KIP-546: Add Client Quota APIs to the Admin Client

Status

Current state: Accepted - 2.6.0 contains *describe* and *alter* functionality, *resolve* is pending for a future release.

Discussion thread: here

JIRA: KAFKA-7740

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Quota management via Admin Client has gone through a couple drafts of proposals (KIP-248, KIP-422). While improvements have been made to the Admin interface for configuration handling, fitting quotas into the API is awkward as they don't fit the natural key-value pairing, nor is the configuration output expressive enough to return all useful information. Therefore, it'd be beneficial to have a quota-native API for managing quotas, which would offer an intuitive and less error-prone interface, convey additional information beyond what the configuration APIs provide, and enable for future extensibility as quotas types are added or evolved.

Background

By default, quotas are defined in terms of a *user* and *client ID*, where the *user* acts as an opaque principal name, and the *client ID* as a generic group identifier. When setting quotas, an administrator has flexibility in how it specifies the *user* and *client ID* for which the quota applies, where the *user* and *clien t ID* may be specifically named, indicated as a default, or omitted entirely. Since quotas have flexible configurations, there is a method for resolving the quotas that apply to a request: a hierarchy structure is used, where the most specific defined quota will be matched to a request's *user* and *client ID*.

As represented by the current ZK node structure, the order in which quotas are matched are as follows, from highest priority to lowest (note <*user*> is a specified user principal, <*client-id*> is a specified client ID, and <*default*> is a special default user/client ID that matches to all users or clients IDs):

```
/config/users/<user>/clients/<client-id>
/config/users/<user>/clients/<default>
/config/users/<user>
/config/users/<default>/clients/<client-id>
/config/users/<default>/clients/<default>
/config/users/<default>
/config/clients/<client-id>
/config/clients/<default>
```

As such, reasoning around quotas can be complex, as it's not immediately obvious which quotas may apply to a given user and/or client ID. Providing descriptive information about how quotas are matched is the first goal of this KIP. Likewise, retrieving and modifying quota values can be done in a more expressive and robust way, which is the second goal of the KIP.

APIs

In order to clearly specify the APIs, let's first disambiguate some terminology: Every client request that is processed is associated with a quota *entity*, which is a map of entity *types* to their corresponding entity *names* for the request. Using the current entity types, an entity is of the form {*user*=test-user, *client-id*=test-client}, where user and client-id are the *types*, and test-user and test-client are the *names*. However, when specifying a quota configuration entry, only a subset of the entity types need to be provided, which is referred to as an entity *match*, for example, {*user*=<default>}.

For describing quotas, there's two modes of operation that are necessary for administration: config-centric, and entity-centric.

1. The config-centric mode describes what is exactly specified for the configuration for the given entity match. However, it would also be useful to also be able to determine which matches have configuration values defined, so the presence of a *filter* is used for gathering information about the entity matches that the administrator is interested in. This is **DescribeClientQuotas**.
2. The entity-centric mode describes what quotas apply to an entity. Note that an entity may match to various configuration entries depending on how the quotas are specified, e.g. the producer byte rate may be specified for the user, but the consumer byte rate for the client ID. Since it may not be clear how quotas were matched for an entity from the configuration, additional information should be returned to provide more context. This is **ResolveClientQuotas**.

Altering quotas only works on a config-centric manner, and therefore doesn't need distinguishing. For a given entity match, the administrator should be able to specify which quotas apply, or alternatively remove existing quotas so they no longer match. This is **AlterClientQuotas**.

The quota values are of type Double, which presents a complication in that the RPC protocol doesn't support floating point values. To accommodate this, RPC protocol message type '**double**' will be added, which will serialize doubles according to the IEEE 754 floating-point "double format" bit layout.

Types Rationale

While there's two defined entity types in AK, a server-side plugin mechanism allows for further expansion. Likewise, as use cases evolve, finer-grained quota control may be necessary. Therefore, entity types should not be statically bound to publicly defined constants, and instead the API should support flexible entity types by interpreting them as a String identifier. Any entity types that the broker doesn't understand should throw an IllegalArgumentException back to the client.

