ATOM DEVELOPER'S GUIDE	Pages (1/19)	
Author : Dipayan Sengupta		Date :21-06-2012

Atom Developer's Guide

Whilst all reasonable care has been taken to ensure that the details are true and not misleading at the time of publication, no liability whatsoever is assumed by Automature LLC, or any supplier of Automature LLC, with respect to the accuracy or any use of the information provided herein.

Any license, delivery and support of software require entering into separate agreements with Automature LLC.

This document may contain confidential information and may not be modified or reproduced, in whole or in part, or transmitted in any form to any third party, without the written approval from Automature LLC.

Copyright © 2012 Automature LLC

All rights reserved.

ATOM DEVELOPER'S GUIDE		Pages (2/19)
Author : Dipayan Sengupta		Date :21-06-2012

Revision History

Ver.	Date	Name	Description of Changes		
1.0	06/01/12	Dipayan Sengupta	First Edition		
1.1	15/06/12	Dipayan Sengupta	Second Edition		
1.2	21/06/12	Dipayan Sengupta	Revised Edition		

ATOM DEVELOPER'S GUIDE	Pages (3/19)	
Author : Dipayan Sengupta		Date :21-06-2012

Contents

1 Introduction	4
1.1 Document Purpose4	
1.2 Intended Audience4	
1.3 References and Other Related Documents4	
2 Dependencies	5
2.1 Zug SDK distribution:5	
3 Writing C# in-process Atoms	6
3.1 Creating a new project6	
3.2 Accessing ZugLib methods7	
3.3 Configuring the Assembly Information	
3.4 Writing a Class File9	
3.5 Architectural Mechanism for calling C# dll atoms10	
3.5 Accessing ZugLib methods10	
3.5.1 Accessing Context Variable10	
3.5.2 Writing Debug Messages10	
3.5.3 Exception Handling10	
3.6 Writing a simple atom11	
3.7 Build the project11	
3.8 Register the dll11	
3.9 Configuring the ZugINI.xml file12	
3.10 Writing a simple test case13	
4 Writing Java in-process atoms	14
4.1 Create a project14	
4.2 Accessing the ZugLib methods15	
4.3 Writing a Class File16	
4.4 Architectural Mechanism for calling Java atoms17	
4.5 Accessing ZugLib methods17	
4.5.1 Accessing Context Variable17	
4.5.2 Writing Debug Messages17	
4.5.3 Exception Handling17	
4.6 Writing a simple atom18	
4.7 Build the project18	
4.8 Configuring the ZugINI.xml file19	
4.9 Writing a simple test case19	

ATOM DEVELOPER'S GUIDE		Pages (4/19)
Author : Dinavan Sengunta		Date :21-06-2012

1 Introduction

- ³⁵ Atoms in Zug, are the basic reusable units of work. They are entities, such as functions, procedures, programs, or scripts that can be executed either within the context of a main program, or as a standalone program at the command line level in a shell (e.g. the Command Prompt in Windows).
- ³⁵ Atoms are intended to be generic enough, so that they can be applied in test case design in a range of situations. When executing as a standalone program, they are called *out-of-process* atoms. When implemented as a function or method, they are referred to as *in-process* atoms, since they can be invoked within the process context of the execution engine.
- Atoms allow users of Zug to extend the functionality of the engine, to apply it to testing different kinds of applications in new domains.

1.1 Document Purpose

This document provides detailed information about developing in-process atoms in various languages e.g. Java, C, C++, C#.

1.2 Intended Audience

This Reference Manual is intended for test case designers and programmers, and assumes prior knowledge of a high level language It also assumes a good understanding of the development and target platform environments.

1.3 References and Other Related Documents

The following documents provide additional useful information about Automature's other products, and how they relate to Zug.

- 1. Zuoz Reference Guide
- 2. Chur Programmer's Guide

ATOM DEVELOPER'S GUIDE		Pages (5/19)
Author : Dipayan Sengupta		Date :21-06-2012

2 Dependencies

Before running ZUG on a computer, it has to meet some basic requirements:

 Operating System –
 Microsoft Windows XP or higher

 Software –
 Java(TM) SE Runtime Environment 1.6

 NetBeans IDE 7.0
 Microsoft Visual Studio 2008 or higher.

2.1 Zug SDK distribution:

Zug SDK is distributed with the standard Zug kit. However, one does not require a run-time license to use it as an SDK. For testing purposes however, it is recommended to ask Automature for a "free" developer license.

3 Writing C# in-process Atoms

Kindly follow the following steps :

3.1 Creating a new project

Create a new project in Visual Studio. (Ex.- DemoProject)



ATOM DEVELOPER'S GUIDE		Pages (7/19)
Author · Dinavan Sengunta		Date :21-06-2012

3.2 Accessing ZugLib methods

You have to add a reference to the ZugLib.dll (located in the SDK folder in Zug installation directory) to your custom C# project. This will expose the ZugLib methods in your project.

