Document Version

Date 6 January 2017

William Lewis ([william.lewis@creativevirtual.com](mailto:william.lewis@creativevirtual.com))

Devang Vyas ([devang.vyas@creativevirtual.com](mailto:devang.vyas@creativevirtual.com))

**Distribution List**

|  |  |
| --- | --- |
| Sr. No. | Name |
|  | **Creative Virtual** |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description of Change** | **Author** |
| 03-Sep-2009 | 0.1 | Initial draft. Creating build instructions for V\_Engine source code. | William Lewis |
| 04-Sep-2009 | 1.0 | Reviewed and Updated | Avinash Gohil |
| 29-Sep-09 | 1.1 | Updated to set Environment variable for ICU and Python Installation | Mayur Panchal |
| 01-Oct-2009 | 2.0 | Reviewed and Updated | Avinash Gohil |
| 26-Oct-2009 | 2.1 | Updated Deployment Strategy | Mayur Panchal / Sonal Quadros |
| 28-Oct-2009 | 2.2 | Reviewed and Updated | Avinash Gohil |
| 30-Oct-2009 | 2.3 | Incorporated review Comments from Thenappan (Test lead). | Mayur Panchal / Sonal Quadros |
| 02-Nov-2009 | 3.0 | Reviewed and Updated | Thenappan Palaniappan |
| 21-Nov-2009 | 3.1 | Added build instructions for DLLs required by V-Builder | Bhushan Mhatre |
| 24-Nov-2009 | 3.2 | Reviewed and Updated | Avinash Gohil |
| 27-Nov-2009 | 4.0 | Reviewed | Thenappan Palaniappan |
| 3-Dec-2009 | 4.1 | Updated as per review comments received from Peter Behrend | Sonal Quadros/ Elizabeth Cherian |
| 08-Dec-2009 | 4.2 | Reviewed and Updated | William Lewis |
| 14-Dec-2009 | 4.3 | Converted text in dark red to back | William Lewis |
| 14-Dec-2009 | 5.0 | Modifying version to 5.0 as no comments were received regarding the document | William Lewis |
| 8-Jul-2012 | 6.0 | Updated for IBM WebSphere changes | Praveen Pillai |
| 21-Mar-2013 | 7.0 | Updated for incorporating steps to build C++ VA Engine in 64bit. | Sampath Kumar |
| 24-Feb-2014 | 7.1 | Updated for KBFileReadWrite DLL | Devang Vyas |
| 23-Dec-2014 | 7.2 | Updating format and instructions | William Lewis, Devang Vyas |
| 06-Jan-2017 | 7.3 | Updated Python installation instructions for V-Engine v2.2.1 onwards | Parvez Shaikh |

**Abbreviations**

|  |  |
| --- | --- |
| VA | Virtual Assistant |
| ACE | Adaptive Communication Environment |
| TAO | The Ace ORB |
| ICU | International Components for Unicode |

The statements in red refer to statements that have been modified from the previous version.

**Table of contents**

[Purpose 6](#_Toc471404712)

[Scope 6](#_Toc471404713)

[Pre-Requisites 7](#_Toc471404714)

[Steps to install Java 1.5 9](#_Toc471404715)

[Steps to install Ant 9](#_Toc471404716)

[Steps to install Tomcat 9](#_Toc471404717)

[Building the Java Web Interface Server 10](#_Toc471404718)

[Configuration Setup 10](#_Toc471404719)

[Compiling Source Files 16](#_Toc471404720)

[Building ACE-TAO Libraries 20](#_Toc471404721)

[Building ICU Libraries 26](#_Toc471404722)

[Installing Python 31](#_Toc471404723)

[Building the Engine Application 34](#_Toc471404724)

[Web Server 38](#_Toc471404725)

[VA-Engine 38](#_Toc471404726)

[VA-EngineFactory 39](#_Toc471404727)

**TABLE of Figures**

[Figure 1 : Password Generation 15](#_Toc471404728)

[Figure 2: Select a Workspace 16](#_Toc471404729)

[Figure 3: Create new JAVA Project 17](#_Toc471404730)

[Figure 4: Specify the Project Name 17](#_Toc471404731)

[Figure 5: Overview of the JAVA workspace 18](#_Toc471404732)

[Figure 6: Ant view for Compilation 18](#_Toc471404733)

[Figure 7: Preparation for compilation 19](#_Toc471404734)

[Figure 8: Compilation Status 19](#_Toc471404735)

[Figure 9: Winzip option to extract the ACE TAO libraries 20](#_Toc471404736)

[Figure 10: Window showing Environment Variables 21](#_Toc471404737)

[Figure 11: Window showing Environment Variables 21](#_Toc471404738)

[Figure 12: Window showing Directories tab after adding include directories for ACE TAO 22](#_Toc471404739)

[Figure 13: ACE Workspace 24](#_Toc471404740)

[Figure 14: Batch build option for ACE 25](#_Toc471404741)

[Figure 15: Selection of Debug/Release Option for ACE 25](#_Toc471404742)

[Figure 16: ICU Workspace 27](#_Toc471404743)

[Figure 17: Active Project Selection for ICU 28](#_Toc471404744)

[Figure 18: Building ICU 30](#_Toc471404745)

[Figure 19: Installing Python – Screen 1 31](#_Toc471404746)

[Figure 20: Installing Python – Screen 2 32](#_Toc471404747)

[Figure 21: Installing Python – Screen 1 33](#_Toc471404748)

[Figure 22: Installing Python – Screen 2 33](#_Toc471404749)

[Figure 23: The Engine Application workspace 35](#_Toc471404750)

[Figure 24: Selection of Active Configuration for the Engine Application Components 36](#_Toc471404751)

[Figure 25: Build KBFileReadWrite DLL 37](#_Toc471404752)

[Figure 26: Folder Structure of VA-Engine Deployment package. 39](#_Toc471404753)

[Figure 27: Folder Structure of VA-EngineFactory Deployment package. 40](#_Toc471404754)

**TABLES**

[Table 1: Nodes of “Web\_Config.xml” 12](#_Toc471404755)

[Table 2: Nodes of “web.xml” 13](#_Toc471404756)

[Table 3: Nodes of “tomcat-users.xml” 14](#_Toc471404757)

1. Introduction

The VA-Engine application being developed for Creative Virtual consists of the following 2 major components:

* **Web-interface server**

The Web-interface-server has been developed in JAVA and is based on the Tomcat Server version 6.0.

* **Engine Application**

The Engine application has been developed in VC++ 6.0 and has been migrated to Visual Studio 2008. The Engine application makes use of the ACE TAO and ICU libraries.

## Purpose

This document provides step-by-step instructions on how to build the VA-Engine application (JAVA and C++) source code. This document also details the build procedure for the third party source code: ACE TAO and ICU. **Please note this build also includes the built version of the ICU libraries.**

## Scope

The scope of this document is to provide instructions on how to compile the source code and build the V-Engine 64bit application. It does not provide information about running the application. The scope is limited to detail the build process on a machine with Windows Operating system.

1. Assumptions

## Pre-Requisites

The following are the pre-requisite’s to build **JAVA Web-interface server** Source code:

1. **Java 1.5** can be downloaded from the following location:

<http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/VerifyItem-Start/jdk-1_5_0_14-windows-i586-p.exe?BundledLineItemUUID=hMVIBe.p9lMAAAEk6XUel3F7&OrderID=WCdIBe.pcpwAAAEkynUel3F7&ProductID=MVjACUFBSfwAAAEYkqk5AXuQ&FileName=/jdk-1_5_0_14-windows-i586-p.exe>

1. **Ant 1.7.1** can be downloaded from the following location: <http://mirror.nyi.net/apache/ant/binaries/apache-ant-1.7.1-bin.zip>
2. **JacORB 2.3.1** can be downloaded from the following location**:** <http://www.jacorb.org/releases/2.3.1/jacorb-2.3.1-src.zip>. Add the following jar files in the libs folder of the project

* idl.jar
* jacorb.jar
* logkit-1.2.jar

1. **Tomcat 6.0.18** which can be downloaded from the following location:

<http://www.ecoficial.com/apachemirror/tomcat/tomcat-6/v6.0.18/bin/apache-tomcat-6.0.18.zip>

* servlet-api.jar
* jsp-api.jar

1. **xalan2.7.1.jar** which can be downloaded from the following location:

<http://www.findjar.com/jar/mule/dependencies/maven2/xalan/xalan/2.7.1/xalan-2.7.1.jar.html>

1. **Jibx1.2.1** which can be downloaded from the following location:

<http://sourceforge.net/projects/jibx/files/jibx/jibx-1.2.1/jibx_1_2_1.zip/download>