The quota types (producer byte rate, consumer byte rate, etc.) should also be given the same consideration. The possible quota applications may expand in the future, and the API shouldn't lock in which quota types are accessible. Modification of quota types that are unknown should also fail with error.

Since a fixed set of entity types aren't defined, an entity should be represented by a Map<String, String>, which maps an entity type to the entity name.

Public Interfaces

Admin client calls will be added to correspond to DescribeClientQuotas, ResolveClientQuotas, and AlterClientQuotas, with supporting types defined in the common.quotas package.

Common types in package org.apache.kafka.common.quota (2.6.0)

```
/**
 * Describes a client quota entity, which is a mapping of entity types to their names.
 */
public class ClientQuotaEntity {

    /**
     * The type of an entity entry.
     */
    public static final String USER = "user";
    public static final String CLIENT_ID = "client-id";

    /**
     * Constructs a quota entity for the given types and names. If a name is null,
     * then it is mapped to the built-in default entity name.
     *
     * @param entries maps entity type to its name
     */
    public ClientQuotaEntity(Map<String, String> entries);

    /**
     * @return map of entity type to its name
     */
    public Map<String, String> entries();
}

/**
 * Describes a component for applying a client quota filter.
 */
public class ClientQuotaFilterComponent {
```

```
    /**
     * Constructs and returns a filter component that exactly matches the provided entity
     * name for the entity type.
     *
     * @param entityType the entity type the filter component applies to
     * @param entityName the entity name that's matched exactly
     */
    public static ClientQuotaFilterComponent ofEntity(String entityType, String entityName);

    /**
     * Constructs and returns a filter component that matches the built-in default entity name
     * for the entity type.
     *
     * @param entityType the entity type the filter component applies to
     */
    public static ClientQuotaFilterComponent ofDefaultEntity(String entityType);

    /**
     * Constructs and returns a filter component that matches any specified name for the
     * entity type.
     *
     * @param entityType the entity type the filter component applies to
     */
    public static ClientQuotaFilterComponent ofEntityType(String entityType);

    /**
     * @return the component's entity type
     */
    public String entityType();

    /**
     * @return the optional match string, where:
     *         if present, the name that's matched exactly
     *         if empty, matches the default name
     *         if null, matches any specified name
     */
    public Optional<String> match();
}

/**
 * Describes a client quota entity filter.
 */
public class ClientQuotaFilter {

    /**
     * A filter to be applied to matching client quotas.
     *
     * @param components the components to filter on
     * @param strict whether the filter only includes specified components
     */
    private ClientQuotaFilter(Collection<ClientQuotaFilterComponent> components, boolean strict);

    /**
     * Constructs and returns a quota filter that matches all provided components. Matching entities
     * with entity types that are not specified by a component will also be included in the result.
     *
     * @param components the components for the filter
     */
    public static ClientQuotaFilter contains(Collection<ClientQuotaFilterComponent> components);

    /**
     * Constructs and returns a quota filter that matches all provided components. Matching entities
     * with entity types that are not specified by a component will *not* be included in the result.
     *
     * @param components the components for the filter
     */
    public static ClientQuotaFilter containsOnly(Collection<ClientQuotaFilterComponent> components);

    /**
     * Constructs and returns a quota filter that matches all configured entities.
     */
```

```
    public static ClientQuotaFilter all();

    /**
     * @return the filter's components
     */
    public Collection<ClientQuotaFilterComponent> components();

    /**
     * @return whether the filter is strict, i.e. only includes specified components
     */
    public boolean strict();
}

/**
 * Describes a configuration alteration to be made to a client quota entity.
 */
public class ClientQuotaAlteration {

    public static class Op {

        /**
         * @param key the quota type to alter
         * @param value if set then the existing value is updated,
         *              otherwise if null, the existing value is cleared
         */
        public Op(String key, Double value);

        /**
         * @return the quota type to alter
         */
        public String key();

        /**
         * @return if set then the existing value is updated,
         *         otherwise if null, the existing value is cleared
         */
        public Double value();
    }

    private final ClientQuotaEntity entity;
    private final Collection<Op> ops;

    /**
     * @param entity the entity whose config will be modified
     * @param ops the alteration to perform
     */
    public ClientQuotaAlteration(ClientQuotaEntity entity, Collection<Op> ops);

    /**
     * @return the entity whose config will be modified
     */
    public ClientQuotaEntity entity();

    /**
     * @return the alteration to perform
     */
    public Collection<Op> ops();
}
```

