



Cognitier SimpleIPC

Exercise 2 – Filtering Arguments

Version 1.0.0.1

April 21, 2009

Copyright © 2009 Cognitier, Inc. All rights reserved.

Cognitier and the Cognitier logo are trademarks of Cognitier, Inc. All other company and product names referred to may be trademarks of their respective owners. This documentation is copyrighted material and is intended exclusively for use by licensed users of Cognitier software. The information in this document is subject to change without notice.

Exercise 2 – Filtering Arguments

The point of this exercise is to demonstrate how to intercept the input arguments sent into a routine or the output arguments prepared by a routine. The situation is that you have a need to filter these inputs and outputs for calls to any routine for the Instance/Application combination. By using the "Before Routine Invoke" and "After Routine Invoke" events, you can write this code just once and it will fire regardless of the routine specified by the client for invocation.

Step 1: If there is an IPC server process running, terminate it before proceeding. If you see a command prompt console window for the IPC server process, you can click on it with your mouse and then hit the ENTER key and it will initiate the server shut-down. Alternatively, you can use the Server Status tab of the CTConsole to terminate the server. Likewise, you could use Task Manager to end the CTServerInst1.exe process.

Step 2: Create the Before Routine Invoke routine. Go to the Author Routines tab of the CTConsole. Create a new Source Code file. Call the file FilterInputs, and select the BasicRoutineRun1 template. For this exercise, use the following code to append some text to any input arguments provided by the client and to add an additional input argument.

```
import System;
import System.Collections;
import System.Reflection;

[assembly:AssemblyVersion("1.0.0.0")]

public class FilterInputs implements CTSHost.IBasicRoutineRun1
{
    public function RunRoutine(oServerContextAccessor:
CTSHost.ServerContextAccessor, oSessionContextAccessor:
CTSHost.SessionContextAccessor, oCallParamsAccessor:
CTSHost.CallParamsAccessor) : System.Boolean
    {
        var oLogUtil: CTCommon.LogUtil;
        var bRet: System.Boolean;

        oLogUtil = CTCommon.LogUtil.Instance;
        bRet = false;
        try
        {
            var iCount: System.Int32 = 0;
```

