

KIP-339: Create a new IncrementalAlterConfigs API

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Protocol APIs](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

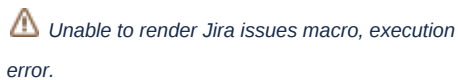
Status

Current state: *Accepted*

Discussion thread: [here](#)

Vote: [here](#)

JIRA:

A screenshot of a Jira error message. It features a yellow warning triangle icon on the left, followed by the text "Unable to render Jira issues macro, execution error." in a blue, monospaced font.

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

Motivation

The *AlterConfigs* RPC gives users a way to alter the configuration of a topic, broker, or other resource. However, the new configuration replaces any existing configuration. This makes *AlterConfigs* unusable in cases where the client does not know the full existing configuration before making the modification. This makes *AlterConfigs* less efficient, since it means that the client needs to call *DescribeConfigs* to retrieve the existing configuration before making any modification. It also introduces the possibility of "lost updates" when multiple clients enter a read-modify-write cycle around the same time.

Even worse, in some cases, the client may be unable to discover the existing configuration. "Sensitive" fields are not returned by *DescribeConfigs*. However, they will be overwritten by *AlterConfigs*.

In order to fix these issues, we should introduce a new RPC named *incrementalAlterConfigs*. The new RPC should operate incrementally, modifying only the configuration values that are specified. We should deprecate *AlterConfigs*.

Public Interfaces

Incremental AlterConfigs

```
@InterfaceStability.Evolving
public class AlterConfigOp {

    public enum OpType {
        SET((byte) 0), DELETE((byte) 1), APPEND((byte) 2), SUBTRACT((byte) 3);

        private static final Map<Byte, OpType> OP_TYPES = Collections.unmodifiableMap(
            Arrays.stream(values()).collect(Collectors.toMap(OpType::id, Function.identity()))
        );

        private final byte id;

        OpType(final byte id) {
            this.id = id;
        }

        public byte id() {
            return id;
        }

        public static OpType forId(final byte id) {
            return OP_TYPES.get(id);
        }
    }

    private final ConfigEntry configEntry;
    private final OpType opType;

    public AlterConfigOp(ConfigEntry configEntry, OpType operationType) {
        this.configEntry = configEntry;
        this.opType = operationType;
    }

    public ConfigEntry configEntry() {
        return configEntry;
    };

    public OpType opType() {
        return opType;
    };
}

public AlterConfigsResult incrementalAlterConfigs(
    Map<ConfigResource, Collection<AlterConfigOp>> configs,
    final AlterConfigsOptions options);
```

Proposed Changes

The new *IncrementalAlterConfigs* API in *AdminClient* will take a collection of operations describing the configuration modifications. There are four types of operations.

- Set: set a configuration to a value. The value must not be null.
- Delete: delete a configuration key
- Append: if a configuration key is a list of values, add to the list.
- Subtract: if a configuration key is a list of values, subtract from the list.

We will use existing *AlterConfigsOptions*, *AlterConfigsResult* API classes to pass the API config options and to return the result of configuration resource modifications.

Similar to existing *alterConfigs* API, we will keep the "transactionality" of updating several configs for the same *ConfigResource* at once. We guarantee that we never do a partial update of a collection of configs for a *ConfigResource* from a single request. On validation/update error, we will return the error for the *ConfigResource*.

Protocol APIs

There will be a new *Incremental AlterConfigsRequest*.

ModifyConfigsRequest

```
IncrementalAlterConfigsOp => INT8
0: SET
1: REMOVE
2: APPEND
3: SUBTRACT

IncrementalAlterConfigsRequest (Version: 0) => [resources] validate_only
validate_only => BOOLEAN
resources => resource_type resource_name [configs]
resource_type => INT8
resource_name => STRING
configs => config_name config_op config_value
config_name => STRING
config_op => INT8
config_value => NULLABLE_STRING
```

The Incremental AlterConfigsResponse is the same as the AlterConfigsResponse.

```
IncrementalAlterConfigsResponse (Version: 0) => [responses]
responses => resource_type resource_name error_code error_message
resource_type => INT8
resource_name => STRING
error_code => INT16
error_message => NULLABLE_STRING
```

There are two new error conditions.

- If a configuration key was specified more than once in the request for given ConfigResource, we fail ConfigResource update with the INVALID_REQUEST.
- If APPEND or SUBTRACT was specified for a configuration key that does not take a list of elements, we fail ConfigResource update with the INVALID_REQUEST.

Compatibility, Deprecation, and Migration Plan

This change is backwards compatible because the existing *alterConfigs* RPC is retained. Clients will migrate to the new *incrementalAlterConfigs* RPC as needed.

Rejected Alternatives

We could have changed *alterConfigs* so that it had an incremental mode. This would have avoided creating a new RPC. However, in order to avoid breaking compatibility, the incremental mode could not have been made the default for *AdminClient*. We would also not have been able to deprecate the non-incremental mode. This would create a confusing and dangerous stumbling block for new users. Because the problems with non-incremental mode are not immediately obvious, it is likely that many users would have made the wrong decision about what API to use.