Kafka data structures in Zookeeper

0.8:

1. Topic registration info:
/brokers/topics/[topic] :

```
Schema:
{ "fields" :
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "partitions",
       "type": {"type": "map",
                "values": {"type": "array", "items": "int", "doc": "a list of replica ids"},
                "doc": "a map from partition id to replica list"},
      }
    ]
}
Example:
{
  "version": 1,
  "partitions": {"0": [0, 1, 3] } }
}
```

2. Partition state info:
/brokers/topics/[topic]/partitions/[partitionId]/state

```
Schema:
{ "fields":
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "isr",
       "type": {"type": "array",
                "items": "int",
                "doc": "an array of the id of replicas in isr"}
      },
      {"name": "leader", "type": "int", "doc": "id of the leader replica"},
      {"name": "controller_epoch", "type": "int", "doc": "epoch of the controller that last updated the leader
and isr info"},
      {"name": "leader_epoch", "type": "int", "doc": "epoch of the leader"}
    ]
}

Example:
{
  "version": 1,
  "isr": [0,1],
  "leader": 0,
  "controller_epoch": 1,
  "leader_epoch": 0
}
```

3. Broker registration info:
/brokers/ids/[brokerId]

```
Schema:
{ "fields":
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "host", "type": "string", "doc": "ip address or host name of the broker"},
      {"name": "port", "type": "int", "doc": "port of the broker"},
      {"name": "jmx_port", "type": "int", "doc": "port for jmx"}
    ]
}

Example:
{
  "version": 1,
  "host": "192.168.1.148",
  "port": 9092,
  "jmx_port": 9999
}
```

4. Controller epoch:

/controller_epoch -> int (epoch)

5. Controller registration:

/controller -> int (broker id of the controller)

6. Consumer registration:

/consumers/[groupId]/ids/[consumerId]

```
Schema:
{ "fields":
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "pattern", "type": "string", "doc": "can be of static, white_list or black_list"},
      {"name": "subscription", "type" : {"type": "map", "values": {"type": "int"},
                                         "doc": "a map from a topic or a wildcard pattern to the number of
streams"}      }    ]
}

Example:
A static subscription:
{
  "version": 1,
  "pattern": "static",
  "subscription": {"topic1": 1, "topic2": 2}
}


A whitelist subscription:
{
  "version": 1,
  "pattern": "white_list",
  "subscription": {"abc": 1}}

A blacklist subscription:
{
  "version": 1,
  "pattern": "black_list",
  "subscription": {"abc": 1}}
```

7. Consumer owner:

/consumers/[groupId]/owners/[topic]/[partitionId] -> string (consumerId)

8. Consumer offset:

/consumers/[groupId]/offsets/[topic]/[partitionId] -> long (offset)

9. Re-assign partitions

/admin/reassign_partitions

```
{
   "fields":[
      {
         "name":"version",
         "type":"int",
         "doc":"version id"
      },
      {
         "name":"partitions",
         "type":{
            "type":"array",
            "items":{
               "fields":[
                  {
                     "name":"topic",
                     "type":"string",
                     "doc":"topic of the partition to be reassigned"
                  },
                  {
                     "name":"partition",
                     "type":"int",
                     "doc":"the partition to be reassigned"
                  },
                  {
                     "name":"replicas",
                     "type":"array",
                     "items":"int",
                     "doc":"a list of replica ids"
                  }
               ],
            }
            "doc":"an array of partitions to be reassigned to new replicas"
         }
      }
   ]
}

Example:
{
  "version": 1,
  "partitions":
     [
        {
            "topic": "Foo",
            "partition": 1,
            "replicas": [0, 1, 3]
        }
     ]
}
```

10. Preferred replication election

/admin/preferred_replica_election

```
{
   "fields":[
      {
         "name":"version",
         "type":"int",
         "doc":"version id"
      },
      {
         "name":"partitions",
         "type":{
            "type":"array",
            "items":{
               "fields":[
                  {
                     "name":"topic",
                     "type":"string",
                     "doc":"topic of the partition for which preferred replica election should be triggered"
                  },
                  {
                     "name":"partition",
                     "type":"int",
                     "doc":"the partition for which preferred replica election should be triggered"
                  }
               ],
            }
            "doc":"an array of partitions for which preferred replica election should be triggered"
         }
      }
   ]
}

Example:

{
  "version": 1,
  "partitions":
     [
        {
            "topic": "Foo",
            "partition": 1
        },
        {
            "topic": "Bar",
            "partition": 0
        }
     ]
}
```

11. Delete topics
/admin/delete_topics/[topic_to_be_deleted] (the value of the path in empty)

0.8.1

Topic Configuration

/config/topics/[topic_name]

Example

```
{
  "version": 1,
  "config": {
    "config.a": "x",
    "config.b": "y",
    ...
  }
}
```

/config/changes/[config_change_x] -> "topic_name"

Contains the name of the topic that changed.

0.9.0

Client and Topic configuration overrides: The content of both znodes has the same structure

/config/clients/[topic_name]

/config/topics/[topic_name]

```
{
  "version": 1,
  "config": {
    "config.a": "x",
    "config.b": "y",
    ...
  }
}
```

Config Change notification (Topic and Client config)

```
{"version" : 1, "entity_type":"topics/clients", "entity_name" : "topic_name/client_id"}
```

ISR Change notification

/isr_change_notification/isr_change_x

Gets created when ISR is changed at any broker, controller watches for these notifications and sends MetadataUpdateRequest to all brokers.

Broker registration info

/brokers/ids/[brokerId]

```
Schema:
{ "fields":
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "host", "type": "string", "doc": "ip address or host name of the broker"},
      {"name": "port", "type": "int", "doc": "port of the broker"},
      {"name": "jmx_port", "type": "int", "doc": "port for jmx"}
      {"name": "endpoints", "type": "array", "items": "string", "doc": "endpoints supported by the broker"}
    ]
}

Example:
{
  "version":2,
  "host","localhost",
  "port",9092
  "jmx_port":9999,
  "timestamp":"2233345666",
  "endpoints": ["PLAINTEXT://host1:9092", "SSL://host1:9093"]
}
```

ACL info. The content of these znodes have the same structure

/kafka-acl/Topic/[topic_name]

/kafka-acl/Cluster/kafka-cluster

/kafka-acl/Group/[groupId]

```
{"version": 1, "acls": [ { "host":"host1", "permissionType": "Allow","operation": "Read","principal": "User:
alice"}]}
```

0.10

Broker registration info

/brokers/ids/[brokerId]

```
Schema:
{ "fields":
    [ {"name": "version", "type": "int", "doc": "version id"},
      {"name": "host", "type": "string", "doc": "ip address or host name of the broker"},
      {"name": "port", "type": "int", "doc": "port of the broker"},
      {"name": "jmx_port", "type": "int", "doc": "port for jmx"}
      {"name": "endpoints", "type": "array", "items": "string", "doc": "endpoints supported by the broker"}
      {"name": "rack", "type": "string", "doc": "Rack of the broker. Optional. This will be used in rack aware
replication assignment for fault tolerance."}
    ]
}

Example:
{
  "version":3,
  "host":"localhost",
  "port":9092,
  "jmx_port":9999,
  "timestamp":"2233345666",
  "endpoints": ["PLAINTEXT://host1:9092", "SSL://host1:9093"],
  "rack": "us-east-1c"
}
```