SETUP GUIDE

Revision History

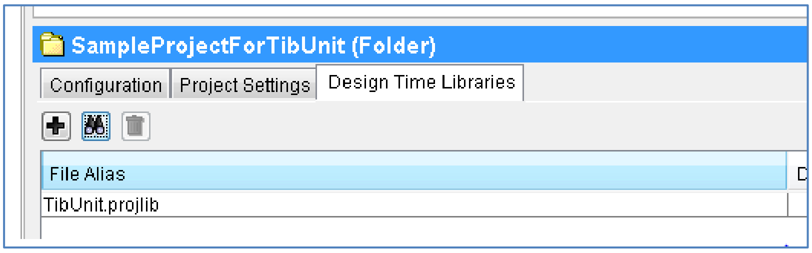
DATE	**VERSION**	**DESCRIPTION**	**AUTHOR**
29-jan-2016	1.0	Draft Version Created	Ashish Trivedi
29-jan-2016	1.1	Review comment incorporation	Ashish Trivedi

What one needs to know (Guidelines) :

1. Each test process is called as a TibUnit.
2. Their needs to be at least one input file corresponding to every TibUnit created. Multiple input files are also accepted. Each TibUnit executes once for every input file corresponding to that TibUnit.
3. The name convention of the input file is "**input_*TibUnitName*_<id>.xml**", where *TibUnitName* is the name of the TibUnit corresponding to which, this file has to act as input. **e.g.**.: input_TestLogin_1.xml
4. The name convention of the output file is "**output_*InputFileName***", where *InputFileName* is the name of the input file corresponding to which, this file has to act as output. **e.g**. : output_ input_TestLogin_1.xml
5. For creating every new TibUnit, you need to copy the SampleTest, rename it and add implementation to it.

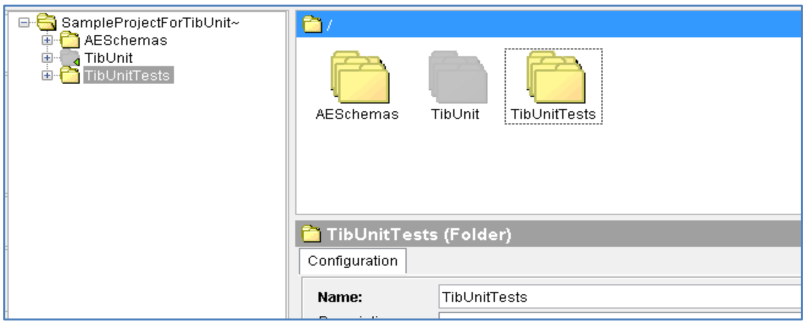
Steps to be followed by developers to use TIBUnitTestSuite:

1. Import the **TibUnitTestSuite.projlib** in the project's design Time Libraries.



File Alias: TibUnit.projlib

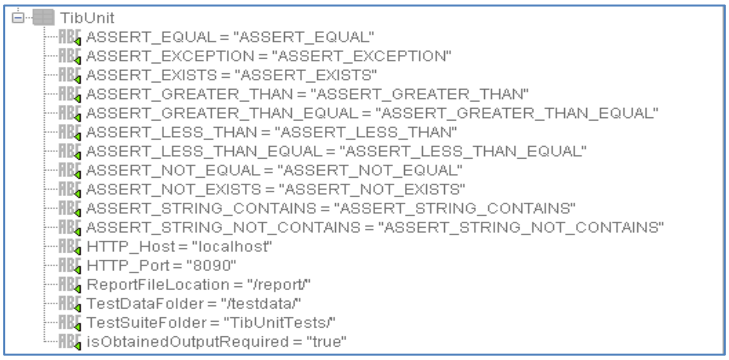
2. In the project at root level, create a folder named: **'TibUnitTests'**. This name is also specified in the global variables list.