Right Click on the References option in the Solution Explorer.

-	Solution Explorer
-	🕒 🛐 👼
÷	Solution 'DemoProject' (1 project)
	DemoProject
	Properties
	References
Add Referen	ce
Add Service	reference

Browse to the Zug installation directory, Navigate to SDK/COM and you will find ZugLib.dll.

😎 Add Reference	? 🔀
.NET COM Projects Browse Recent	
Look in: 🚺 COM	- G 🖻 🖻 🖽 -
Name	Date modified Ty
S ZugLib.dll	6/21/2012 5:31 PM A
•	4
File name: ZugLib	•
Files of type: Component Files (*.dll;*.tlb;*.olb;*.	ocx;*.exe;*.manifest)
	OK Cancel

3.3 Configuring the Assembly Information

You need to give the assembly information before writing the class file.

```
[ComVisible(true), Guid("b33b95fb-b977-4e2c-836a-64d33da85382"),
ClassInterface(ClassInterfaceType.None),
ComSourceInterfaces(typeof(IZugAtomExecutor))]
ProgId("Automature.DemoProject")]
```

You need to configure the GUID and also set the ProgID before you write the class file.

Right Click on the Project Name and Select Properties.



Click on Assembly Language. Select the check box stating Make assembly COM-visible and copy the GUID and replace it in the code stated above.

Assembly Informatio	n					? 🗙
Title:	DemoP	roject				
Description:						
Company:	Microso	oft				
Product:	DemoP	roject				
Copyright:	Copyrig	jht © Mi	crosoft 2	012		
Trademark:						
Assembly version:	1	0	0	0		
File version:	1	0	0	0		
GUID:	c53a7b3	31-6c60-	4288-b54	e-de594	72371	2a
Neutral language:	(None)					→ -
Make assembly	COM-V	isible		ОК		Cancel

3.4 Writing a Class File

You will need to import the following libraries additionally as mentioned below:

```
using System.Runtime.InteropServices;
using System.Reflection;
using Automature.Zug;
```

You will have to implement the dispatch method inside your custom class. The signature of the Dispatch method is as follows:

```
int dispatch(string method_name, String inputs);
```

Following is the dispatch method:

```
public class DemoClass : IZugAtomExecutor
    {
      public void dispatch(string method name, string inputs)
       ParameterInfo[] args;
       bool method_found_flag = false;
       bool arg match flag=false;
       try
          {
            PNServiceBus iClass = new PNServiceBus();
            Type PNSBObj = iClass.GetType();
            MethodInfo[] iMethods = PNSBObj.GetMethods();
            Object[] iParams = ZugLib.DelimitArg(inputs);
            foreach (MethodInfo lMethod in iMethods)
              if (lMethod.Name.ToUpper().Equals(method name.ToUpper()) )
                  {
                     method found flag = true;
                     args = lMethod.GetParameters();
                     if (iParams.Length == args.Length)
                          {
                            arg_match_flag = true;
                             IMethod.Invoke(iClass, iParams);
                            break;
                          }
                    }
                }
              if (!method found flag)
              ł
               throw new Exception("Method not found : " + method name);
              }
             if (!arg_match_flag)
              ł
              throw new Exception("Argument mismatch : " + method name);
              }
           }
```



```
Console.WriteLine("Message from dispatch ::" +
e.InnerException); prints the root cause just as in getCause in Java
ZugLib.Log("error", "error occured while invoking dispatch");
throw e;
}
```

3.5 Architectural Mechanism for calling C# dll atoms



3.5 Accessing ZugLib methods

}

3.5.1 Accessing Context Variable

In order to access context variable methods from atoms you need to write

ZugLib.SetContextVar(contextvar name, contextvar value)

3.5.2 Writing Debug Messages

In order to write debug messages from atoms you need to write ZugLib.("loglevel".message)

This "loglevel" can be "error" or "debug" or "info".

3.5.3 Exception Handling

When an exception occurs in a C# atom it can be handled using the Try and Catch block. The exception should be thrown back.

Please refer to the Section 3.6 which illustrates an example of how to handle Exceptions.