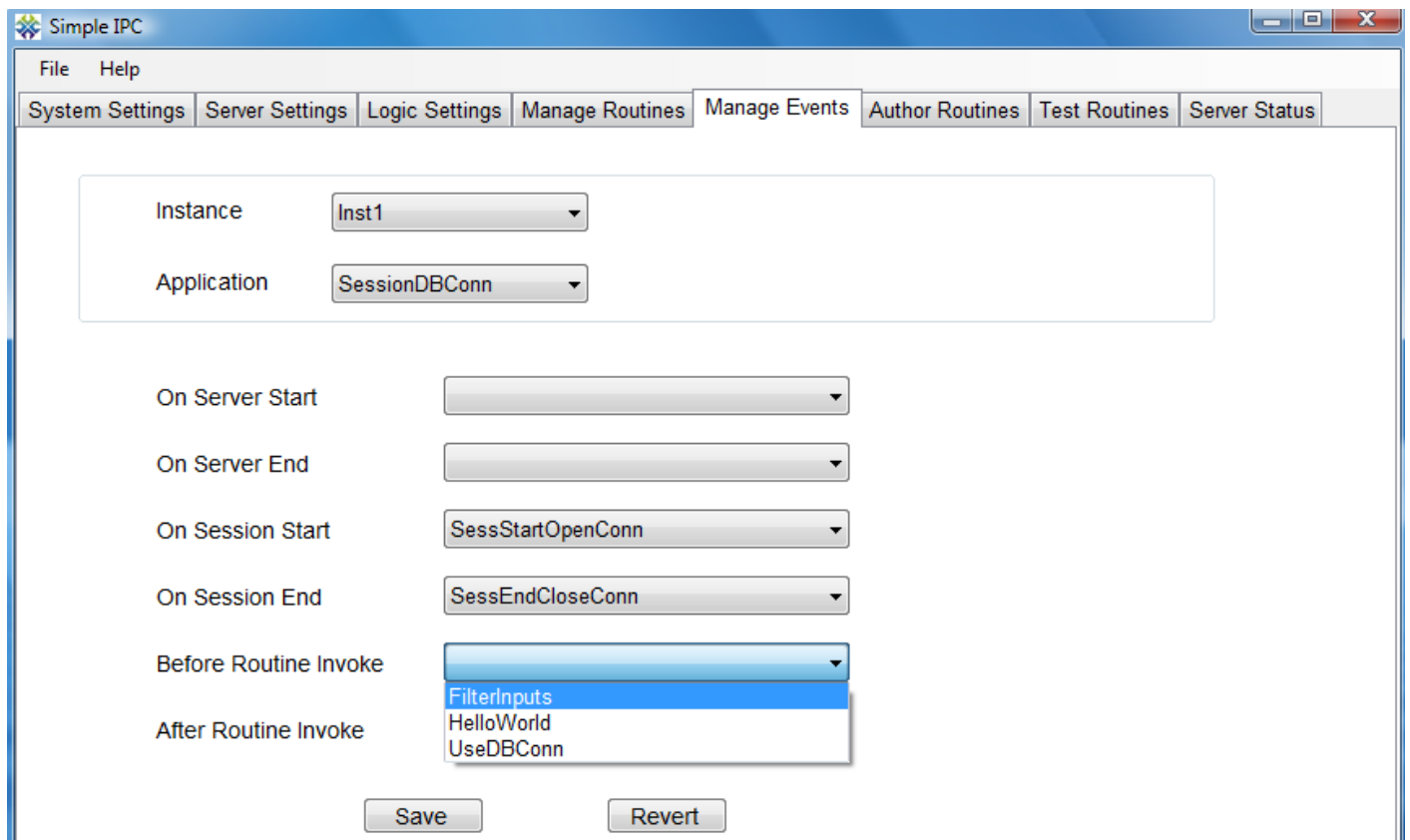
```

    var sValue: System.String = String.Empty;
    //Loop through all input arguments and append some extra text to each
string
    while (iCount < oCallParamsAccessor.InputArgs.Count)
    {
        sValue = oCallParamsAccessor.InputArgs[iCount];
        oCallParamsAccessor.InputArgs[iCount] = sValue + " - plus text added
by FilterInputs";
        iCount++;
    }
    //Add one additional input argument
    oCallParamsAccessor.InputArgs.Add("Input arg added by FilterInputs
Routine");
    bRet = true;
}
catch (e)
{
    //Write errors to the server log file
    oLogUtil.LogOutput(CTCommon.Constants.LogLevel.Errors,
CTCommon.Constants.LogOpType.RoutineExecution, e);
}
    return bRet;
} //end function
} //end class

```

Step 3: Register the routine we just created. Go to the Manage Routines tab and click the "Refresh List" button so that our new dll will appear in the dlls drop-down. Select FilterInputs.dll, enter a routine name of FilterInputs, and save it.

Step 4: Associate the routine we just created with the "Before Routine Invoke" event of the SessionDBConn application. Go to the Manage Events tab and select the App1 application and then the SessionDBConn application (in order to refresh the lists of routines in the event drop-downs). Select the FilterInputs routine from the drop-down for the "Before Routine Invoke" event and click the Save button.