Name: TibUnitTests

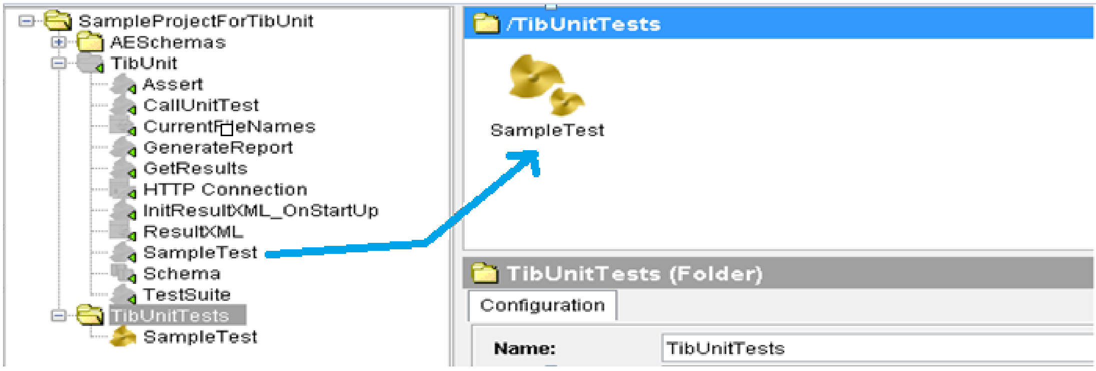
3. All TibUnits are to be created inside this folder. Any hierarchy can be followed inside this folder to systematically organize the TibUnits.

4. In the global variables, there is a folder named ‘TibUnit’ which contains the variables which configure the TibUnit test suite. Location for test data (input and output files) and report files are mentioned here. They can be changed if required by overriding. Note: **Don’t** alter the ASSERT variables as they are a part of implementation.



```
TibUnit
  ASSERT_EQUAL = "ASSERT_EQUAL"
  ASSERT_EXCEPTION = "ASSERT_EXCEPTION"
  ASSERT_EXISTS = "ASSERT_EXISTS"
  ASSERT_GREATER_THAN = "ASSERT_GREATER_THAN"
  ASSERT_GREATER_THAN_EQUAL = "ASSERT_GREATER_THAN_EQUAL"
  ASSERT_LESS_THAN = "ASSERT_LESS_THAN"
  ASSERT_LESS_THAN_EQUAL = "ASSERT_LESS_THAN_EQUAL"
  ASSERT_NOT_EQUAL = "ASSERT_NOT_EQUAL"
  ASSERT_NOT_EXISTS = "ASSERT_NOT_EXISTS"
  ASSERT_STRING_CONTAINS = "ASSERT_STRING_CONTAINS"
  ASSERT_STRING_NOT_CONTAINS = "ASSERT_STRING_NOT_CONTAINS"
  HTTP_Host = "localhost"
  HTTP_Port = "8090"
  ReportFileLocation = "/report/"
  TestDataFolder = "/testdata/"
  TestSuiteFolder = "TibUnitTests/"
  isObtainedOutputRequired = "true"
```

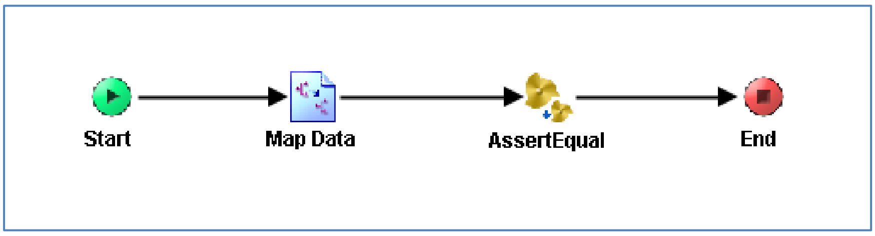
5. A TibUnit is to be created to test any process. For this, copy the sample test provided in TibUnit folder and rename it as per the requirements.



