Tutorial: using Java to access the Vault API

Barbara Han
Autodesk Developer Technical Services

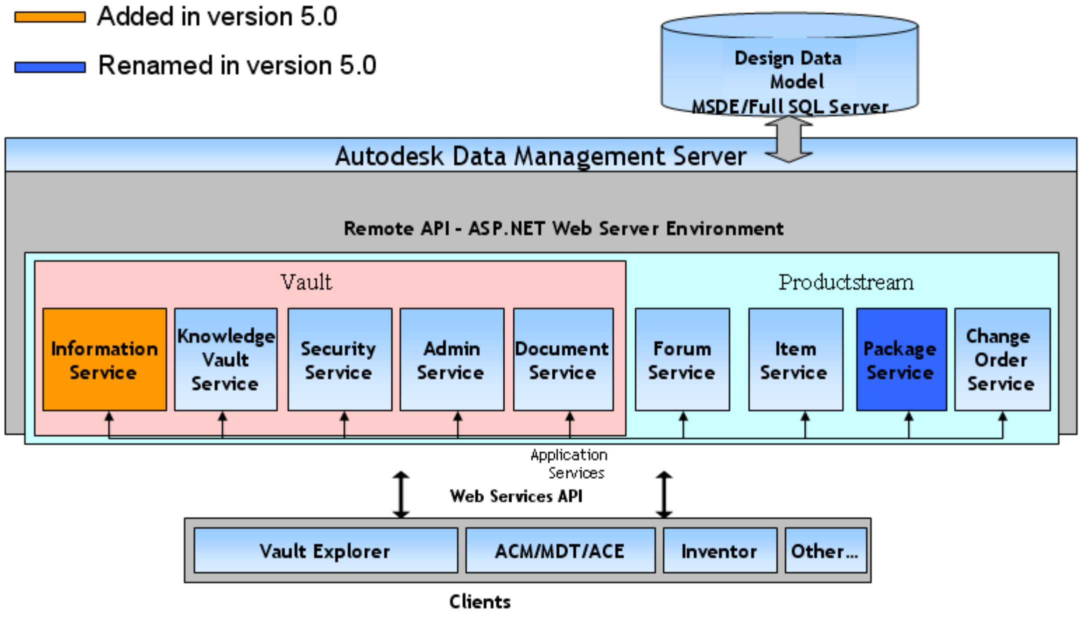
July 7th 2006

Table of contents

Introduction

Autodesk Vault provides its API through Web Services which are Web Service Interop (WS-I) compliant.

The following diagram illustrates the Vault/ProductStream API architecture.
Note: a full description of these web services is outside the scope of this article, and is covered in several other introductory documents available on the Autodesk Developer Network (ADN) website.



You can access the Vault API via C#, or VB.NET from Microsoft VisualStudio.Net.

The SDK is delivered with Vault Server Installer and within which there are comprehensive samples you can use as a reference.

You can also use Java to access Vault API. This article will explains how to get started with developing Autodesk Vault client applications using Java.

Solution Approach

The following points are common to both .NET and the Java development environments, when accessing the Vault API:

1. The Vault API services can be accessed using SOAP (Simple Object Access Protocol) which uses the document encoding style.
2. Files are exchanged as DIME (Direct Internet Message Encapsulation) attachments.

There are also some differences between these two environments that you need be aware of:

1. Before you can create a Java web service client application, you must first generate the client stubs for the Vault API web services. For C# or VB.NET, this step isn't necessary.
2. There isn't a SecurityHeaderValue property of the security service for SecurityHeader in generated client stubs (Java code), so we need work around this by adding a method that can return a SecurityHeader which is then used for other web service calls.
3. The data types may change in generated client stubs (Java code) when compared with the Vault/Productstream API help documentation.

This article discusses these issues in more detail later.

Prerequisites

These are the main prerequisites required to work through the examples provided in the article.

- **Autodesk Vault Server 5.0**
- **JBuilder 2006 (or any other IDE for Java)**
  - **Axis**
  - **J2EE**

Generating client stubs

You can use WSDL2Java to generate client stubs. JBuilder 2006 contains the Axis library (WSDL2Java), thus you can use it directly. If you use another IDE which doesn't provide the Axis library, you can download it from:-
http://www.apache.org/dyn/closer.cgi/ws/axis/1_4

Example 1: create the client stubs for the DocumentService -
The following illustrates how to do this:

```
set AXIS_LIB=C:/Borland/JBuilder2006/thirdparty/ws-axis/lib

C:\Borland\JBuilder2006\jdk1.5\bin\java -cp
"%AXIS_LIB%/wsdl4j-1.5.1.jar;%AXIS_LIB%/axis.jar;%AXIS_LIB%/commons-logging-1.0.4.jar
;%AXIS_LIB%/commons-discovery-0.2.jar;%AXIS_LIB%/jaxrpc.jar;%AXIS_LIB%/log4j-1.2.8.ja
r;%AXIS_LIB%/saaj.jar;%AXIS_LIB%/axis-ant.jar" org.apache.axis.wsdl.WSDL2Java -a -v -o
C:\Borland
http://localhost/autodeskDM/Services/DocumentService.asmx?WSDL
```

As you see, WSDL2Java takes the URL to the WSDL for the web service and the output directory as its input.