Step 5: Create the After Routine Invoke routine. Go to the Author Routines tab of the CTConsole. Create a new Source Code file. Call the file FilterOutputs, and select the BasicRoutineRun1 template. For this exercise, use the following code to add an additional output argument.

```
import System;
import System.Collections;
import System.Reflection;

[assembly: AssemblyVersion("1.0.0.0")]

public class FilterOutputs implements CTSHost.IBasicRoutineRun1
{
    public function RunRoutine(oServerContextAccessor:
CTSHost.ServerContextAccessor, oSessionContextAccessor:
CTSHost.SessionContextAccessor, oCallParamsAccessor:
CTSHost.CallParamsAccessor) : System.Boolean
    {
        var oLogUtil: CTCommon.LogUtil;
        var bRet: System.Boolean;
```

```

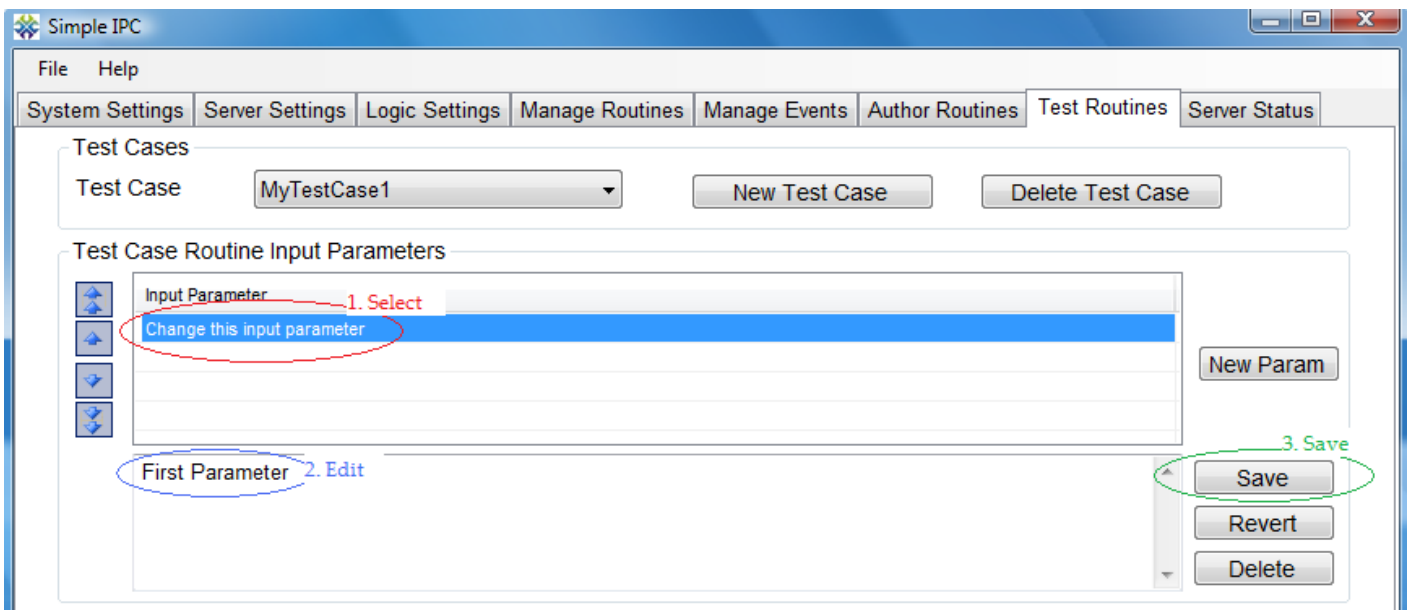
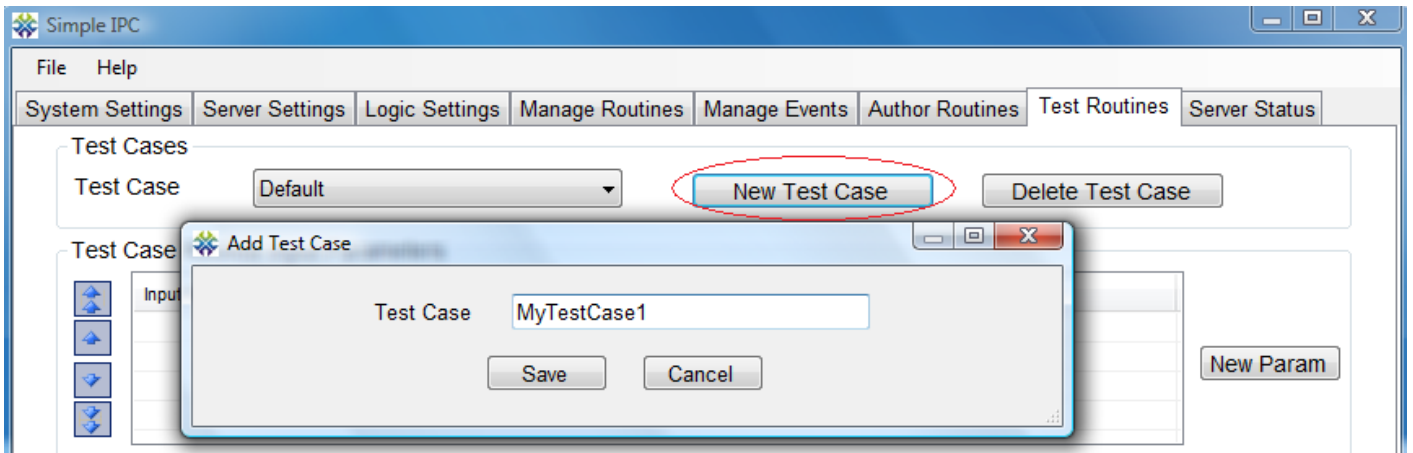
oLogUtil = CTCommon.LogUtil.Instance;
bRet = false;
try
{
oCallParamsAccessor.OutputArgs.Add("Output arg added by FilterOutputs");
bRet = true;
}
catch (e)
{
//Write errors to the server log file
oLogUtil.LogOutput(CTCommon.Constants.LogLevel.Errors,
CTCommon.Constants.LogOpType.RoutineExecution, e);
}
return bRet;
} //end function
} //end class

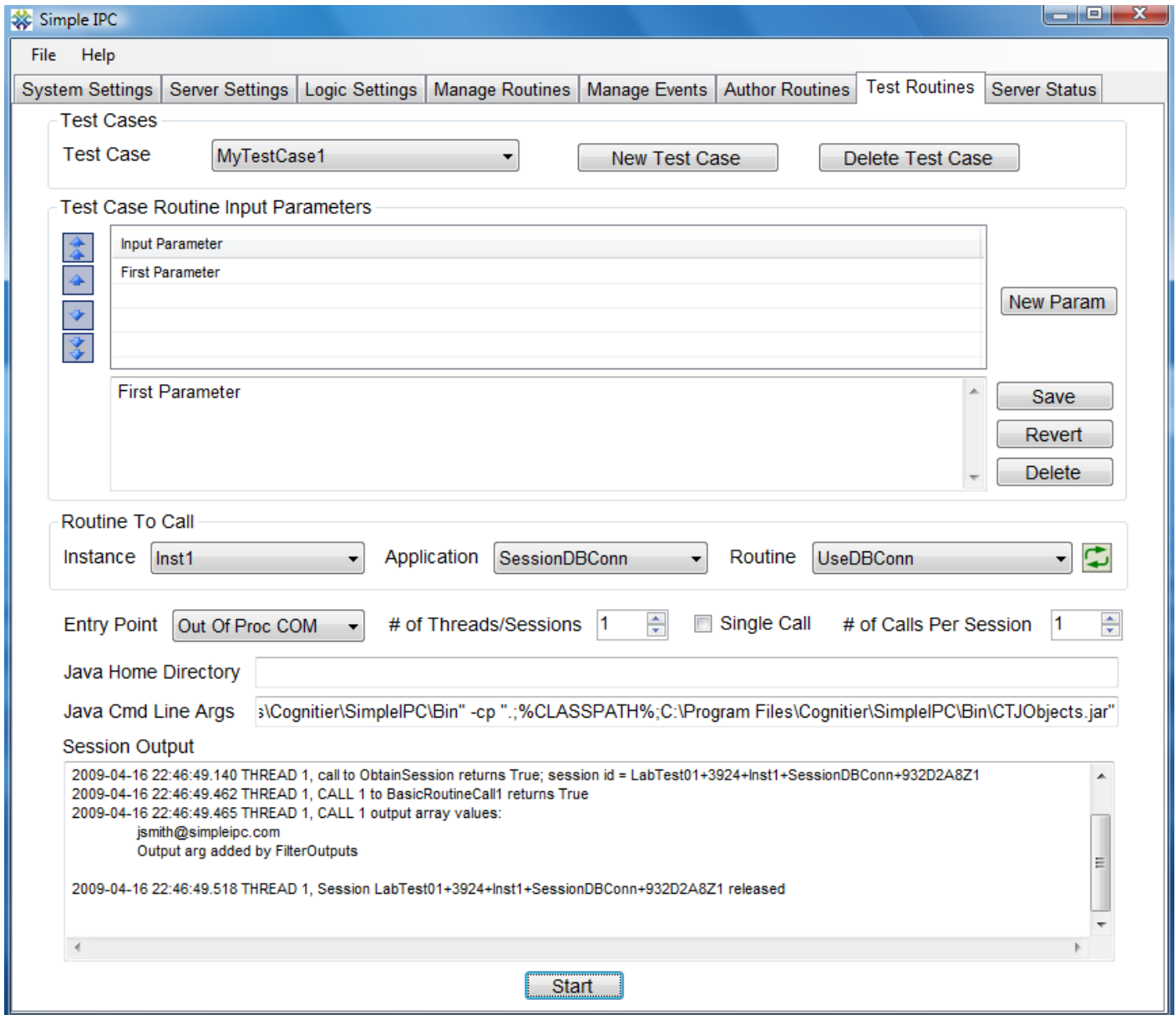
```