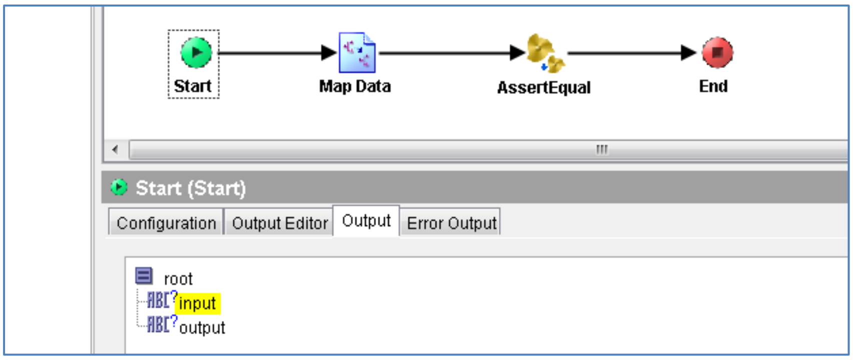
```
SampleProjectForTibUnit
  AESchemas
  TibUnit
    Assert
    CallUnitTest
    CurrentFileNames
    GenerateReport
    GetResults
    HTTP Connection
    InitResultXML_OnStartUp
    ResultXML
    SampleTest
    Schema
    TestSuite
  TibUnitTests
    SampleTest
```

Name: TibUnitTests

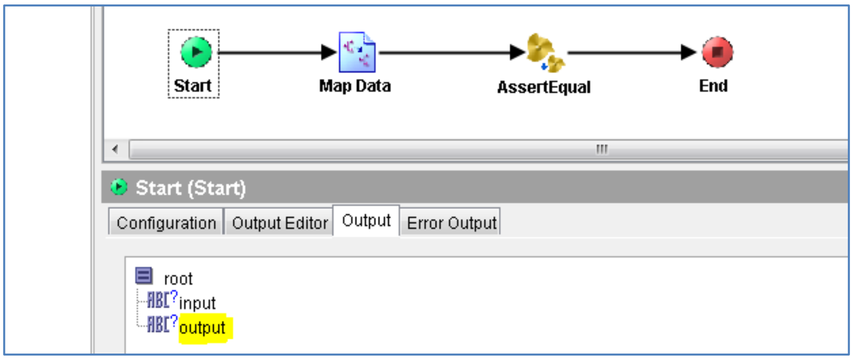
6. By default, the SampleTest will look like this:



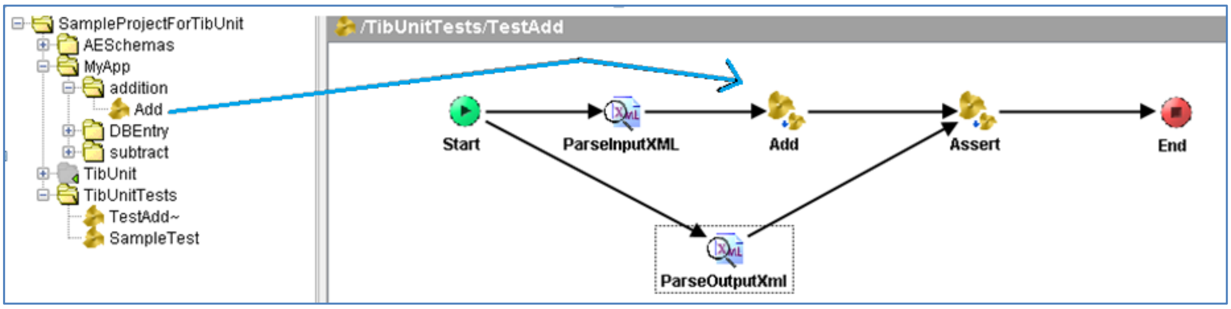
7. For every TibUnit you create, input and output files are to be provided (Follow the naming conventions mentioned at the start of this document). Input Files are mandatory as for each input file, the TibUnit runs once. So for different test data, different input files are to be provided in the location mentioned in the global variables. Input file has to contain the input data for the process, the content of this file will be available in the TibUnits Start's output, highlighted in yellow. You need to parse this input as xml and then can simply map it to the input of your process.