* jibx-bind.jar
* jibx-run.jar
* jibx-schema.jar
* jibx-tools.jar
* bcel.jar
* xpp.jar

1. **JavaMail 1.4.2** which can be downloaded from the following location:

<https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=javamail-1.4.2-oth-JPR@CDS-CDS_Developer>

* mail.jar

1. **JAF1.1.1** which can be downloaded from the following location:

<https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jaf-1.1.1-fcs-oth-JPR@CDS-CDS_Developer>

* activation.jar

1. **log4j-1.2.15** which can be downloaded from the following location:

<http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.15/apache-log4j-1.2.15.zip>

* log4j-1.2.15.jar

**Note**: The jar files mentioned in point nos. [5 to 9] have been provided with the source code.

The following are the pre-requisites to build **C++ Engine Application** Source code:

1. **Visual Studio 2008** which includes **Microsoft Visual C++ 2008**, Service Pack 6 should be installed on the machine where the application is going to be built.
2. **ACE TAO version 1.4a** source code package. This should be downloaded from the following location: <http://download.ociweb.com/TAO-1.4a/ACE+TAO-1.4a_with_latest_patches.zip>
3. **ICU version 4.2.1** source code package. This should be downloaded from the following location: <http://download.icu-project.org/files/icu4c/4.2.1/icu4c-4_2_1-src.zip>
4. **Python**
5. **Pre V-Engine v2.2.1**

**Python version 2.6.2** 64 bit Windows installer package. This should be downloaded from following location: <https://www.python.org/ftp/python/2.6.2/python-2.6.2.amd64.msi>

1. **V-Engine v2.2.1 onwards**

**Python version 2.7.12** 64 bit Windows installer package. This should be downloaded from following location: <https://www.python.org/ftp/python/2.7.12/python-2.7.12.amd64.msi>

**Please note that the source code will not build without the above mentioned pre-requisites.**

1. Building the java web-interface server

## Steps to install Java 1.5

1. Java 1.5.0\_14 can be downloaded from the following location: <https://dct.sun.com/dct/forms/reg_us_0809_958_0.jsp>

Follow the instructions provided with the download in order to install Java.

1. There are some **System Environment Variables** which need to be set. This can be done in the following manner in a Windows XP Operating system environment:

Navigate to ***Start -> Control Panel -> System -> Advanced -> Environment Variables***

Window shown in the figure [below](#Figure_EnvironmentVars) will pop up.

The following System variables along with their values need to be added. If already present, existing variables should be edited by adding these values using “;” as a separator.

|  |  |
| --- | --- |
| System Variable Name | Value |
| JAVA\_HOME | C:\Program Files\Java\jdk1.5.0\_14 |
| PATH | %JAVA\_HOME%\bin |

## Steps to install Ant

1. Ant 1.7.1 can be downloaded from the following location: <http://mirror.nyi.net/apache/ant/binaries/apache-ant-1.7.1-bin.zip>. Extract the downloaded zip file to the following location: “*D:\apache-ant-1.7.1*”
2. There are some **System Environment Variables** which need to be set. This can be done in the following manner in a Windows XP Operating system environment:

Navigate to ***Start -> Control Panel -> System -> Advanced -> Environment Variables***

Window shown in the figure [below](#Figure_EnvironmentVars) will pop up.

The following System variables along with their values need to be added. If already present, existing variables should be edited by adding these values using “;” as a separator.

|  |  |
| --- | --- |
| System Variable Name | Value |
| ANT\_HOME | D:\apache-ant-1.7.1 |
| PATH | %ANT\_HOME%\bin |

## Steps to install Tomcat

1. Tomcat 6.0.18 can be downloaded from the following location:

<http://www.ecoficial.com/apachemirror/tomcat/tomcat-6/v6.0.18/bin/apache-tomcat-6.0.18.zip>

Follow the instructions provided with the download in order to install Tomcat.

1. There are some **System Environment Variables** which need to be set. This can be done in the following manner in a Windows XP Operating system environment:

Navigate to ***Start -> Control Panel -> System -> Advanced -> Environment Variables***

Window shown in the figure [below](#Figure_EnvironmentVars) will pop up.

The following System variables along with their values need to be added. If already present, existing variables should be edited by adding these values using “;” as a separator.

|  |  |
| --- | --- |
| System Variable Name | Value |
| TOMCAT\_HOME | D:\apache-tomcat-6.0.18 |
| PATH | %TOMCAT\_HOME%\bin |

1. To change Tomcat’s default ORB to JacORB, the following needs to be incorporated.
2. Create a directory named endorsed in the tomcat installation folder

e.g. D:\apache-tomcat-6.0.18\endorsed

1. Add all the jar files present in the lib directory of the JacORB 2.3.1 download in this “endorsed” directory.

**Please note, JacORB 2.3.1 can be downloaded from the location mentioned point 3 of** [**section 2.1**](#_Pre-Requisites)**.**

## Building the Java Web Interface Server

### Configuration Setup

Perform the following steps to setup the configuration required by the web application:

1. The web configuration files that need to be configured for the application include:
2. ..\UILayer\config\Web\_Config.xml
3. ..\UILayer\web.xml
4. ..\UILayer\config\log4j.xml
5. ..\UILayer\config\VAProperties.properties
6. ..\ apache-tomcat-6.0.18\conf\tomcat-users.xml
7. ..\ apache-tomcat-6.0.18\conf\server.xml
8. Edit the contents of **Web\_Config.xml** as per the description given below.

|  |  |  |
| --- | --- | --- |
| Node Name | Attribute Name | Description |
| <Web\_Config/Domain> | - | This would contain the domain name of the server.  E.g. mastek.com |
| <ConnectionInfo/NameServiceAddress> | - | This would contain the address of the Naming Service for connection with the engine factory.  E.g. corbaloc::172.16.232.172:9990/NameService |
| <ConnectionInfo/EngineFactoryBindName> | - | This would contain the path binding the engine factory to the name service.  E.g. CreativeVirtual/EngineFactory |
| <URLs/URL> | name | This would contain the name of the URL to be displayed to the user.  E.g. CLOSED\_URL |
| path | This would contain the path of the web page corresponding to the mentioned URL name present on the tomcat server.  E.g. vaclosed.html |
| <EngineFailure/Email> | id | This would contain the email id of the recipient  E.g. tom@creativevirtual.com |
| name | This would contain the name of the recipient  E.g. Tom |
| <SMTPInfo/SMTPServer> | - | This would contain the IP address of the SMPT server for sending emails  E.g. 172.16.232.172 |
| <SMTPInfo/SMTPPort> | - | This would contain the port number of the SMTP server  E.g. 25 |
| <SMTPInfo/SMTPFrom> | - | This would contain the email sender’s name  E.g. nick@creativevirtual.com |
| <Default/Allow> | - | This would contain the list of IP Addresses which would be granted access to the application  E.g. 127.0.0.1, 172.16.232.179 |
| <Default/Deny> | - | This would contain the list of IP Addresses which would be denied access to the application  E.g. 127.0.0.1, 172.16.232.171/178 |
| <IPRestrictions/ReqMode> | type | This would contain the user’s request mode.  E.g. normal, admin, trigger, state |
| <ReqMode/Allow> | - | This would contain the list of IP Addresses which would be granted access to the application for the specific request mode type specified in the ‘type’ attribute  E.g. 127.0.0.1, 172.16.232.181 |
| <ReqMode/Deny> | - | This would contain the list of IP Addresses which would be denied access to the application for the specific request mode type specified in the ‘type’ attribute  E.g. 127.0.0.1, 172.16.232.180 |

Table 1: Nodes of “Web\_Config.xml”

Sample “*Web\_Config.xml”* is given below:



1. Edit the contents of web.xml as per the description given below.

|  |  |  |
| --- | --- | --- |
| Node Name | Attribute Name | Description |
| <servlet/init-param/param-value> | - | This would contain the module id which would be the same as the module id of the V-Engine that the web application would be connecting with, present in the EngineFactory.xml.  E.g.  <init-param>  <param-name>ClientDetails  </param-name>  <param-value> **CreativeVirtual**  </param-value>  </init-param>  **Note:** The <param-value> **CreativeVirtual**  </param-value>  needs to be changed in both the existing instances. |
| <security-role\role-name> | - | Role Names authorized to access the web application.  Eg.  <security-role>  <role-name>admin</role-name>  </security-role> |
| <security-constraint/auth-constraint/role-name> | - | Role Name authorized to access the administrator application. This role-name should be present in tomcat-users.xml [Ref 5]  Eg.  <auth-constraint>  <role-name>admin</role-name>  </auth-constraint> |

Table 2: Nodes of “web.xml”

Also please note for caching related support, please check and follow the instructions mentioned in the attached web.xml file.

