



Title Of The Document	Creating Test Cases
Author	Sai Sruthi Adina
Reviewed By	
Signed Off by	
Date Completed	02/04/2024

1. Introduction

This article describes the steps performed to create a test case for a message flow node by using messages that are recorded as they pass through a message flow.

2. TestCases Creation

Step1: Create a simple message flow consisting of Http Input,reply and Compute node.

Step2: By using Flow Exerciser record the flow of messages through message flow. After completion of recording we will get files with request & response by saving the record process.

Note: We can directly give the request & response from file explorer path (or) we can paste those files in application without recording.

Step3: Right click on the node & click on “Create test case” option

Step4: By clicking Create test case we will get the wizard to choose input & output file

Create test case

Create test case

Use this wizard to create a test case for a message flow node. Choose an input and output file to create input and output message assemblies. An input message assembly is used to invoke the node in the test case. An output message assembly is used to compare the expected and actual output message.

Application

Unit_Test

Browse...

Flow

Test_Cases

Node

Compute

Input terminal

In

Browse...

Output terminal

Out

Browse...

Select an input file to construct a message assembly to invoke the node.

Browse...

Clear

Select an output file to compare against the actual output message.

Browse...

Clear

Select matchers to be included in the generated test:

☒ Assert node call count

☒ Assert terminal propagate count

Select message trees to compare in the generated test:

☐ Message body

☐ Environment

☐ Local environment

☐ Exception list

Select setup and cleanup methods

☐ BeforeAll

☐ AfterAll

* Unit_Test_Test_Cases_Compute_0002_Test

* Test generated by IBM App Connect Enterprise Toolkit 12.0.11.3 on Apr 4, 2024, 4:00:07 PM

*/

@AfterEach

public void cleanupTest() throws TestException {

// Ensure any mocks created by a test are cleared after the test runs

TestSetup.restoreAllMocks();

}

@Test

public void Unit_Test_Test_Cases_Compute_TestCase_001() throws TestException {

// Define the SpyObjectReference

SpyObjectReference nodeReference = new SpyObjectReference().application("Unit_Test").messageFlow("Test_Cases").node("Compute");

// Initialise a NodeSpy

NodeSpy nodeSpy = new NodeSpy(nodeReference);

// Check that the actual Node name matches the expected name

assertThat(nodeSpy.nodeName(), Matchers.equalTo("Compute"));

// Assert the number of times that the node is called

assertThat(nodeSpy, nodeCallCounts(0));

// Assert the terminal propagate count for the message

assertThat(nodeSpy, terminalPropagateCounts("failure", 0));

assertThat(nodeSpy, terminalPropagateCounts("out", 0));

assertThat(nodeSpy, terminalPropagateCounts("out1", 0));

assertThat(nodeSpy, terminalPropagateCounts("out2", 0));

assertThat(nodeSpy, terminalPropagateCounts("out3", 0));

assertThat(nodeSpy, terminalPropagateCounts("out4", 0));

}

Activate Windows

Go to Settings to activate Windows.

< Back

Next >

Finish

Cancel

Step5: Click on finish then, it will create a Test project with java code.

Step6: Run that test project & observe response in JUnit view of the toolkit.