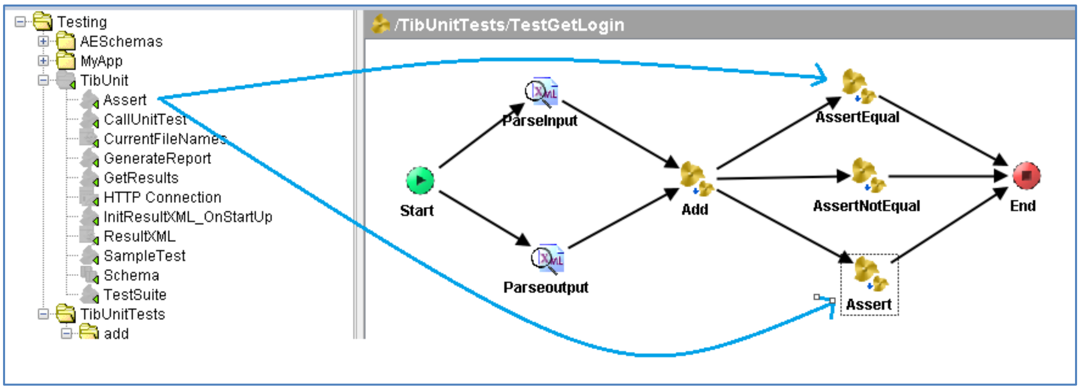
8. Start's Output also contains 'output', which contains the content of the output file (if provided) corresponding to the current input file. This provides as the expected output of your process. You also need to parse this according to the schema you want and per the assertion you need to apply. This need not necessarily be the exact schema as of the output of your process.



9. Once the sample test is copied, renamed and provided with the test data, call the process which needs to be tested and provide the input from the file or hard code it if required.



10. By default, one assert process is available in one sample test. Multiple Assert processes can also be used in a single TibUnit. For this, assert process can be dragged and dropped inside the TibUnit. One weak point for this is all asserts have the same input. So it is recommended to use different asserts in separate TibUnits.



11. Providing Inputs for Assert:

    a. AssertType: It has to take a value from global variables, depending on what the user wants to assert. Variables are self-explanatory.

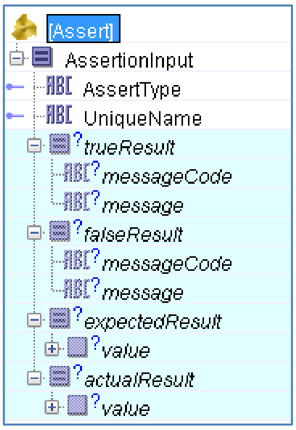
    b. UniqueName: To be provided by user to uniquely identify the TibUnit from the report file.

    c. trueResult: Message code and Message to be displayed in report file in case the assert is a success.

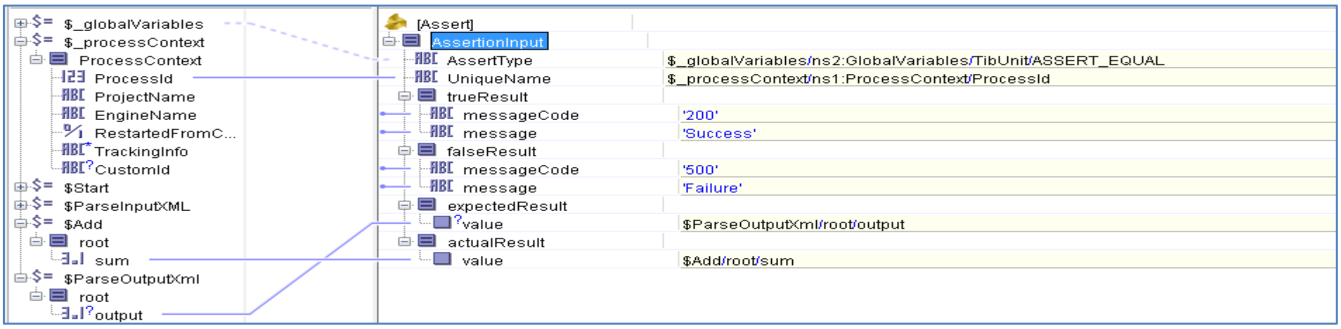
    d. falseResult: Message code and Message to be displayed in report file in case the assert is a success.