Sample “w*eb.xml”* is given below:



1. Edit the log4j.xml file present in config directory of the application to change the name of the log file to make it client specific.

E.g. ../UILayer/config/log4j.xml

To change the name of the log file generated to BtDiagnostic, change the file name in log4j.xml as given

<appender name="VA\_ROLLING\_FILE" class="org.apache.log4j.DailyRollingFileAppender">

<param name="File" value="***../logs/BtDiagnostic.log***" />

1. Edit the VAProperties.properties present in config directory of the application to set the server type as follows:

**server.type=TOMCAT**

Comment out the entry “**server.type=IBM”**

1. Add the following contents in tomcat-users.xml to configure users to access the Administrator application.

Eg.

<tomcat-users>

<role rolename="admin"/>

<user username="admin" password="password" roles="admin"/>

</tomcat-users>

|  |  |  |
| --- | --- | --- |
| Node Name | Attribute Name | Description |
| <role> | rolename | This would be the name of the role which would be authorised to access the administrator application |
| <user> | username | This would be the username of the user intending to access the administrator application |
| password | This would be the password of the user accessing the administrator application |
| roles | This would be the role name for the user |

Table 3: Nodes of “tomcat-users.xml”

Mulitple roles and multiple users can be configured as shown below:

Eg.

<role rolename="manager"/>

<role rolename="admin"/>

<user username="manager" password="manager" roles="manager"/>

<user username="admin" password="admin" roles="admin"/>

Sample “*tomcat-users.xml”* is given below:



**NOTE:**

To add the password in an encrypted format in tomcat-users.xml, do as given below

1. Run the digest batch file present in bin directory of the tomcat installation directory

Eg. D:\apache-tomcat-6.0.18\bin\digest.bat

* + Open the command prompt and enter the directory path of the digest.bat file

Eg. cd D:\apache-tomcat-6.0.18\bin

* + Type the command ***digest –a*** (algorithm) ***algorithm name*** ***password value***

Eg. digest –a sha password

***sha*** – Name of the algorithm

***password value*** – password

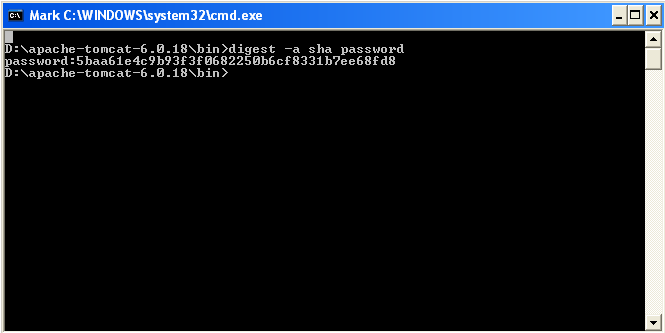


Figure 1 : Password Generation

1. The encrypted password value generated as shown in the Figure 1 can then be added in the password attribute of tomcat-users.xml.
2. Edit the contents of server.xml as given below only if the password being added in tomcat-users.xml is in an encrypted format.

For the Realm with className and resource name as given below

Eg.

<Realm className="org.apache.catalina.realm.UserDatabaseRealm"

resourceName="UserDatabase"/>

Add the name of the algorithm that you would be using to encrypt the passwords.

Eg.

<Realm className="org.apache.catalina.realm.UserDatabaseRealm"

***digest="sha"***

resourceName="UserDatabase"/>

### Compiling Source Files

There are two ways in which the Java source can be built:

**Method 1: Using command line**

* To compile the source files, open a command prompt and go to the home directory of the project.
* E.g. cd D:\CreativeVirtual\UILayer
* Type the command ‘ant UI-War’
* This will start the compilation of the java source files and a cv.war file will be created in the build directory
* E.g. D:\CreativeVirtual\UILayer\build\cv.war

**Method 2: Using Eclipse 3.4**

* Download eclipse 3.4(Eclipse IDE for Java EE Developers) from the following location: <http://build.eclipse.org/technology/phoenix/torrents/jee/eclipse-jee-galileo-SR1-win32.zip.torrent>
* Click on the eclipse icon to start the IDE
* When you start Eclipse, Eclipse will ask you to specify the workspace (a directory used by Eclipse to store java programs) to use
* Specify an existing directory where your java program is stored as the workspace.

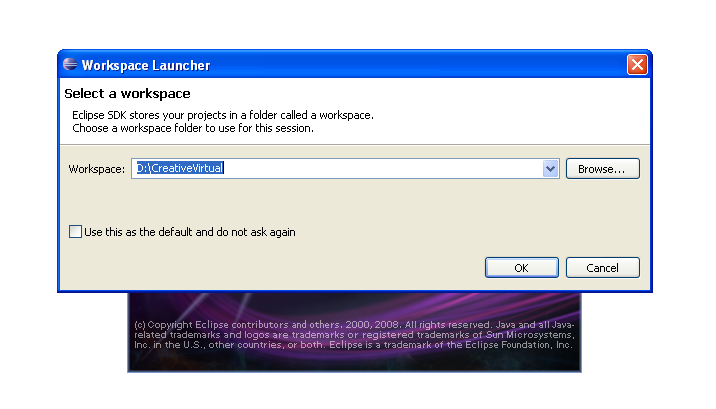


Figure 2: Select a Workspace

* Once you have started eclipse, click on the Java Project.

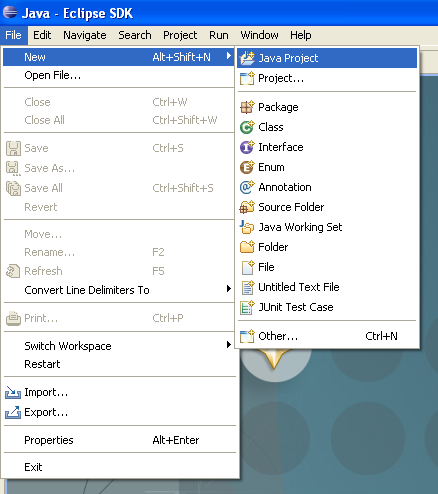


Figure 3: Create new JAVA Project

* Eclipse will further prompt you for the name of the project.
* If the project already exists in the workspace mentioned while initializing eclipse, add the name of the existing project.
* If the project exists at some other location, select the Create project from existing source and use the browse button to select the source of the project.
* Click on Finish to complete this task.

