

# KIP-453: Add close() method to RocksDBConfigSetter


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

## Status

**Current state:** Accepted

**Discussion thread:** [here](#)

JIRA:

 Unable to render Jira issues macro, execution error.

**PR:** [#6697](#)

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

## Motivation

We allow users to configure RocksDB through RocksDBConfigSetter#setConfig which passes in the RocksJava Options object. Many of the options take RocksObjects as parameters, which the user will probably construct themselves in their ConfigSetter. These objects inherit from AbstractNativeReference meaning they must be explicitly closed in order to free up the memory of the backing C++ object. Currently users have no way of closing these objects however, and since we don't know which new objects the user has constructed, we cannot close them either – so we may be leaking memory.

## Public Interfaces

### RocksDBConfigSetter

```
public interface RocksDBConfigSetter {

    /**
     * This method will be called when the store is closed and should be used to close any user-constructed
     objects
     * that inherit from RocksObject. Any such object created with new in setConfig should have close called on
     it
     * here to avoid leaking off-heap memory.
     * Objects to be closed can be saved by the user or retrieved back from options using its getter methods.
     *
     * Example objects needing to be closed include Filter and Cache
     *
     * @param storeName    the name of the store being configured
     * @param options      the Rocks DB options
     */
    default void close();
}
```

## Proposed Changes

As above, we want to extend the RocksDBConfigSetter interface with a new method, close, that we will call when closing the store. This should have a default to avoid breaking existing implementations, but users should be aware that they may want to implement this for their existing apps if they do construct any RocksObjects in their ConfigSetter. Basically any objects created in RocksDBConfigSetter#setConfig should have a match call to close() in RocksDBConfigSetter#close.

# Compatibility, Deprecation, and Migration Plan

This change should be done in a backwards compatible way. However since this is essentially a fix for existing apps, we should inform users that they will want to leverage this new feature if it applies to them/their ConfigSetter. Unfortunately this is not guaranteed to fix existing memory leaks if users ignore this functionality, but as we cannot manage the lifetime of these objects on the behalf of users (see below) we can't really do anything about it.

## Rejected Alternatives

1) Leverage the RocksDBGenericOptionsToDbOptionsColumnFamilyOptionsAdapter wrapper class to intercept every option being set, save any objects that pass through, and call close ourselves on any of them that extend RocksObject. Besides being difficult to maintain as the list of options is very long and growing, some of the RocksObjects do not necessarily share the lifetime of the store. For example, to share the memory allocated for caching across all rocksdb instances you must pass in the same Cache object to every instance. If a user passed in a new Cache object we would not be able to know whether it was meant to be shared or not, and therefore whether or not it should also be closed when a specific store is closed.

Basically, users should be responsible for closing any objects that they themselves create, and should be provided with the means of doing so.