DescribeClientQuotas (2.6.0)

```
public class DescribeClientQuotasOptions extends AbstractOptions<DescribeClientQuotasOptions> {
    // Empty.
}

/**
 * The result of the {@link Admin#describeClientQuotas(Collection, DescribeClientQuotasOptions)} call.
 */
public class DescribeClientQuotasResult {

    /**
     * Maps an entity to its configured quota value(s). Note if no value is defined for a quota
     * type for that entity's config, then it is not included in the resulting value map.
     *
     * @param entities future for the collection of entities that matched the filter
     */
    public DescribeClientQuotasResult(KafkaFuture<Map<ClientQuotaEntity, Map<String, Double>>> entities);

    /**
     * Returns a map from quota entity to a future which can be used to check the status of the operation.
     */
    public KafkaFuture<Map<ClientQuotaEntity, Map<String, Double>>> entities();
}

public interface Admin extends AutoCloseable {
    ...

    /**
     * Describes all entities matching the provided filter that have at least one client quota configuration
     * value defined.
     * <p>
     * The following exceptions can be anticipated when calling {@code get()} on the future from the
     * returned {@link DescribeClientQuotasResult}:
     * <ul>
     *   <li>{@link org.apache.kafka.common.errors.ClusterAuthorizationException}
     *   If the authenticated user didn't have describe access to the cluster.</li>
     *   <li>{@link org.apache.kafka.common.errors.InvalidRequestException}
     *   If the request details are invalid. e.g., an invalid entity type was specified.</li>
     *   <li>{@link org.apache.kafka.common.errors.TimeoutException}
     *   If the request timed out before the describe could finish.</li>
     * </ul>
     * <p>
     * This operation is supported by brokers with version 2.6.0 or higher.
     *
     * @param filter the filter to apply to match entities
     * @param options the options to use
     * @return the DescribeClientQuotasResult containing the result
     */
    DescribeClientQuotasResult describeClientQuotas(ClientQuotaFilter filter, DescribeClientQuotasOptions
options);
}
```

ResolveClientQuotas (pending future release)

```
public class ResolveClientQuotasOptions extends AbstractOptions<ResolveClientQuotasOptions> {
    // Empty.
}

/**
 * The result of the {@link Admin#resolveClientQuotas(Collection, ResolveClientQuotasOptions)} call.
 */
public class ResolveClientQuotasResult {
    /**
     * Information about a specific quota configuration entry.
     */
    public class Entry {
        /**
         * @param source the entity source for the value
         * @param value the non-null value
         */
        public Entry(QuotaEntity source, Double value);
    }

    /**
     * Information about the value for a quota type.
     *
     * NOTE: We maintain a `Value` class because additional information may be added, e.g.,
     *       a list of overridden entries.
     */
    public class Value {
        /**
         * @param entry the quota entry
         */
        public Value(Entry entry);
    }

    /**
     * Maps a collection of entities to their resolved quota values.
     *
     * @param config the quota configuration for the requested entities
     */
    public ResolveClientQuotasResult(Map<QuotaEntity, KafkaFuture<Map<String, Value>>> config);

    /**
     * Returns a map from quota entity to a future which can be used to check the status of the operation.
     */
    public Map<QuotaEntity, KafkaFuture<Map<String, Value>>> config();

    /**
     * Returns a future which succeeds only if all quota descriptions succeed.
     */
    public KafkaFuture<Void> all();
}

public interface Admin extends AutoCloseable {
    ...

    /**
     * Resolves the effective quota values for the provided entities.
     *
     * @param entities the entities to describe the resolved quotas for
     * @param options the options to use
     * @return the resolved quotas for the entities
     */
    ResolveClientQuotasResult resolveClientQuotas(Collection<QuotaEntity> entities, ResolveClientQuotasOptions
options);
}
```