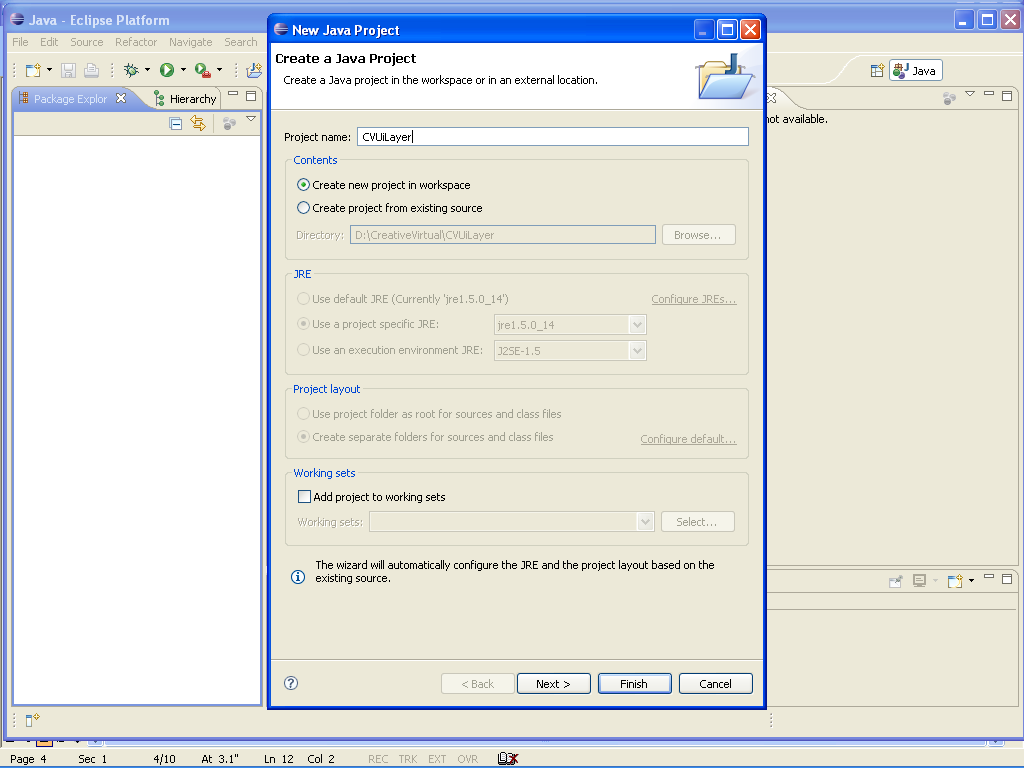


Figure 4: Specify the Project Name

* Eclipse will display the selected project as shown below

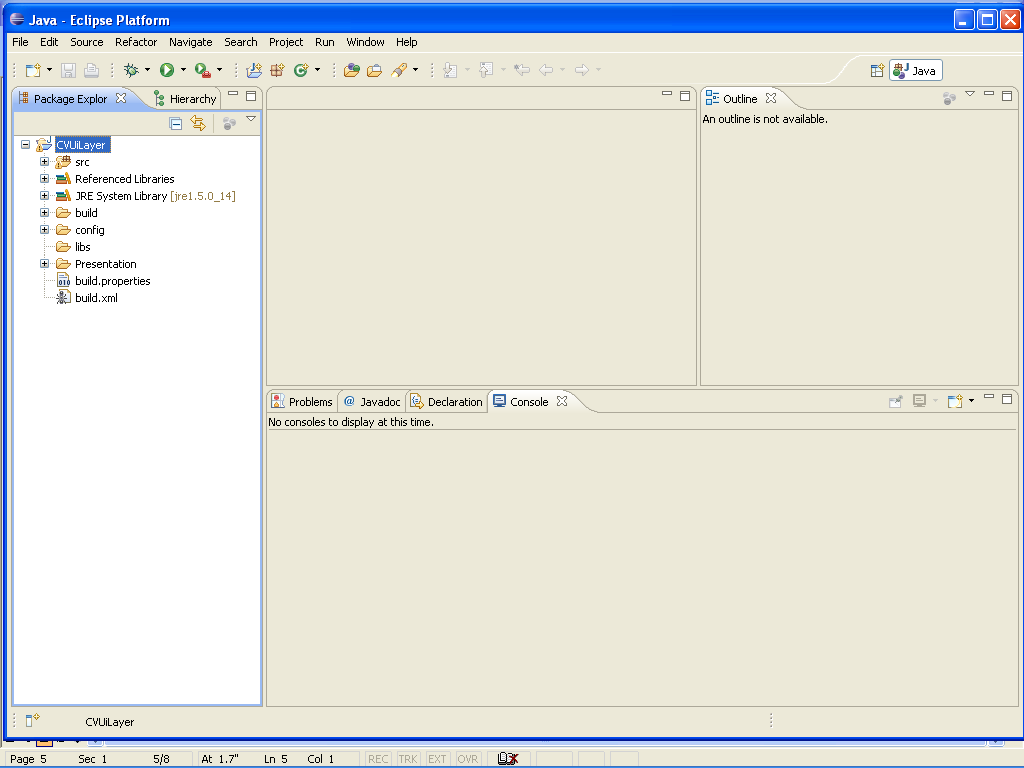


Figure 5: Overview of the JAVA workspace

* To compile the source files of the project, select the Ant View as shown below

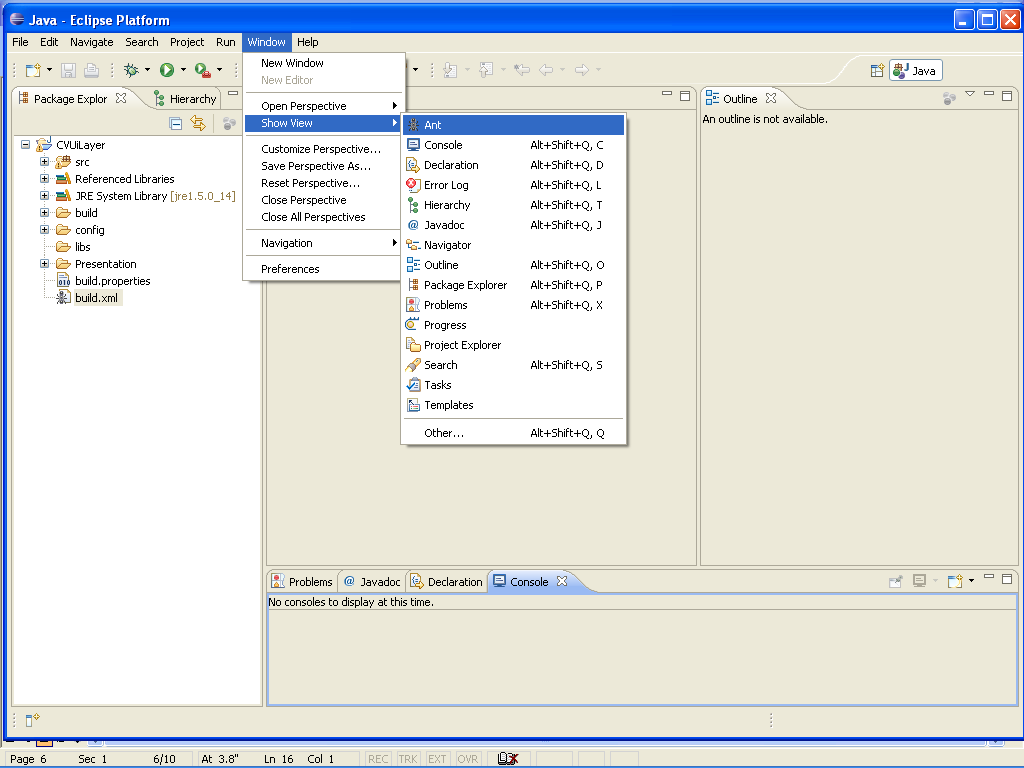


Figure 6: Ant view for Compilation

* Drag the build.xml present in the project to the Ant window as shown below:

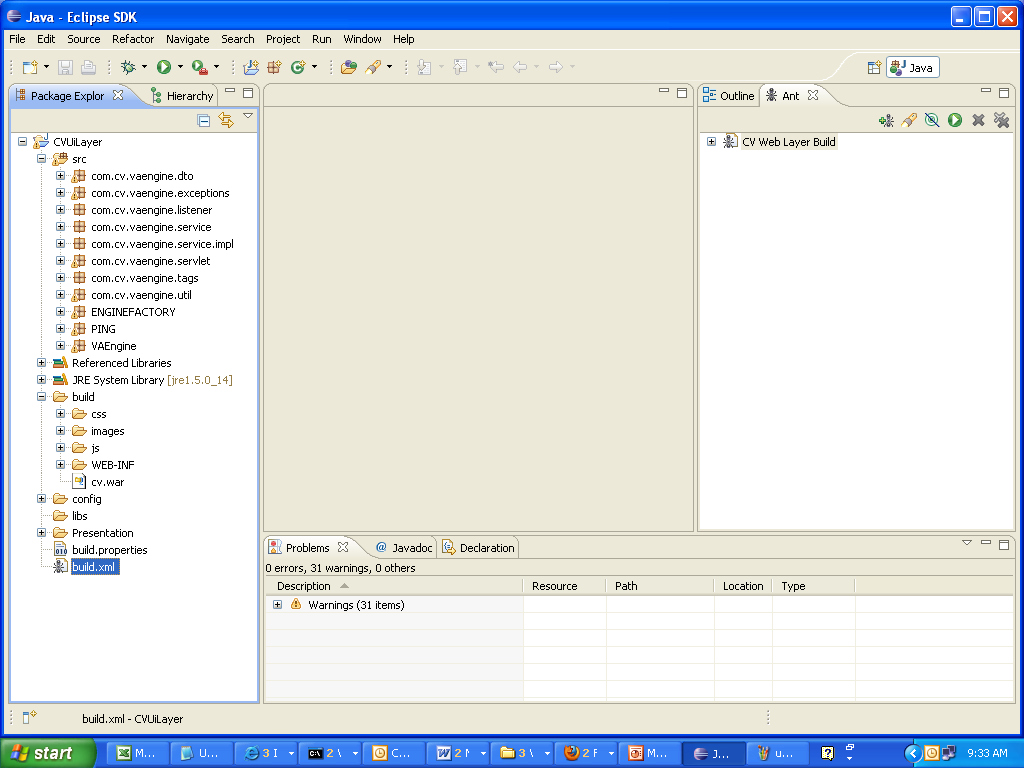


Figure 7: Preparation for compilation

* Double click on the Ant task or expand the same and click on the task that reads [default], to start compiling the source files of the project.