Please copy and paste above code into a .bat file then move the file to the folder: <JBuilder_installpath>\jdk1.5\bin\ and run it, this will generate all the Java stubs for the Document Service, which are then ready for use in the folder C:\Borland\AutodeskDM.

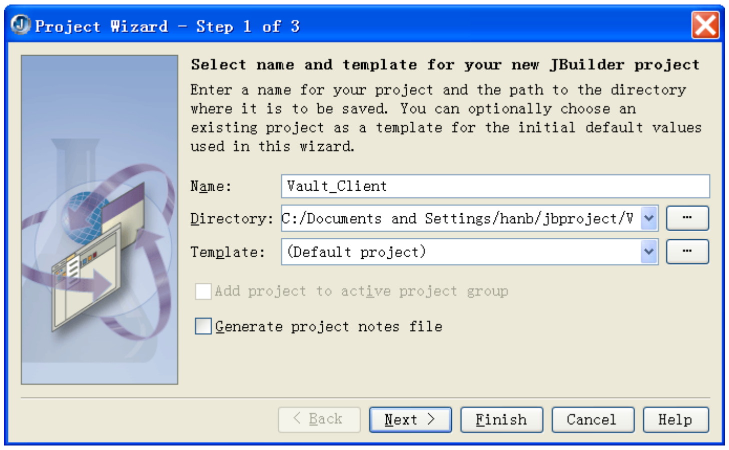
Example 2: create the client stubs for the SecurityService -
Just replace DocumentService.asmx with SecurityService.asmx in the above code -

http://localhost/autodeskDM/Services/SecurityService.asmx?WSDL

You can also use another JBuilder utility, Web Services Designer, to generate the client stubs for those Services as well. For more details, please refer to the topic "Importing a WSDL document in the JBuilder help documentation.

Client functionality implementation details

First, let's create a new JBuilder project from which to reference the Vault API within. Click File->New project..., set the project name "Vault_Client" in step 1 of the project Wizard Dialog, then click Next.



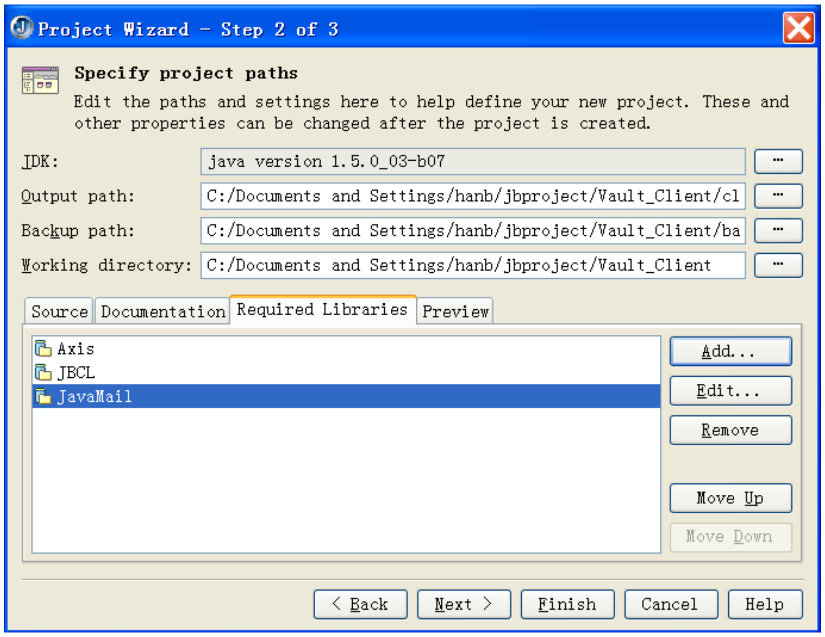
Select name and template for your new JBuilder project

Enter a name for your project and the path to the directory where it is to be saved. You can optionally choose an existing project as a template for the initial default values used in this wizard.

Name: Vault_Client
Directory: C:/Documents and Settings/hanb/jbproject/V
Template: (Default project)

☐ Add project to active project group
☐ Generate project notes file

In the step 2, select the "Required libraries" tab, and add the "Axis", "JBCL" and "JavaMail" libraries into the project, accept other default settings, and click finish to quit the project Wizard.