AlterClientQuotas (2.6.0)

AlterQuotas

```
/**
 * Options for {@link Admin#alterClientQuotas(Collection, AlterClientQuotasOptions)}.
 *
 * The API of this class is evolving, see {@link Admin} for details.
 */
public class AlterClientQuotasOptions extends AbstractOptions<AlterClientQuotasOptions> {

    /**
     * Returns whether the request should be validated without altering the configs.
     */
    public boolean validateOnly();

    /**
     * Sets whether the request should be validated without altering the configs.
     */
    public AlterClientQuotasOptions validateOnly(boolean validateOnly);
}

/**
 * The result of the {@link Admin#alterClientQuotas(Collection, AlterClientQuotasOptions)} call.
 *
 * The API of this class is evolving, see {@link Admin} for details.
 */
@InterfaceStability.Evolving
public class AlterClientQuotasResult {

    /**
     * Maps an entity to its alteration result.
     *
     * @param futures maps entity to its alteration result
     */
    public AlterClientQuotasResult(Map<ClientQuotaEntity, KafkaFuture<Void>> futures);

    /**
     * Returns a map from quota entity to a future which can be used to check the status of the operation.
     */
    public Map<ClientQuotaEntity, KafkaFuture<Void>> values();

    /**
     * Returns a future which succeeds only if all quota alterations succeed.
     */
    public KafkaFuture<Void> all();
}

public interface Admin extends AutoCloseable {
    ...

    /**
     * Alters client quota configurations with the specified alterations.
     * <p>
     * Alterations for a single entity are atomic, but across entities is not guaranteed. The resulting
     * per-entity error code should be evaluated to resolve the success or failure of all updates.
     * <p>
     * The following exceptions can be anticipated when calling {@code get()} on the futures obtained from
     * the returned {@link AlterClientQuotasResult}:
     * <ul>
     *   <li>{@link org.apache.kafka.common.errors.ClusterAuthorizationException}
     *   If the authenticated user didn't have alter access to the cluster.</li>
     *   <li>{@link org.apache.kafka.common.errors.InvalidRequestException}
     *   If the request details are invalid. e.g., a configuration key was specified more than once for an
entity.</li>
     *   <li>{@link org.apache.kafka.common.errors.TimeoutException}
     *   If the request timed out before the alterations could finish. It cannot be guaranteed whether the
update
     *   succeed or not.</li>
     * </ul>
     * <p>
     * This operation is supported by brokers with version 2.6.0 or higher.
     *
     * @param entries the alterations to perform
     * @return the AlterClientQuotasResult containing the result
```

```
     */
    AlterClientQuotasResult alterClientQuotas(Collection<ClientQuotaAlteration> entries,
AlterClientQuotasOptions options);
}
```

kafka-configs.sh/ConfigCommand (2.6.0)

As a result of introducing the APIs, the `ConfigCommand` will be updated to support the *users* and *clients* entity types when using the `--bootstrap-server` option. The modification to `ConfigCommand` was adopted in KIP-543, and usage will remain unchanged from the original `--zookeeper` function ality.

kafka-client-quotas.sh/ClientQuotasCommand (pending future release)

A `ClientQuotasCommand` would be constructed with an associated `bin/kafka-client-quotas.sh` script for managing quotas via command line, and would have three modes of operation, roughly correlating to each of the API calls:

1. Describe: Describes the quota entities for the given entity specification and their corresponding quota values, as explicitly specified in the configuration. The user may provide explicit entity types+names, or a pattern to apply to an entity type find matching entity names. If an entity type is omitted from the input, it is treated as a wildcard.
2. Resolve: Resolves the effective quotas for an entity, including contextual information about how those quotas were derived. This includes what configuration entries matched to the entity.
3. Alter: Modifies a quota configuration entry in an incremental manner, i.e. specify which entries to add, update, and/or remove.