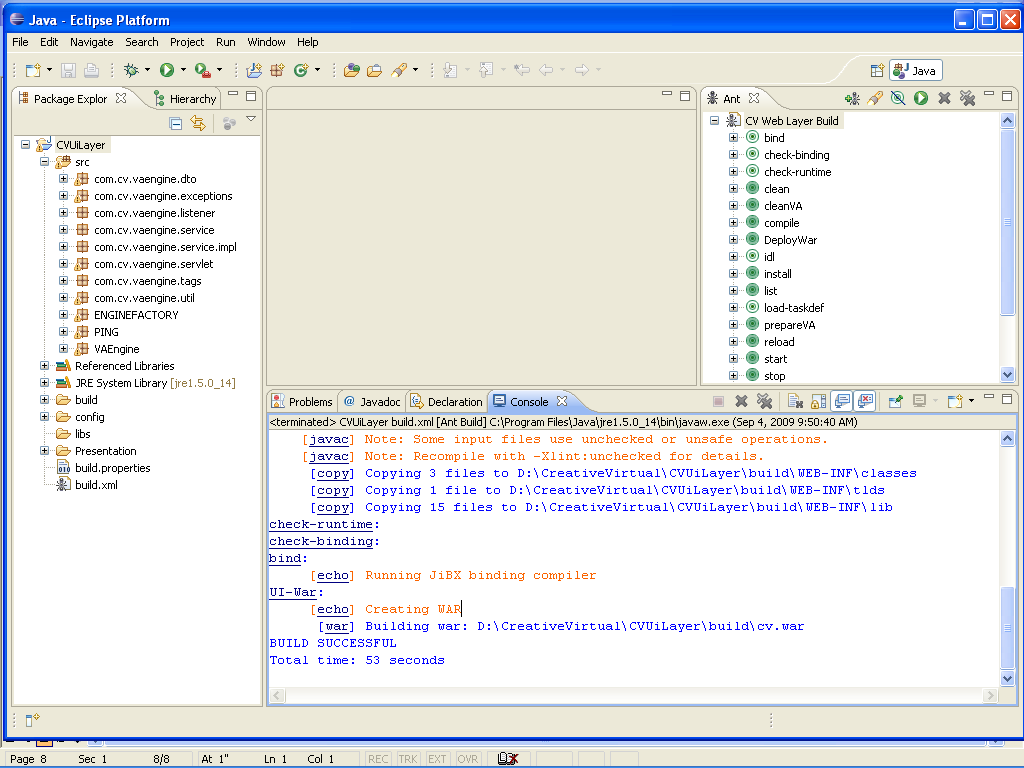


Figure 8: Compilation Status

1. Building the c++ engine application

As mentioned earlier, the C++ engine application makes use of the ACE TAO, ICU and Python libraries. Before the engine application is built, it is necessary to build the ACE TAO libraries and to install Python. The build process for the same has been detailed in the following sections.

## Building ACE-TAO Libraries

The following steps should be performed to build the ACE-TAO libraries:

1. **ACE TAO version 1.4a** source code package should be **downloaded** from the following location: <http://download.ociweb.com/TAO-1.4a/ACE+TAO-1.4a_with_latest_patches.zip>
2. Once the file has been downloaded, copy it to the D:\ drive of the machine on which the engine application has to be built. Use the “***Extract to here***” option in **winzip** to extract the source files. The selected option has been shown in the following figure:

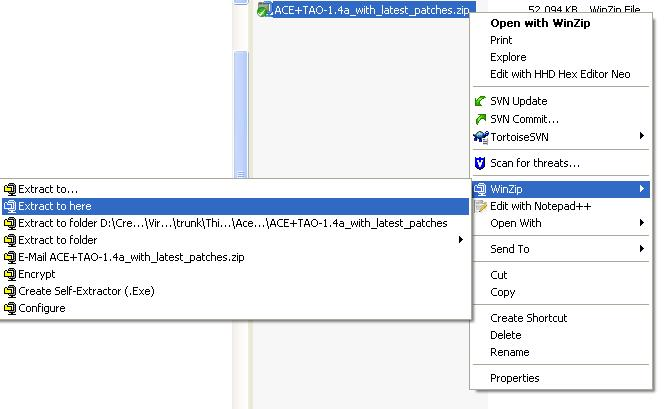


Figure 9: Winzip option to extract the ACE TAO libraries

After the unzipping process has completed, a folder *“****ACE\_wrappers****”* is created in the D:\ drive.

1. There are some **System Environment Variables** which need to be set. This can be done in the following manner in a Windows 7 64bit Operating system environment:

Navigate to ***Start -> Control Panel -> System -> Advanced -> Environment Variables***

The following window will pop up:

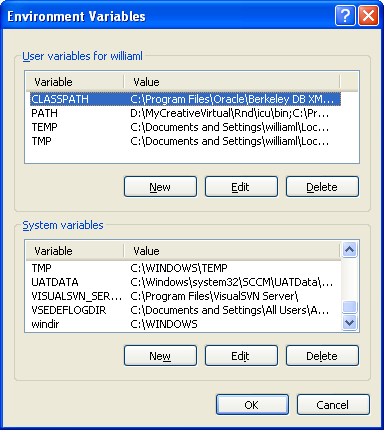


Figure 10: Window showing Environment Variables

The following System variables along with their values need to be added:

|  |  |
| --- | --- |
| System Variable Name | Value |
| ACE\_ROOT | D:\ACE\_wrappers |
| TAO\_ROOT | %ACE\_ROOT%\TAO |
| PATH | %ACE\_ROOT%\bin;  %TAO\_ROOT%\orbsvcs\orbsvcs;  %ACE\_ROOT%\lib; |

To add a system variable click on ***New*** in the System Variables section and add variables as shown in the following figure:

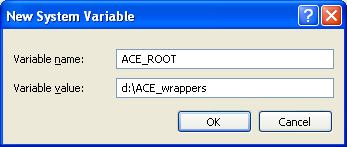


Figure 11: Window showing Environment Variables

If a variable already exists, then “***Edit***” the value by choosing the edit option and add the variable value by using “;” as a separator.

1. Visual Studio 2008 needs to be setup for compiling ACE TAO as well as project using ACE TAO. The following settings need to be done:

Start Microsoft Visual C++ 2008 and navigate to ***Tools -> Options –> Projects and Solutions->VC++Directories***

In the “***Directories”*** tab, select the platform as ***x64***.

Next, select “***Include files***” in the dropdown “***Show directories for***” and add the following 3 Directories:

*D:\ACE\_wrappers*

*D:\ACE\_wrappers\TAO*

*D:\ACE\_wrappers\TAO\orbsvcs*

After adding these directories, the “***Directories”*** tab will look as shown in the following figure:

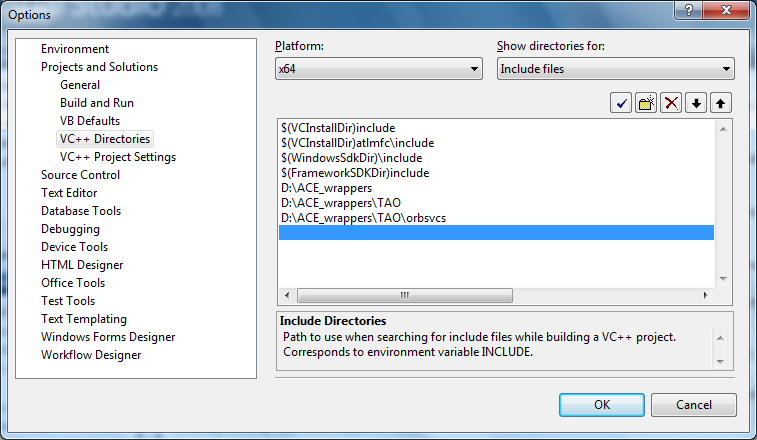
****

Figure 12: Window showing Directories tab after adding include directories for ACE TAO

In a similar manner, the following directory paths need to be added under “***Executable files***” and “***Library files***”:

**Executable files:**

*D:\ACE\_wrappers\bin*

**Library files:**

*D:\ACE\_wrappers\ace*

*D:\ACE\_wrappers\TAO\tao*

*D:\ACE\_wrappers\TAO\orbsvcs\orbsvcs*

*D:\ACE\_wrappers\lib*

1. Next, navigate to the folder “*D:\ACE\_wrappers\ace*”. Create a file “config.h” and add the following content into it:

#include "ace/config-win32.h"

Or simply copy the following object as “*config.h*” into the folder “*D:\ACE\_wrappers\ace”*



Copy the following file into the folder “D:\ACE\_wrappers\ace”