Step 6: Register the routine we just created. Go to the Manage Routines tab and click the "Refresh List" button so that our new dll will appear in the dlls drop-down. Select FilterOutputs.dll, enter a routine name of FilterOutputs, and save it.

Step 7: Associate the routine we just created with the "After Routine Invoke" event of the SessionDBConn application. Go to the Manage Events tab and select the App1 application and then the SessionDBConn application (in order to refresh the lists of routines in the event drop-downs). Select the FilterOutputs routine from the drop-down for the "After Routine Invoke" event and click the Save button.

Step 8: Go to the Test Routines tab of the CTConsole and test your work. Click the "New Test Case" button and create a new Test Case. Call it "MyTestCase1". A default parameter will be added to the test case. Change the value of the test case parameter to "First Parameter". To do this, select the parameter within the grid (it will already be selected), change the value in the text box beneath the grid, and click the "Save" button. Make sure the selected instance is "Inst1", the selected application is "SessionDBConn", and the selected routine is "UseDBConn". Set the number of threads/sessions to one and the number of calls per session to one and click the "Start" button. Review the test execution output in the text area at the bottom of the form. Ensure that the output argument created by the FilterOutputs routine is present.





Open the log file for the server that ran to host this method invocation. It should be the file with the latest timestamp in the \Cognitier\SimpleIPC\Inst1\ServerState\SessionDBConn\Log\[Current Date] directory. Use Notepad to open the file and do a "Find" on the string "#####Input Arg". Ensure that your routine received the input arguments as modified by your FilterInputs routine. The output in the log should be similar to the following:

```
4/17/2009 3:46:49 AM|Verbose|RoutineExecution|3924|4|Inst1|SessionDBConn|#####Input Arg
0 = First Parameter - plus text added by FilterInputs
4/17/2009 3:46:49 AM|Verbose|RoutineExecution|3924|4|Inst1|SessionDBConn|#####Input Arg
1 = Input arg added by FilterInputs Routine
```