Flags

Various flags will be used to accomplish these operations.

Common flags:
`--bootstrap-server`: The standard bootstrap server.
`--command-config`: Property file for the Admin client.

Operations (mutually exclusive):
`--describe`: Describes the entities that match the given specification, and prints out their configuration values.
`--resolve:` Resolves the effective quota values for an entity.
`--alter`: Alters the configuration for the given specification.

Entity specification flags (common to all):
`--names`: Comma-separated list of type=name pairs, e.g. "user=some-user,client-id=some-client-id"
`--defaults`: Comma-separated list of entity types with the default name, e.g. "defaults=user,client-id" (Note a separate flag is necessary since names are opaque.)

Exclusive to `--describe`: None.

Exclusive to `--resolve`: None.

Exclusive to `--alter`:
`--add`: Comma-separated list of entries to add or update to the configuration, in format "name=value".
`--delete`: Comma-separated list of entries to remove from the configuration, in format "name".
`--validate-only`: If set, validates the alteration but doesn't perform it.

Input

When specifying configuration entries, the form: `quota-name[=quota-value]` is used.

Output

In general, the output of the entities will be of the form: `{entity-type=entity-name, ...}`, where `entity-name` is sanitized for output since it is an opaque string. When displaying configuration values, the form: `quota-name=quota-value`.

Describe:

```
$./bin/kafka-client-quotas.sh --bootstrap-server localhost:9092 --describe \
                              --names=client-id=my-client

{user=user-one, client-id=my-client}
consumer_byte_rate=4000000
producer_byte_rate=1000000

{user=user-two, client-id=my-client}
producer_byte_rate=2000000

{user=<default>, client-id=my-client}
consumer_byte_rate=1000000
producer_byte_rate=500000
```

Resolve:

```
$./bin/kafka-client-quotas.sh --bootstrap-server localhost:9092 --resolve \
                              --names=user=user-two,client-id=my-client

consumer_byte_rate=2000000 {user=user-two, client-id=my-client}
producer_byte_rate=500000 {user=<default>, client-id=my-client}
```

Alter:

```
$./bin/kafka-client-quotas.sh --bootstrap-server localhost:9092 --alter   \
                              --names=client-id=my-client --defaults=user \
                              --add=consumer_byte_rate=2000000            \
                              --delete=producer_byte_rate

<no output on success>

$./bin/kafka-client-quotas.sh --bootstrap-server localhost:9092 --describe \
                              --names=client-id=my-client --defaults=user

{user=<default>, client-id=my-client}
consumer_byte_rate=2000000
```

Proposed Changes

In addition to the API changes above, the following write protocol would be implemented:

DescribeClientQuotas (2.6.0)

```
{
  "apiKey": 48,
  "type": "request",
  "name": "DescribeClientQuotasRequest",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "Components", "type": "[]ComponentData", "versions": "0+",
      "about": "Filter components to apply to quota entities.", "fields": [
      { "name": "EntityType", "type": "string", "versions": "0+",
        "about": "The entity type that the filter component applies to." },
      { "name": "MatchType", "type": "int8", "versions": "0+",
        "about": "How to match the entity {0 = exact name, 1 = default name, 2 = any specified name}." },
      { "name": "Match", "type": "string", "versions": "0+", "nullableVersions": "0+",
        "about": "The string to match against, or null if unused for the match type." }
    ]},
    { "name": "Strict", "type": "bool", "versions": "0+",
      "about": "Whether the match is strict, i.e. should exclude entities with unspecified entity types." }
  ]
}

{
  "apiKey": 48,
  "type": "response",
  "name": "DescribeClientQuotasResponse",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The error code, or `0` if the quota description succeeded." },
    { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
      "about": "The error message, or `null` if the quota description succeeded." },
    { "name": "Entries", "type": "[]EntryData", "versions": "0+", "nullableVersions": "0+",
      "about": "A result entry.", "fields": [
      { "name": "Entity", "type": "[]EntityData", "versions": "0+",
        "about": "The quota entity description.", "fields": [
        { "name": "EntityType", "type": "string", "versions": "0+",
          "about": "The entity type." },
        { "name": "EntityName", "type": "string", "versions": "0+", "nullableVersions": "0+",
          "about": "The entity name, or null if the default." }
      ]},
      { "name": "Values", "type": "[]ValueData", "versions": "0+",
        "about": "The quota values for the entity.", "fields": [
        { "name": "Key", "type": "string", "versions": "0+",
          "about": "The quota configuration key." },
        { "name": "Value", "type": "float64", "versions": "0+",
          "about": "The quota configuration value." }
      ]}
    ]}
  ]
}
```