Project Wizard - Step 2 of 3

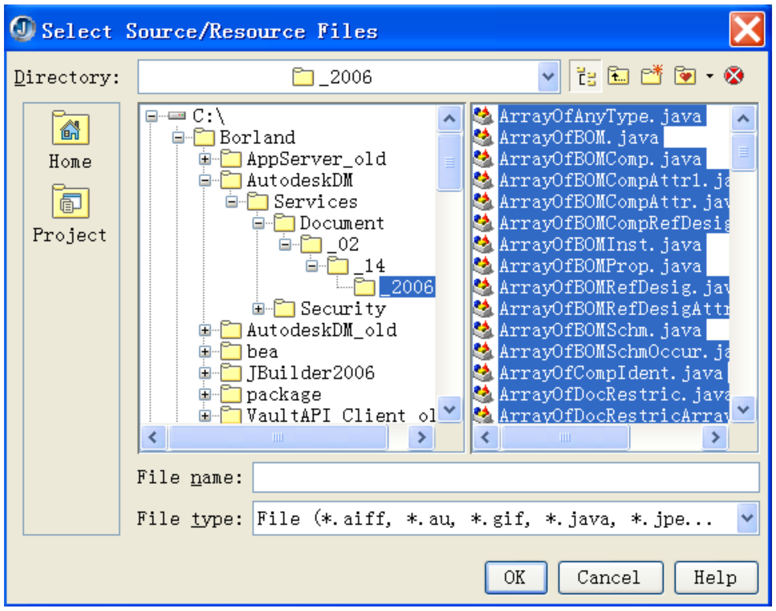
Specify project paths

Edit the paths and settings here to help define your new project. These and other properties can be changed after the project is created.

JDK:	java version 1.5.0_03-b07
Output path:	C:/Documents and Settings/hanb/jbproject/Vault_Client/cl
Backup path:	C:/Documents and Settings/hanb/jbproject/Vault_Client/ba
Working directory:	C:/Documents and Settings/hanb/jbproject/Vault_Client

Required Libraries:
- Axis
- JBCL
- JavaMail

To add the generated stubs (java code) to the JBuilder project, click Project->Import Source... to open Import Source Wizard, click the "Add" button, and locate those generated stubs from the apprpriate folder, e.g. *C:\Borland\AutodeskDM\Services\Document_02_14_2006*, then select all .java files under that folder and click "ok" to add the java code into the project.


Select Source/Resource Files

Directory: _2006

```
C:\
  Borland
    AppServer_old
    AutodeskDM
      Services
        Document
          _02
            _14
              _2006
        Security
    AutodeskDM_old
    bea
    JBuilder2006
    package
    VaultAPI Client ol
```

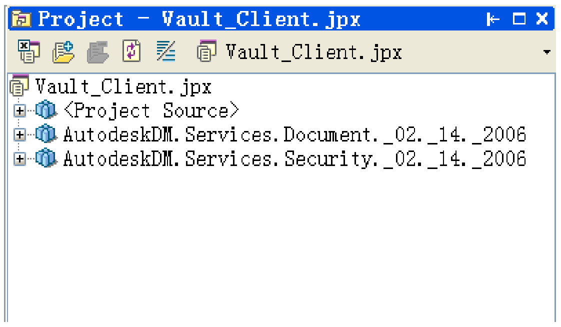
```
ArrayOfAnyType.java
ArrayOfBOM.java
ArrayOfBOMComp.java
ArrayOfBOMCompAttr1.ja
ArrayOfBOMCompAttr.jav
ArrayOfBOMCompRefDesig
ArrayOfBOMInst.java
ArrayOfBOMProp.java
ArrayOfBOMRefDesig.jav
ArrayOfBOMRefDesigAttr
ArrayOfBOMSchm.java
ArrayOfBOMSchmOccur.ja
ArrayOfCompIdent.java
ArrayOfDocRestric.java
ArrayOfDocRestricArray
```

File name:

File type: File (*.aiff, *.au, *.gif, *.java, *.jpe...

Repeat above steps to add generated stubs for SecurityService. Click "OK" to quit the Import Source Wizard. There are now two new packages appearing in the JBuilder project pane, they are:

- AutodeskDM.Services.Document._02._14._2006
- AutodeskDM.Services.Security._02._14._2006



```
Vault_Client.jpx
<Project Source>
AutodeskDM.Services.Document._02._14._2006
AutodeskDM.Services.Security._02._14._2006
```

You might be confused by the ._02._14._2006 suffix on the stub classes. Each web service has a signature which changes from release to release. This is why a release 4 client can't log in to another release server but a release 4 server. The release 5 signature is 02/14/2006 and the stub generator made that part as a part of the class name. Other Vault releases will have another "date" signature in their stub name too.

