5.4. GeneralResponse	31
5.5. HashMap«string,object»	31
5.6. HostAndPort	31
5.7. JMXConfiguration	32
5.8. JMXFilter	32
5.9. JMXMetricData	32
5.10. JMXMetricDataV1	33
5.11. JMXQuery	33
5.12. Map«int,long»	33
5.13. Pattern	33
5.14. ReassignWrapper	34
5.15. TopicAndPartition	34
5.16. TopicBrief	34
5.17. TopicDetail	34
5.18. TopicMeta	35
5.19. TopicPartitionInfo	35
5.20. ZkServerClient	35
5.21. ZkServerEnvironment	36
5.22. ZkServerStat	36

Chapter 1. Overview

Kafka REST API SwaggerUI

1.1. Version information

Version : 0.1.0

1.2. Contact information

Contact : gnuhpc

Contact Email : gnuahpc@gmail.com

1.3. URI scheme

Host : localhost:8080

BasePath : /

1.4. Tags

- collector-controller : Rest API for Collecting JMX Metric Data
- kafka-controller : Kafka Controller
- zookeeper-controller : Zookeeper Controller

Chapter 2. Chapter of manual content 1

This is some dummy text

2.1. Sub chapter

Dummy text of sub chapter

Chapter 3. Chapter of manual content 2

This is some dummy text

Chapter 4. Resources

4.1. Collector-controller

Rest API for Collecting JMX Metric Data

4.1.1. Fetch all JMX metric data

```
GET /jmx/v1
```

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

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

```
POST /jmx/v2
```

Parameters

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

Responses

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

Consumes

- [application/json](#)

Produces

- [/](#)

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

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2. Kafka-controller

Kafka Controller

4.2.1. List brokers in this cluster

```
GET /kafka/brokers
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

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

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

Parameters

Type	Name	Description	Schema
Path	offset <i>required</i>	offset	integer(int64)
Path	partition <i>required</i>	partition	integer(int32)
Path	topic <i>required</i>	topic	string
Query	decoder <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	consumergroup <i>required</i>	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

- `/`

4.2.4. getLastCommitTimestamp

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

Parameters

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

Responses

HTTP Code	Description	Schema
200	OK	< string, < string, integer(int64) > map > map
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

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

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.6. List all consumer groups from zk and kafka

```
GET /kafka/consumergroups
```

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.7. 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	consumerGroup <i>required</i>	consumerGroup	string
Path	type <i>required</i>	type	enum (NEW, OLD)

Responses

HTTP Code	Description	Schema
200	OK	< string, < ConsumerGroupDescription > 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	consumerGroup <i>required</i>	consumerGroup	string
Path	type <i>required</i>	type	enum (NEW, OLD)

Responses

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

Consumes

- `application/json`

Produces

- `/`

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

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

Parameters

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

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

Responses

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

Consumes

- [application/json](#)

Produces

- [/](#)

4.2.10. Add a partition to the topic

POST /kafka/partitions/add

Parameters

Type	Name	Description	Schema
Body	addPartition <i>required</i>	addPartition	AddPartition

Responses

HTTP Code	Description	Schema
200	OK	TopicMeta

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

- `/`

4.2.11. Check the partition reassignment process

PUT /kafka/partitions/reassign/check

Parameters

Type	Name	Description	Schema
Body	reassignStr <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

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.12. Execute the partition reassignment

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

Parameters

Type	Name	Description	Schema
Body	reassignStr <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

- `/`

4.2.13. Generate plan for the partition reassignment

POST /kafka/partitions/reassign/generate

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.14. List topics

GET /kafka/topics

Responses

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

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

Consumes

- `application/json`

Produces

- `/`

4.2.15. Create a topic

POST /kafka/topics/create

Parameters

Type	Name	Description	Schema
Query	reassignStr <i>optional</i>	reassignStr	string
Body	topic <i>required</i>	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

- `/`

4.2.16. Describe a topic by fetching the metadata and config

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.17. Delete a topic (you should enable topic deletion)

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

Parameters

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

- `/`

4.2.18. Create topic configs

POST /kafka/topics/{topic}/conf

Parameters

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

- `/`

4.2.19. Get topic configs

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.20. Update topic configs

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

Parameters

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

- `/`

4.2.21. Delete topic configs

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

Parameters

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

- `/`

4.2.22. Get topic config by key

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.23. Delete a topic config by key

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.24. Create a topic config by key

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.2.25. Update a topic config by key

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

Parameters

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

Type	Name	Description	Schema
Path	value <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. Tell if a topic exists

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

Parameters

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

Responses

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

HTTP Code	Description	Schema
404	Not Found	No Content

Consumes

- `application/json`

Produces

- `/`

4.2.27. Write a message to the topic, for testing purpose

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

Parameters

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

- `/`

4.2.28. 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. Zookeeper-controller

Zookeeper Controller

4.3.1. Get the connection state of zookeeper

```
GET /zk/connstate
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.2. Get the environment information of zookeeper

```
GET /zk/env
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.3. List a zookeeper path

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

Parameters

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

Responses

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

Consumes

- `application/json`

Produces

- `/`

4.3.4. Get the service state of zookeeper

```
GET /zk/stat
```

Responses

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

Consumes

- `application/json`

Produces

- `/`

Chapter 5. Definitions

5.1. AddPartition

Name	Schema
numPartitionsAdded <i>optional</i>	integer(int32)
replicaAssignment <i>optional</i>	string
topic <i>optional</i>	string

5.2. BrokerInfo

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

5.3. ConsumerGroupDesc

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

5.4. GeneralResponse

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

5.5. HashMap«string,object»

Type : < string, object > map

5.6. HostAndPort

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

5.7. JMXConfiguration

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

5.8. JMXFilter

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

5.9. JMXMetricData

Name	Description	Schema
collected <i>optional</i>		boolean

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

5.10. JMXMetricDataV1

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

5.11. JMXQuery

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

5.12. Map«int,long»

Type : < string, integer(int64) > map

5.13. Pattern

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

5.14. ReassignWrapper

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

5.15. TopicAndPartition

Type : object

5.16. TopicBrief

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

5.17. TopicDetail

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

5.18. TopicMeta

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

5.19. TopicPartitionInfo

Name	Schema
endOffset <i>optional</i>	integer(int64)
in_sync <i>optional</i>	boolean
isr <i>optional</i>	< string > array
leader <i>optional</i>	string
messageAvailable <i>optional</i>	integer(int64)
partitionId <i>optional</i>	integer(int32)
replicas <i>optional</i>	< string > array
startOffset <i>optional</i>	integer(int64)

5.20. ZkServerClient

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

5.21. ZkServerEnvironment

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

5.22. ZkServerStat

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

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