ResolveClientQuotas (pending future release)

```
{
  "apiKey": 50,
  "type": "request",
  "name": "ResolveClientQuotasRequest",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "Entity", "type": "[]QuotaEntityData", "versions": "0+",
      "about": "The quota entity description.", "fields": [
      { "name": "EntityType", "type": "string", "versions": "0+",
        "about": "The entity type." },
      { "name": "EntityName", "type": "string", "versions": "0+",
        "about": "The entity name." }
    ]}
  ]
}

{
  "apiKey": 50,
  "type": "response",
  "name": "ResolveClientQuotasResponse",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "Entry", "type": "[]QuotaEntryData", "versions": "0+",
      "about": "Resolved quota entries.", "fields": [
      { "name": "ErrorCode", "type": "int16", "versions": "0+",
        "about": "The error code, or `0` if the resolved quota description succeeded." },
      { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
        "about": "The error message, or `null` if the resolved quota description succeeded." },
      { "name": "QuotaEntity", "type": "[]QuotaEntity", "versions": "0+",
        "about": "Resolved quota entries.", "fields": [
        { "name": "EntityType", "type": "string", "versions": "0+",
          "about": "The entity type." },
        { "name": "EntityName", "type": "string", "versions": "0+",
          "about": "The entity name." }
      ]},
      { "name": "QuotaValues", "type": "[]QuotaValueData", "versions": "0+",
        "about": "Quota configuration values.", "fields": [
        { "name": "Type", "type": "string", "versions": "0+",
          "about": "The quota type." },
        { "name": "Entry", "type": "[]ValueEntryData", "versions": "0+",
          "about": "Quota value entries.", "fields": [
          { "name": "QuotaEntity", "type": "[]ValueQuotaEntity", "versions": "0+",
            "about": "Resolved quota entries.", "fields": [
            { "name": "EntityType", "type": "string", "versions": "0+",
              "about": "The entity type." },
            { "name": "EntityName", "type": "string", "versions": "0+",
              "about": "The entity name." }
          ]},
          { "name": "Value", "type": "double", "versions": "0+",
            "about": "The quota configuration value." }
        ]}
      ]}
    ]}
  ]
}
```

AlterClientQuotas (2.6.0)

```
{
  "apiKey": 49,
  "type": "request",
  "name": "AlterClientQuotasRequest",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "Entries", "type": "[]EntryData", "versions": "0+",
      "about": "The quota configuration entries to alter.", "fields": [
      { "name": "Entity", "type": "[]EntityData", "versions": "0+",
        "about": "The quota entity to alter.", "fields": [
        { "name": "EntityType", "type": "string", "versions": "0+",
          "about": "The entity type." },
        { "name": "EntityName", "type": "string", "versions": "0+", "nullableVersions": "0+",
          "about": "The name of the entity, or null if the default." }
      ]},
      { "name": "Ops", "type": "[]OpData", "versions": "0+",
        "about": "An individual quota configuration entry to alter.", "fields": [
        { "name": "Key", "type": "string", "versions": "0+",
          "about": "The quota configuration key." },
        { "name": "Value", "type": "float64", "versions": "0+",
          "about": "The value to set, otherwise ignored if the value is to be removed." },
        { "name": "Remove", "type": "bool", "versions": "0+",
          "about": "Whether the quota configuration value should be removed, otherwise set." }
      ]}
    ]},
    { "name": "ValidateOnly", "type": "bool", "versions": "0+",
      "about": "Whether the alteration should be validated, but not performed." }
  ]
}

{
  "apiKey": 49,
  "type": "response",
  "name": "AlterClientQuotasResponse",
  "validVersions": "0",
  "flexibleVersions": "none",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "Entries", "type": "[]EntryData", "versions": "0+",
      "about": "The quota configuration entries to alter.", "fields": [
      { "name": "ErrorCode", "type": "int16", "versions": "0+",
        "about": "The error code, or `0` if the quota alteration succeeded." },
      { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
        "about": "The error message, or `null` if the quota alteration succeeded." },
      { "name": "Entity", "type": "[]EntityData", "versions": "0+",
        "about": "The quota entity to alter.", "fields": [
        { "name": "EntityType", "type": "string", "versions": "0+",
          "about": "The entity type." },
        { "name": "EntityName", "type": "string", "versions": "0+", "nullableVersions": "0+",
          "about": "The name of the entity, or null if the default." }
      ]}
    ]}
  ]
}
```

