

# CTest for `hadoop.service.shutdown.timeout`

October 11, 2023



Can you write a test for Hadoop-Common project, the purpose of the project is to test one configuration parameter "hadoop.service.shutdown.timeout". One example value of this parameter is "30s". The parameter is used by the method `static long getShutdownTimeout(Configuration conf)` method in `org/apache/hadoop/util/ShutdownHookManager.java` with

Can you generate a test that set and test this configuration parameter with the given method above? The test is in the package called `org.apache.hadoop.llmgenerated`.

Please do not use any reflection.



To test the `hadoop.service.shutdown.timeout` configuration parameter, you can create a JUnit test that:

1. Sets the configuration parameter.
2. Calls the `getShutdownTimeout` method.
3. Checks that the returned value matches the expected value.

Here's an example of how this test might look:

```
java Copy code

package org.apache.hadoop.llmgenerated;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.util.ShutdownHookManager;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ShutdownTimeoutTest {

    @Test
    public void testShutdownTimeout() {
        // 1. Set the configuration parameter.
        Configuration conf = new Configuration();
        conf.set("hadoop.service.shutdown.timeout", "30s");

        // 2. Call the getShutdownTimeout method.
        long timeout = ShutdownHookManager.getShutdownTimeout(conf);

        // 3. Check that the returned value matches the expected value.
        assertEquals(30 * 1000, timeout); // Expected 30 seconds in milliseconds.
    }
}
```

Ensure you have the required dependencies for Hadoop and JUnit in your build file.

This test assumes that `ShutdownHookManager.getShutdownTimeout(Configuration conf)` is a public static method and is accessible from the test class. If the method's accessibility is different, modifications might be needed to make the test work.



Can you also test the timeout with the code logic rather than purely check the value is expected? The timeout is used at the static phase when "ShutdownHookManager" object is created.



Okay! I understand that you want to test the behavior that results from the timeout setting, rather than just checking the returned value of the `getShutdownTimeout` method.

Since the timeout impacts the behavior of the `ShutdownHookManager` during its static phase, we'll need to check that behavior directly. One common scenario where this timeout is applied is when shutting down a service (or JVM). If the shutdown hook takes longer than the configured timeout, then it should be forced to stop.

For the sake of this test, let's create a mock service that takes a long time to shut down. We'll add this service to the `ShutdownHookManager` and then try to shut it down. We should expect the shutdown to be forced after the configured timeout.

Here's how the test might look:

```
java Copy code

package org.apache.hadoop.llmgenerated;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.util.ShutdownHookManager;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertTrue;

public class ShutdownTimeoutTest {

    private Thread longRunningHook;

    @Before
    public void setup() {
        longRunningHook = new Thread(() -> {
            try {
                // Simulate a long shutdown process that takes 60 seconds.
                Thread.sleep(60 * 1000);
            } catch (InterruptedException e) {
                // This should be caught if the shutdown is forced before 60 seconds.
            }
        });

        ShutdownHookManager.get().addShutdownHook(longRunningHook, 0);
    }

    @After
```

```

public void cleanup() {
    // Cleanup any shutdown hooks that might have been left over.
    ShutdownHookManager.get().removeShutdownHook(longRunningHook);
}

@Test
public void testShutdownTimeoutBehavior() throws InterruptedException {
    // Set the configuration parameter for 30 seconds.
    Configuration conf = new Configuration();
    conf.set("hadoop.service.shutdown.timeout", "30s");

    // Start the long running shutdown hook.
    longRunningHook.start();

    // Wait for 40 seconds (10 seconds more than the expected timeout).
    Thread.sleep(40 * 1000);

    // Assert that the long running hook was interrupted and has finished.
    assertTrue(!longRunningHook.isAlive());
}
}

```

This test will validate that the shutdown hook is actually being forced to stop after the 30-second timeout. If the `ShutdownHookManager` respects the configured timeout, the `longRunningHook` should have been interrupted and stopped before the 40-second sleep in the test completes.

Note: This test assumes that `ShutdownHookManager.get()` is a way to obtain an instance of `ShutdownHookManager`. The exact methods and behavior might vary based on the actual Hadoop implementation.



The class `ShutdownHookManager` is loaded in `@Before` but you set the parameter later, is this correct? Can you set the timeout to a smaller value and check whether it is indeed passed into `ShutdownHookManager` with static long `getShutdownTimeout(Configuration conf)`



You're right; my apologies for the oversight. The configuration must be set before the `ShutdownHookManager` class is loaded in order to affect the timeout during the static phase.

Let's correct that:

1. We'll set the configuration first, ensuring the `ShutdownHookManager` class is loaded afterward.
2. We'll use a shorter timeout to make the test run faster.
3. We'll verify the timeout was correctly picked up by the `ShutdownHookManager`.

Here's the revised test:

java

Copy code

```

package org.apache.hadoop.llmgenerated;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.util.ShutdownHookManager;
import org.junit.After;
import org.junit.Before;

```

```

import org.junit.Test;

import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertEquals;

public class ShutdownTimeoutTest {

    private Thread longRunningHook;
    private long configuredTimeout;

    @Before
    public void setup() {
        // 1. Set the configuration parameter for a shorter timeout.
        Configuration conf = new Configuration();
        conf.set("hadoop.service.shutdown.timeout", "5s");

        // Verify the timeout was picked up by ShutdownHookManager
        configuredTimeout = ShutdownHookManager.getShutdownTimeout(conf);
        assertEquals(5 * 1000, configuredTimeout); // 5 seconds in milliseconds

        // 2. Add a long running shutdown hook.
        longRunningHook = new Thread(() -> {
            try {
                // Simulate a long shutdown process that takes 20 seconds.
                Thread.sleep(20 * 1000);
            } catch (InterruptedException e) {
                // This should be caught if the shutdown is forced before 20 seconds.
            }
        });

        ShutdownHookManager.get().addShutdownHook(longRunningHook, 0);
    }

    @After
    public void cleanup() {
        // Cleanup any shutdown hooks that might have been left over.
        ShutdownHookManager.get().removeShutdownHook(longRunningHook);
    }

    @Test
    public void testShutdownTimeoutBehavior() throws InterruptedException {
        // Start the long running shutdown hook.
        longRunningHook.start();

        // Wait for 10 seconds (5 seconds more than the expected timeout).
        Thread.sleep(10 * 1000);
    }
}

```

```
        // Assert that the long running hook was interrupted and has finished.  
        assertTrue(!longRunningHook.isAlive());  
    }  
}
```

With this revised test, we're:

1. Setting the configuration to a 5-second timeout.
2. Verifying that ShutdownHookManager has indeed picked up this configuration.

[Continue this conversation](#)

[Report content](#) | [Terms of use](#) | [Privacy policy](#)