ATOM DEVELOPER'S GUIDE		Pages (11/19)
Author : Dipayan Sengupta		Date :21-06-2012

3.6 Writing a simple atom

The following is an example of a simple atom which takes two inputs as text and compares them. It also shows how to handle Exceptions by using the Try and Catch Block.

```
public void Compare (string First input, string Second input)
        {
          try
            {
           if (string.Equals(First input, Second input,
StringComparison.OrdinalIgnoreCase)
                {
                    Console.WriteLine("Strings matched successfully");
                }
                else
                {
                    ZugLib.Log("error", "Strings do not match");
                    throw new Exception("Strings do not match");
                }
            }
            catch (Exception e)
            ł
                ZugLib.Log("error", "error occurred while invoking
Compare");
                throw e;
            }
        }
```

3.7 Build the project

Build the project and DemoProject.dll file is created in the project output directory e.g. -C:\Users\Administrator\Documents\Projects\DemoProject\DemoProject\bin\Debug\DemoProject .dll

3.8 Register the dll

Open the Command Prompt as Administrator and run the following command:

%windir%\Microsoft.NET\Framework\v<*framework_version*>2.0.50727\RegAsm.exe /verbose /nologo /codebase "<*path of the dll file*>"

For 64-bit Operating System, rewrite the same as

```
%windir%\Microsoft.NET\Framework64\v<framework_version>2.0.50727\RegAsm.exe
/verbose /nologo /codebase "<path of the dll file>"
```

ATOM DEVELOPER'S GUIDE		Pages (12/19)
Author : Dipayan Sengupta		Date :21-06-2012

Example:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>RegAsm.exe /verbose
/nologo /codebase "C:\Users\Admin\Documents\Visual Studio
2010\Projects\DemoProject\DemoProject\bin\Debug\DemoProject.dll"
```

The output will be as follows:

Administrator: Command Prompt	- • •
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>RegAsm.exe /verbose /no ebase "C:\Users\Admin\Documents\Visual Studio 2010\Projects\DemoProject ect\bin\Debug\DemoProject.dll" RegAsm : warning RA0000 : Registering an unsigned assembly with /codeba use your assembly to interfere with other applications that may be inst the same computer. The /codebase switch is intended to be used only wit assemblies. Please give your assembly a strong name and re-register it. Types registered successfully C:\Windows\Microsoft NET\Examework64\u4 0 30319>	logo ∕cod ∖DemoProj se can ca alled on h signed

3.9 Configuring the ZugINI.xml file

Inside the Zug Installation directory you will find the ZugINI.xml file.

Please add the following inside <inprocesspackages> tag :

```
<prog-id>Automature.DemoProject</prog-id></prog-id>
```

</inprocesspackage>

ATOM DEVELOPER'S GUIDE		Pages (13/19)
Author : Dipayan Sengupta		Date :21-06-2012

3.10 Writing a simple test case

The following is a simple test case written in Chur. The atom simply compares two strings.

TestCase ID	Description	property	Step	Action	ActionArg_1	ActionArg_2
ZCOM001				COMDemo.Compare	This is a simple test line.	THIS IS A simple TEST line.

For more information on writing test cases in Chur, please refer to Chur Programmer's Guide.

The output of the test case is as follows:

Command Prompt		×
***************************************	******	** ^
Total time taken to initialize the Harness is -> 2114 milli Seconds.		
***************************************		**
*** Number of TestCase in Chur Sheet 5 ***		
*** Start Executing the testcases ***		
***************************************	******	**
Running TestCase ID ZCOM001 [ZCOM001] Execution Started Action COMDemo.Compare with values [This i test line., THIS IS A simple TEST line.] Strings matched successfully	s a simp	le
EZCOM001] Action COMDEMO.COMPARE SUCCESSFULLY Executed		
STATUS : PASS FOR TestCase ID ZCOM001 ***********************************	*****	××
Following are the Details of the TestCases Result Executed by ZUG Vers Premium 4.4.20120615.075	ion -> Z	UG
TestCase ID Status Time Taken(In milli-seconds) Comme	nts	
ZCOM001 pass 403		
***************************************	******	**
Total time taken to execute all the test cases (End to End) is -> 3192 onds.	milli S	ec≡

ATOM DEVELOPER'S GUIDE		Pages (14/19)
Author : Dipayan Sengupta		Date :21-06-2012

4 Writing Java in-process atoms

4.1 Create a project

Create a project in NetBeans. (Example - JavaDemo)





ATOM DEVELOPER'S GUIDE	Pages (15/19)	
Author : Dipayan Sengupta		Date :21-06-2012

4.2 Accessing the ZugLib methods

You need to add zuglib.jar file to your custom Java project.

Right Click on Libraries and add Jar Folder.



Navigate to the Zug installation Directory where you will find SDK folder. Within the SDK folder, you will find JAVA folder which contains zuglib.jar.

Look in: 🛄 JAVA 🔹 🚽 🏚 📂 🖽 🖩	3
Recent Items Recent Items Recent Items Reference as Refer	x86)/Autom
Network File name: zuglib.jar Files of type: Classpath Entry (folder, ZIP or JAR file)	Open Cancel

ATOM DEVELOPER'S GUIDE		Pages (16/19)
Author : Dipayan Sengupta		Date :21-06-2012

4.3 Writing a Class File

You need to import the following packages:

```
import com.automature.zug.exceptions.AtomExecutionException;
import com.automature.zug.exceptions.AtomNotFoundException;
import java.lang.reflect.Method;
import com.automature.zug.lib.AtomExecutor;
import com.automature.zug.lib.ZugLib;
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
```

You will have to implement the dispatch method inside your custom class. Following is the dispatch method:

```
public class JavaDemo implements AtomExecutor{
public void dispatch(String method name, ArrayList<String> inputs)
throws AtomExecutionException, AtomNotFoundException {
        boolean method found flag = false, arg match flag=false;
        try {
            Class<?> iClass = Class.forName(this.getClass().getName());
            Method[] iMethods = iClass.getMethods();
            Object iObject = iClass.newInstance();
            Object iParams[] = inputs.toArray();
            method found flag = false;
            for (Method lMethod : iMethods) {
              if (lMethod.getName().equalsIgnoreCase(method name)) {
                        method found flag = true;
 if((lMethod.getParameterTypes().length == inputs.toArray().length) ||
(inputs.isEmpty() && lMethod.getParameterTypes().length==0)) {
                                    arg match flag = true;
                                    IMethod.invoke(iObject,iParams);
                                    break;
                              }
                        }
                  if (method found flag && !arg match flag) {
                  throw new AtomNotFoundException("Argument mismatch");
            if (method found flag == false) {
      throw new AtomNotFoundException(method name);
                  }
            catch (InvocationTargetException It) {
                  throw new
AtomExecutionException("StringOperation.dispatch\n", It.getCause());
            }
            catch (Exception e) {
                  throw new
AtomExecutionException("StringOperation.dispatch\n"+e.getClass(), e);
                        }
```

ATOM DEVELOPER'S GUIDE	Pages (17/19)	
Author : Dipayan Sengupta		Date :21-06-2012

4.4 Architectural Mechanism for calling Java atoms



4.5 Accessing ZugLib methods

4.5.1 Accessing Context Variable

In order to access context variable methods from atoms you need to write ZugLib.SetContextVar(contextvar name, contextvar value);

4.5.2 Writing Debug Messages

In order to write debug messages from atoms you need to write

ZugLib.log(log_level, Message);

This "loglevel" can be "error" or "debug" or "info".

4.5.3 Exception Handling

When an exception occurs in a Java atom it can be handled using the Try and Catch block. The exception should be thrown back.

Please refer to the Section 4.6 which illustrates an example of how to handle Exceptions.

ATOM DEVELOPER'S GUIDE	Pages (18/19)
Author : Dipayan Sengupta	Date :21-06-2012

4.6 Writing a simple atom

The following is an example of a simple atom which takes two input strings and compares them. It also shows how to handle Exceptions by using the Try and Catch Block.

```
public void compare (String str1, String str2) throws
AtomExecutionException
    {
            String msg = "";
                String msg2="";
            try {
                  msg="Error while comparing two strings.";
                  if (!str1.equalsIgnoreCase(str2)) {
                        msg="Strings do not match";
                        throw new Exception("String do not match");
                  }
                 else
                 System.out.println("Strings matched Successfully");
            } catch (Exception e) {
                  ZugLib.log("error",e.getMessage());
      }
```

4.7 Build the project

Build the project and JavaDemo.jar file is created in the project output directory

e.g.-C:\Users\Admin\Documents\NetBeansProjects\JavaDemo\dist\JavaDemo.jar



4.8 Configuring the ZugINI.xml file

Inside the Zug Installation directory you will find the ZugINI.xml file.

Please add the following inside <inprocesspackages> tag :

4.9 Writing a simple test case

The following is a simple test case written in Chur. The atom simply compares two strings.

A	В	С	D	E	F	G
TestCase ID	Description	property	Step	Action	ActionArg_1	ActionArg_2
ZJAVA001				Demo.Compare	This is a simple test line.	THIS IS A simple TEST line.

For more information on writing test cases in Chur, please refer to Chur Programmer's Guide.

The output of the test case is as follows:

Command Prompt		đ	×
***************************************	****	****	iexe 🔺
*** Number of TestCase in Chur Sheet 5 ***			
*** Start Executing the testcases ***			
***************************************	****	****	K X
Running TestCase ID ZJAVA001 [ZJAVA001] Execution Started Action Demo.Compare with values [This is est line., THIS IS A simple TEST line.J Strings matched Successfully	a si	mple	t
[ZJAVA001] Action DEMO.COMPARE SUCCESSFULLY Executed			
STATUS : PASS FOR TestCase ID ZJAVA001 ***********************************	****	****	××
Following are the Details of the TestCases Result Executed by ZUG Vers Premium 4.4.20120615.075	ion ·	-> ZI	JG
TestCase ID Status Time Taken(In milli-seconds) Comme	nts		
ZJAVA001 pass 443			
***************************************	****	****	K X
Total time taken to execute all the test cases (End to End) is -> 4020 onds.	mil	1i So	9C