Kafka RPC 'double' support (2.6.0)

Note that, while the `ByteBuffer` natively supports serializing a `Double`, the format in which the value is serialized is not strongly specified, so the preference is to explicitly ensure a standard representation of double-precision 64-bit IEEE 754 format. This is achieved in Java using `Double.doubleToRawLongBits()` and `Double.longBitsToDouble()` and should be easily portable to other languages.

clients/src/main/java/org/apache/kafka/common/utils/ByteUtils.java

```
    /**
     * Read a double-precision 64-bit format IEEE 754 value.
     *
     * @param buffer The buffer to read from
     * @return The long value read
     */
    public static double readDouble(ByteBuffer buffer) {
        return Double.longBitsToDouble(buffer.getLong());
    }

    /**
     * Write the given double following the double-precision 64-bit format IEEE 754 value into the buffer.
     *
     * @param value The value to write
     * @param buffer The buffer to write to
     */
    public static void writeDouble(double value, ByteBuffer buffer) {
        buffer.putLong(Double.doubleToRawLongBits(value));
    }
```

The protocol type definition:

clients/src/main/java/org/apache/kafka/common/protocol/types/Type.java

```
    public static final DocumentedType DOUBLE = new DocumentedType() {
        @Override
        public void write(ByteBuffer buffer, Object o) {
            ByteUtils.writeDouble((Double) o, buffer);
        }

        @Override
        public Object read(ByteBuffer buffer) {
            return ByteUtils.readDouble(buffer);
        }

        @Override
        public int sizeOf(Object o) {
            return 8;
        }

        @Override
        public String typeName() {
            return "DOUBLE";
        }

        @Override
        public Double validate(Object item) {
            if (item instanceof Double)
                return (Double) item;
            else
                throw new SchemaException(item + " is not a Double.");
        }

        @Override
        public String documentation() {
            return "Represents a double-precision 64-bit format IEEE 754 value. " +
                    "The values are encoded using eight bytes in network byte order (big-endian).";
        }
    };
```

In `generator/src/main/java/org/apache/kafka/message/MessageGenerator.java`, the following operations will be used (code omitted for brevity):

generator/src/main/java/org/apache/kafka/message/MessageGenerator.java

```
Hash code: Double.hashCode(value)

Empty value: (double) 0

Parsing a default value string: Double.parseDouble(defaultValue)
```

Compatibility, Deprecation, and Migration Plan

All changes are forward-compatible, and no migration plan is necessary. It's outside the scope of this KIP to deprecate any functionality.

Rejected Alternatives

- Use existing describeConfigs/incrementalAlterConfigs for quota functionality. This falls short for a couple reasons. First, quotas entity names are more dynamic than brokers and tasks which makes them awkward to fit into generic tools which expect a single unique, distinct key, e.g. ConfigCommand. Second, there's no tool that expresses a way to get the resolved quota for an entity without some heavy engineering on the client side, which lacks extensibility and is more expensive to perform, especially over large collection of entities. Therefore, it makes sense to approach quotas as a standalone set of APIs that provide more targeted information and can properly support future extensibility.