1. Open the workspace file “*D:\ACE\_wrappers\ace\ace\_vc8.sln*” in Microsoft Visual C++ 2008. On opening the ace\_vc8.sln file, it will ask to convert the solution from vc8 to vc9. Click next to proceed and finally click on finish button to convert the solution. On opening the file, the following workspace will be visible:

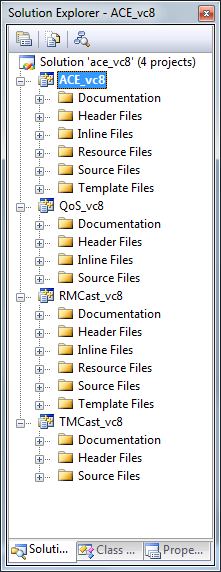


Figure 13: ACE Workspace

1. To build the application click, on the “Batch Build” option under the “Build” menu as shown in the following figure:

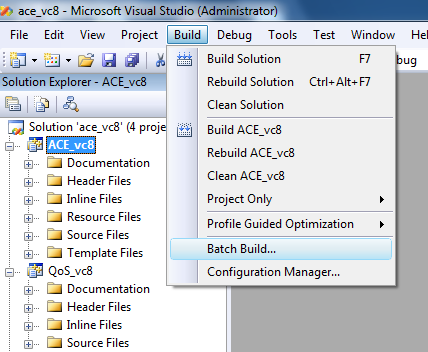


Figure 14: Batch build option for ACE

On clicking “***Batch Build***” the following window will pop up:

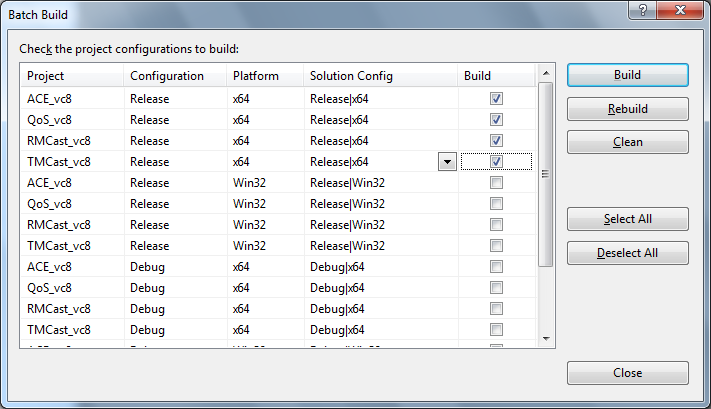


Figure 15: Selection of Debug/Release Option for ACE

Depending on which build is required, Debug or Release may be selected. Also, remember to select the x64 Platform for 64bit build. Please note that both Debug and Release libraries can be built together.

Click on ***Build*** to build ACE.

1. In a similar manner to build TAO, Open the workspace file “*D:\ACE\_wrappers\TAO\* *TAOACE\_vc8.sln*” in Microsoft Visual C++ 2008 and “***Batch Build***” all the libraries as explained in steps 6 and 7. Repeat this till there are no errors.

## Building ICU Libraries

To Build ICU the following steps need to be followed:

The ICU libraries help in making the engine application UNICODE compliant. The following steps should be performed to build the ICU libraries:

1. **ICU version 4.2.1** source code package should be **downloaded** from the following location: <http://download.icu-project.org/files/icu4c/4.2.1/icu4c-4_2_1-src.zip>
2. Once the file has been downloaded, copy it to the folder “***..\VirtualAssistant\VAEngine \ThirdParty***” of the V-Engine Application source code. **Please note that this step is very important and if not done will result in a build failure of the engine application**. If “ThirdParty\icu” folder already exists then delete “icu” folder. Use the “***Extract to here***” option in **winzip** to extract the source files. A new folder “..***\VAEngine\ThirdParty\icu***” will be created and the code will be extracted to it.
3. Open the following workspace file in Microsoft Visual C++ 2008: “..\VirtualAssistant*\VAEngine\ThirdParty\icu\source\allinone\* *allinone.sln*”. On opening the file, the following workspace will be visible:

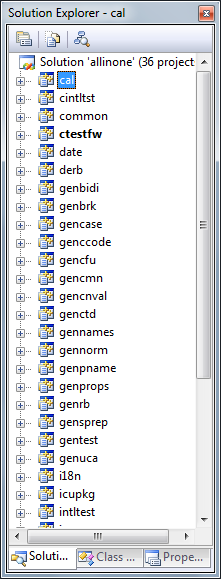


Figure 16: ICU Workspace

1. To build the application, first the active configuration needs to be selected. This can be done by navigating to the ***Build -> Configuration Manager...*** option in Microsoft Visual C++ 2008. The following options are displayed:

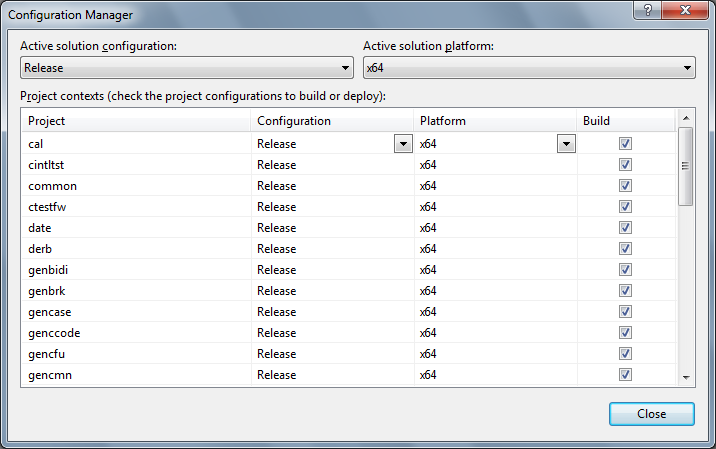
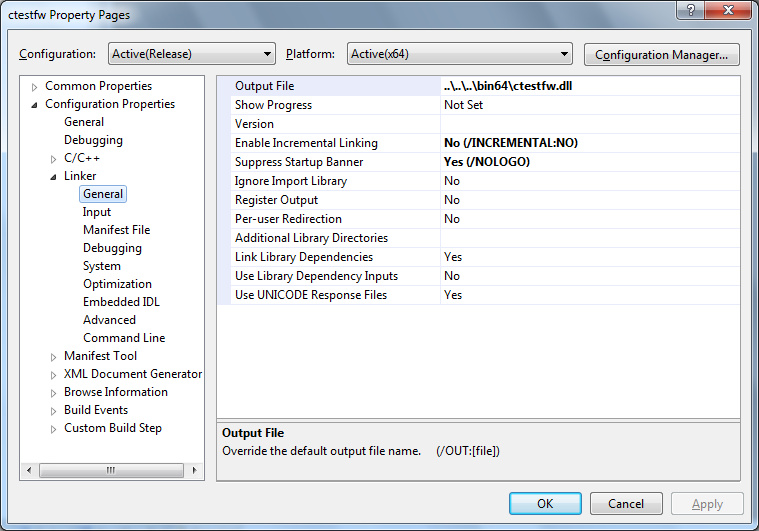


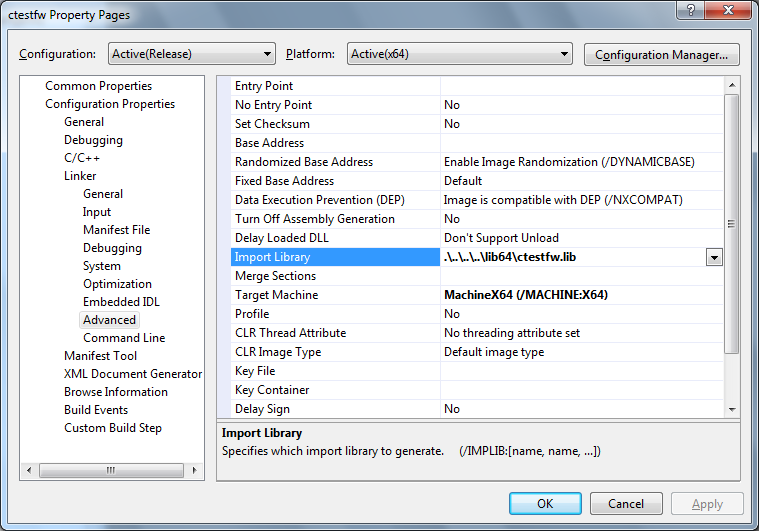
Figure 17: Active Project Selection for ICU

It is recommended that both the builds are created so that the engine application can be executed in both release as well as debug modes. Also please select the x64 Platform for 64 bit build.

Right Click ctestfw project in solution explorer and select properties. Change Linker->General->Output file from “..\..\..\bin\icutestd.dll” to “..\..\..\bin64\ctestfw.dll” as shown below.



