CORBA

This topic covers basic steps and operations to test a CORBA (Common Object Request Broker Architecture) server. There are several ways to ensure the correct functionality of a CORBA server; the following are a few examples and simple exercises to help you better understand how SOAtest can simplify the process of server testing. Different scenarios will show how SOAtest can be incorporated into the testing of non-SOAP servers.

Sections include:

Scenario 1: CORBA Client Has Not Yet Been Implemented

Note: Continue to Scenario 2 if you already have a Java client created.

To use the interfaces/IDL offered by the server, you need to generate the java stubs on the client side. In this section we are going to cover simple IDL to Java conversions.

A sample Calculator.idl file for the following exercise is included in `<SOAtest installation directory/<version>/eclipse/plugins/com.parasoft.xtest.libs.web_<version-date>/root/build/examples/CORBA`. In order to use IDLJ, make sure you have J2SDK installed and set the `PATH` variable so you can access the J2SDK's executables from any directory.

To convert IDL to Java using IDLJ, complete the following:

1. In command prompt, change the current directory to the folder that contains Calculator.idl (In this example `C:\Program Files\Parasoft\SOAtest\[SOAtest version number]\eclipse\plugins\com.parasoft.xtest.libs.web_9.6.0.20130917\root\build\examples\CORBA`)
2. Type: "`idlj –pkgTranslate Persistent examples.CORBA –fall Calculator.idl`" to automatically generate packages with correct paths.
3. Compile the java files by typing: `javac/examples/CORBA/*.java`.

Now you have the necessary class files needed to communicate with the server. Please continue on to Scenario 2 to interface SOAtest with an existing java client.

For more information on IDLJ see the Oracle Java documentation.

Scenario 2: Interfacing SOAtest with an Existing Java Client

In this section we will demonstrate how to invoke Java services from a CORBA server by using SOAtest's Extension tool.

1. Create an Extension tool by right-clicking on the test suite and select **Add Test> Standard Test> New Tool> Extension**.
2. Select the Extension tool node, in the right GUI panel select the appropriate language from the **Language** drop-down menu to access your CORBA Java Client.
   For example, for Jython you can enter something similar to the following in the **Text** field:

```
# In our example, examples.CORBA.PersistentClient is our CORBA Java Client
        from examples.CORBA import *
        from java.lang import *
        def foo(input, context):
                # Here we are Initializing the client by providing location of the server,
                 # port number, and the service name
                 client = PersistentClient("goldfish.parasoft.com", 2222, "GoldfishCorbaServer")
                # Here we are making the actual Method Invocation onto the Service "add(x,y)"
                return client.add(3, 5)
```

3. Right-click within the **Text** field and select **Evaluate** from the shortcut menu to make sure the syntax is correct. If the syntax is correct, the name of the function should be auto-populated into the **Method** drop-down menu: `foo()`.
4. Right click on the Extension tool node and select **Add Return Value Output> Existing Output> Edit** to show the returned values after execution of the test.
5. Run the test, if the test succeeds the return values should appear in the right GUI panel.
6. If the test failed, returning a Null Pointer exception on the edit screen; check the CORBA server and make sure the server is listening on the designated port and that the service is up and running.

Scenario 3: Interfacing SOAtest with an Existing non-Java Client

In this section we will demonstrate how to invoke non-Java services from CORBA server by using SOAtest's External tool.

1. Create an external tool by right clicking on the test suite and select **Add Test> Standard Test> New Tool> External Tool**.
2. Select the External tool node and change its name to **CORBA Client**.
3. Click on the **Browse** button and select the path to the CORBA client executable.
4. If CORBA client takes in parameters, add each argument buy clicking on the **ADD** button. A new line will get generated, allowing users to input a flag and argument associated with the executable.
5. Double-click on the line generated to enter flag and argument. A new dialog box will pop up; change the name and argument accordingly.
6. If you wish to use a parameterized value, select **Parameterized** in the **Value** drop-down menu and select variable name in the **Variable** drop-down menu then click **OK**.
7. In the right GUI panel select the **Keep output** check box to keep the returned values after each test run.
8. Right-click the External tool node and select **Add Return Value Output> Existing Output> Edit** to show the returned values after execution of the test.
9. Run the test, if the test succeeds return values should appear in the right GUI panel.
10. If the test failed, returning a Null Pointer exception on the edit screen; check the CORBA server and make sure the server is listening on the designated port and that the service is up and running.