    e. expectedResult: To be mapped from the expected output obtained from 'Start' of the process.

    f. actualResult: To be mapped from the output obtained from the original process.



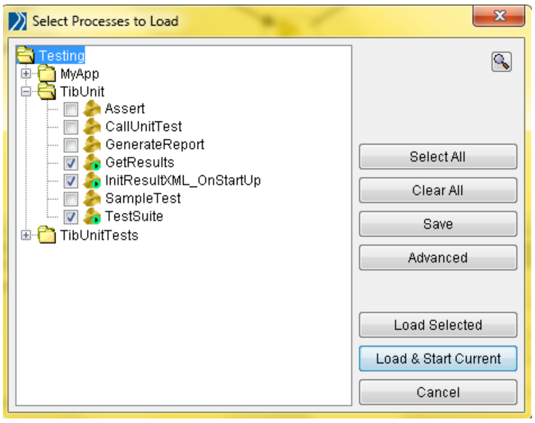
```
[Assert]
  AssertionInput
    AssertType
    UniqueName
    ?trueResult
      ?messageCode
      ?message
    ?falseResult
      ?messageCode
      ?message
    ?expectedResult
      ?value
    ?actualResult
      ?value
```



```
$_globalVariables
$_processContext
  ProcessContext
    ProcessId
    ProjectName
    EngineName
    RestartedFromC...
    TrackingInfo
    ?CustomId
$Start
$ParseInputXML
$Add
  root
    sum
$ParseOutputXml
  root
    ?output
```

[Assert]	
AssertionInput	
AssertType	$_globalVariables/ns2:GlobalVariables/TibUnit/ASSERT_EQUAL
UniqueName	$_processContext/ns1:ProcessContext/ProcessId
trueResult	
messageCode	'200'
message	'Success'
falseResult	
messageCode	'500'
message	'Failure'
expectedResult	
?value	$ParseOutputXml/root/output
actualResult	
value	$Add/root/sum

12. Once you have provided all the required inputs for all the TibUnits and input and output files in their respective locations, the test Suite is set to run. 3 Processes need to be run from the TibUnit Folder:
    a. TestSuite: Which will run all the TibUnits
    b. InitResultXML_OnStartUp: Initializing Shared Variables.
    c. GetResults: To see the result after the test has been successfully run.
    d. Note: Starter Processes of the actual project also need to be started if internally they are getting used inside any of the TibUnits.



Testing
- MyApp
- TibUnit
  - □ Assert
  - □ CallUnitTest
  - □ GenerateReport
  - ☑ GetResults
  - ☑ InitResultXML_OnStartUp
  - □ SampleTest
  - ☑ TestSuite
- TibUnitTests

13. Once the TestSuite run has completed, the report of the TestSuite can be seen at the URL: ‘http://localhost:8090/results’

14. The same report is also saved as a html file in the report file location mentioned in the global variables. The URL will give the result of the last run. Whereas the report is saved for every run executed and can be seen any time later on.

15. If any of the assert fails, the obtained output from the actual process is also saved in the location where expected output is saved with the same file name appended by ‘obtainedOutput’. To check and compare where the issue might have aroused.