Similarly again Right Click ctestfw project in solution explorer and select properties. Change Linker->Advanced->Import Library from “.\..\..\..\lib\icutestd.lib” to “.\..\..\..\lib64\ctestfw.lib” as shown below,



1. Once the active project configuration is selected, Build the libraries using the Build option on the Build menu as shown in the following figure:

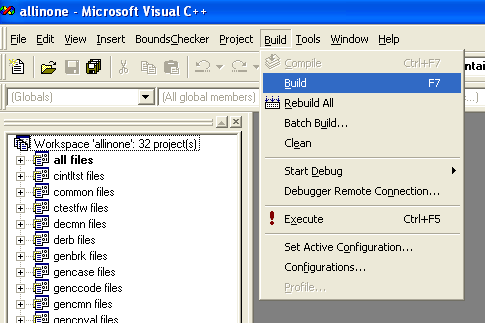
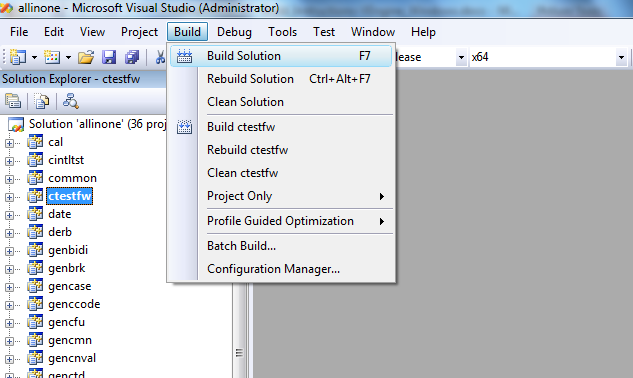


Figure 18: Building ICU

This will build the ICU libraries which will be used by the engine application. After build process is complete rename the .\ThirdParty\ICU\lib64 with ThirdParty\ICU\lib and .\ThirdParty\ICU\bin64 with .\ThirdParty\ICU\bin

1. There are some **System Environment Variables** which need to be set. This can be done in the following manner in a Windows XP Operating system environment:

Navigate to ***Start -> Control Panel -> System -> Advanced -> Environment Variables***

Window shown in the figure [below](#Figure_EnvironmentVars) will pop up.

The following System variables along with their values need to be added. If already present, existing variables should be edited by adding these values using “;” as a separator.

|  |  |
| --- | --- |
| System Variable Name | Value |
| PATH | <ICU\_Absolute\_Path>\bin; <ICU\_Absolute\_Path>\lib |

Note: Keyword <ICU\_Absolute\_Path> – refer to absolute path for folder “ThirdParty\icu”

E.g. D:\ThirdParty\icu

## Installing Python

The following steps should be performed to install the Python libraries:

1. **Pre V-Engine 2.2.1**
2. **Python version 2.6.2 amd 64bit** Windows installer package should be **downloaded** from the following location:

[**http://www.python.org/ftp/python/2.6.2/python-2.6.2.amd64.msi**](http://www.python.org/ftp/python/2.6.2/python-2.6.2.amd64.msi)

1. Once the file has been downloaded, install it in “C:\Python26” drive. Follow steps as shown in following images,

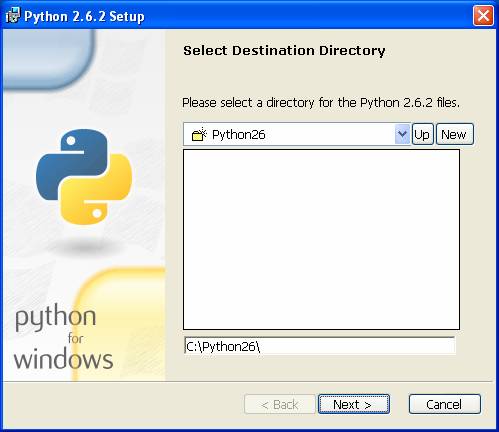


Figure 19: Installing Python – Screen 1

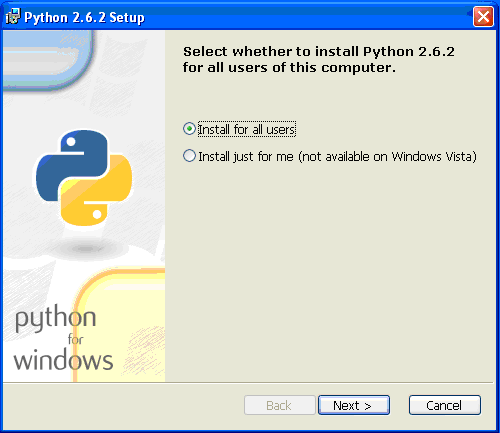


Figure 20: Installing Python – Screen 2

1. Once python is installed, delete the existing “pyconfig.h” file from the “C:\Python26\include” directory.
2. Copy the attached “*pyconfig.h*” file into the “C:\Python26\include” directory.



1. **V-Engine v2.2.1 onwards**
2. **Python version 2.7.12 amd 64bit** Windows installer package should be **downloaded** from the following location:

<https://www.python.org/ftp/python/2.7.12/python-2.7.12.amd64.msi>

1. Once the file has been downloaded, install it in “C:\Python27” drive. Follow steps as shown in following images,



Figure 21: Installing Python – Screen 1

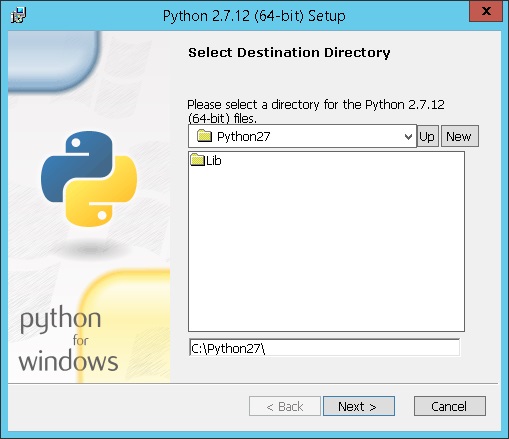


Figure 22: Installing Python – Screen 2

1. Once python is installed, delete the existing “pyconfig.h” file from the “C:\Python27\include” directory.
2. Copy the attached “*pyconfig.h*” file into the “C:\Python27\include” directory.



## Building the Engine Application

The C++ Engine workspace consists of the following components:

1. Hardware DLL
2. KBFileReadWrite DLL
3. Language DLL
4. VAEngine DLL
5. VAServer EXE
6. VAEngineFactory EXE
7. VATestEngine DLL

Please Note: Out of the above mentioned components only Language DLL, VAEngine DLL, VAServer EXE, VAEngineFactory EXE are required for VA Engine application.

The Microsoft Visual Studio project files for each of the above mentioned components have been included in a common Microsoft Visual Studio workspace file, “*Virtual Assistant.dsw*”. This file will be present in the folder “***..\VirtualAssistant\VAEngine\Virtual Assistant.sln***”of the V-Engine application source code. On opening this file in Microsoft Visual C++ 2008, the following files are displayed:

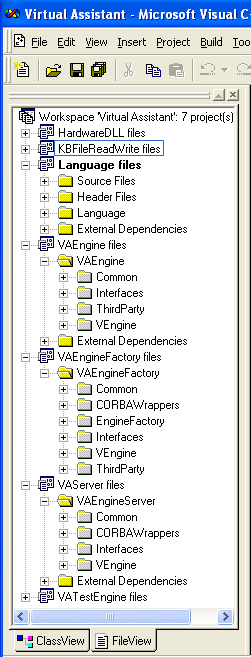
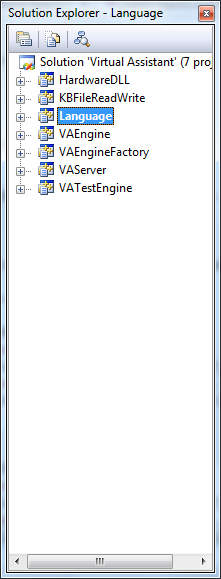


Figure 23: The Engine Application workspace

Build Language Dll:

* Select Solution configuration to Release x64 as shown below,

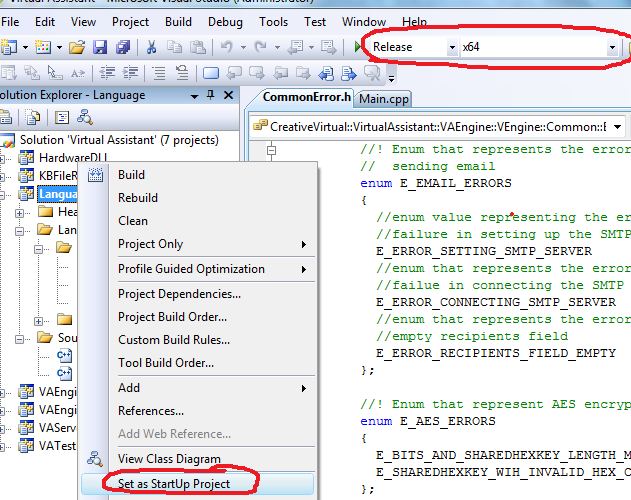


Figure 24: Selection of Active Configuration for the Engine Application Components

* Right Click Language Dll and set as startup project.
* Right Click Language Dll->Project Only->Build Only Language

The Debug\Release option may be selected as per requirement.

**Note: Projects “HardwareDLL-Win32 Release” and “VATestEngine-Win32 Release” do not require to be built.**

Build VAEngine Dll:

* Select Solution configuration to Release x64 as shown below,
* Right Click VAEngine and set as startup project.
* Right Click VAEngine->Project Only->Build Only VAEngine

Build VAEngineFactory Exe:

* Select Solution configuration to Release x64 as shown below,
* Right Click VAEngineFactory and set as startup project.
* Right Click VAEngineFactory->Project Only->Build Only VAEngineFactory

Build VAServer Exe:

* Select Solution configuration to Release x64 as shown below,
* Right Click VAServer and set as startup project.
* Right Click VAServer->Project Only->Build Only VAServer

Build KBFileReadWrite DLL:

* Select Solution configuration to Release x64 as shown below,
* Right Click on KBFileReadWrite project and Build

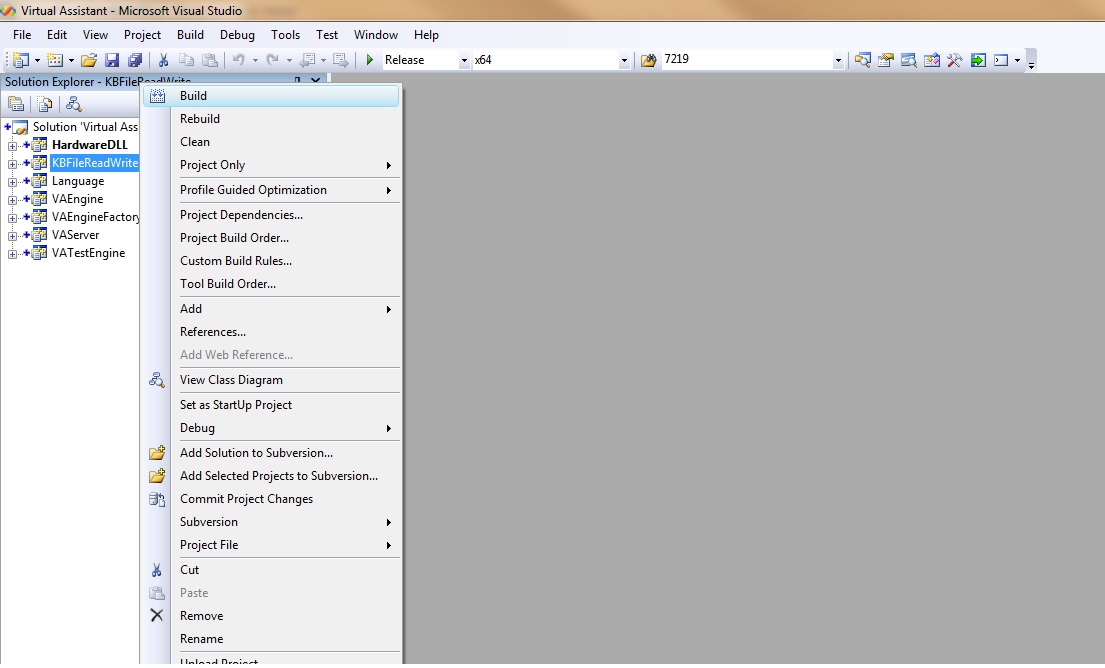


Figure 25: Build KBFileReadWrite DLL

Please note: The VAEngine DLL is dependent on the Langauge DLL and hence will not build until the Language DLL is not built. Also, a debug version of the VAEngine DLL requires the debug version of the Language DLL to be built. Similarly, a release version of the VAEngine DLL requires the release version of the Language DLL to be built. Hardware and VATestEngine are used by V-Builder. KBFileReadWrite DLL is used by V-Portal for macro/file encrypt/decrypt.

**Install Microsoft Visual C++ 2008 Redistributable Package:**

Install Microsoft Visual C++ 2008 redistributable package (x64) where V-Portal is deployed as KBFileReadWrite DLL will require it.

The Microsoft Visual C++ 2008 Redistributable Package (x64) installs runtime components of Visual C++ Libraries required to run 64-bit applications developed with Visual C++ on a computer that does not have Visual C++ 2008 installed.

The package can be downloaded from following link,

<http://download.microsoft.com/download/d/2/4/d242c3fb-da5a-4542-ad66-f9661d0a8d19/vcredist_x64.exe>

1. Packaging

### Web Server

Copy the generated war file present in the build directory of the UILayer for e.g \UILayer\build\bt.war for deploying the same on the production server.

### VA-Engine

This section explains how to create the package of VA-Engine (C++ end). This package contains required executable, configuration files and folders which are compulsory to run VA-Engine (VAServer.exe) on Windows machine without any dependency.

Perform following steps to create the package for V-Engine:

1. Ensure that VA-Engine application, ICU library and ACE library have built in release mode.
2. Copy the following “Deployment\_VEngine.bat” file to “D:” drive.



1. Open command prompt & change directory to “D:” drive.
2. Execute Deployment batch command as given below,

*Deployment\_VEngine.bat –t "Deployment Directory" –s "VA-Engine Root Directory" –a "ACE Root Directory"*

Where

1. Deployment Directory: It is directory where required executable, configuration files and folders are copied.
2. VA-Engine Root Directory: It is Root directory of VA-Engine code that is where “Virtual Assistant.dsw” file is located.
3. ACE Root Directory: It is Root directory of ACE that is where “ace” folder is located.

E.g.: *Deployment\_VEngine.bat –t "D:\VAEngine\_Deploy" -s "D:\MySVN" –a "D:\ACE\_wrappers"*

On success “*Deployment Directory*” will contain deployment package. Use commands given in section 3.1 and 3.3 ‘VirtualAssistant-InstallationInstructions-VAEngine.doc’ to run Naming Service and VA Server respectively from “Deployment Directory”.

Following image shows folder structure of VA-Engine Deployment package:

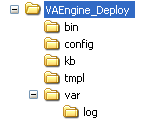


Figure 26: Folder Structure of VA-Engine Deployment package.

**Please Note: The License file required for running the VAServer application should be named as “License.lic” only. It should be placed within the *bin* folder of VAEngine\_Deploy.**

### VA-EngineFactory

This section explains how to create deployment package of VA-EngineFactory (C++ end). This package contains required executable, configuration files and folders which are compulsory to run VA-EngineFactory (VAEngineFactory.exe) on Windows machine without any dependency.

Perform following steps to create deployment package for V-Engine:

1. Ensure that VA-EngineFactory application, ICU library and ACE library have built in release mode.
2. Copy following “Deployment\_EngineFactory.bat” file to “D:” drive.



1. Open command prompt & change directory to “D:” drive.
2. Execute Deployment batch command as given below:

*Deployment\_EngineFactory.bat –t "Deployment Directory" –s "VA-Engine Root Directory" –a "ACE Root Directory"*

Where

1. Deployment Directory: It is directory where required executable, configuration files and folders are copied.
2. VA-Engine Root Directory: It is Root directory of VA-Engine code that is where “Virtual Assistant.dsw” file is located.
3. ACE Root Directory: It is Root directory of ACE that is where “ace” folder is located.

E.g.: *Deployment\_EngineFactory.bat –t "D:\VAEngineFactory\_Deploy" -s "D:\MySVN" –a "D:\ACE\_wrappers"*

On success “*Deployment Directory*” will contain deployment package. Use commands given in section 3.1 and 3.2 of ‘VirtualAssistant-InstallationInstructions-VAEngine.doc’ to run Naming Service and VA-EngineFactory respectively from “*Deployment Directory*”.

Manually copy following files from icudt42.dll, icuin42.dll, icuio42.dll, icuuc42.dll from \ThirdParty\ICU\bin\ to VAEngineFactory\_Deploy\bin and VAEngine\_Deploy\bin

Following image shows folder structure of VA-EngineFactory Deployment package.



Figure 27: Folder Structure of VA-EngineFactory Deployment package.