Login

Clients need to sign in the Vault Server before making any calls to the Vault API services. There are two methods you can use to sign in the Vault – SignIn() and SignIn2(). The difference between these two methods is that SignIn() authenticates you to a specific Knowledge Vault, but the later doesn't. For Admin Service, you don't need to log in a specific knowledge vault, so you should use SignIn2().

The SignIn() and SignIn2() methods for the SecurityService returns void. In .NET IDE, although there is no return value, a successful sign in will populate the SecurityHeaderValue property of the security service for SecurityHeader. The SecurityHeaderValue information is then used for other web service calls.

But with the generated Java stubs for SecurityService, there is no property for SecurityHeader. We need to workaround this problem by adding a specific method to return a valid SecurityHeader object.

The following example creates a SignIn3() method in a SecurityServiceSoapStub.java file to return a SecurityHeader object.

```
    public SecurityHeader signIn3(java.lang.String userName,
                                  java.lang.String userPassword,
                                  java.lang.String knowledgeVault) throws
java.rmi.RemoteException {
        …
        // same code from the original method of signIn()
        …

        Message msg = _call.getResponseMessage();
        SOAPHeaderElement soapHeader = msg.getSOAPEnvelope().getHeaderByName(
                "http://AutodeskDM/Services/Security/02/14/2006/",
                "SecurityHeader");
        java.util.Iterator itr = soapHeader.getChildElements();
        SecurityHeader securityHeader = new SecurityHeader();
        while (itr.hasNext()) {
            javax.xml.soap.SOAPElement elem = (javax.xml.soap.SOAPElement) itr.
                                                                    next();
            if (elem.getElementName().getLocalName().equals("Ticket")) {
                securityHeader.setTicket(elem.getValue());
            }
            if (elem.getElementName().getLocalName().equals("UserId")) {
                securityHeader.setUserId(Long.parseLong(elem.getValue()));
            }
        }
        return securityHeader;
    }
```

Because you need to use a Message class and a SOAPHeaderElement class in the above SignIn3() method, you must import the corresponding libraries in the relevant file -SecurityServiceSoapStub.java -

```
import org.apache.axis.Message;
import javax.xml.soap.SOAPHeaderElement;
```

User Interface

The following table indicates some controls that will be used in the example.

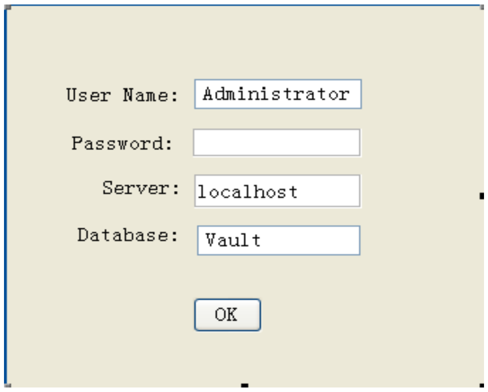
class	name	text
JPanel	jPanel1	
JLabel	jLabel1	User Name:
JTextField	m_txtUserName	Administrator
JLabel	jLabel2	Password:
JTextField	m_txtPassword	
JLabel	jLabel3	Server:
JTextField	m_txtServer	localhost
JLabel	jLabel4	Database:
JTextField	m_txtVaultName	Vault
JButton	btnOk	OK

First, we will have a dialog through which we can sign into the Vault.

Right click on the project name and select New->Package..., add a new package "clientSample". Then right click on the new package, select New->Class... to open the Class Wizard dialog. Select the Base Class javax.swing.JDialog, class name "LoginDialog" and click "OK" to quit the wizard.

You will see LoginDialog appears in the clientsample package.

Add some controls listed in the above table on the LoginDialog.



Now we need a “main()” method in the project as an entry point. Add a TestMain class into the project using the steps outlined above, but check the "Generate main method" option in Class Wizard dialog. The TestMain class can serve as the entry point.

Modify the code of TestMain class as below:

```
 public TestMain() {
     LoginDialog dlg = new LoginDialog();
     dlg.pack();
     dlg.setModal(true);

     Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
     Dimension frameSize = dlg.getSize();
     if (frameSize.height > screenSize.height)
         frameSize.height = screenSize.height;
     if (frameSize.width > screenSize.width)
         frameSize.width = screenSize.width;
     dlg.setLocation((screenSize.width - frameSize.width) / 2,
                     (screenSize.height - frameSize.height) / 2);

     dlg.setVisible(true);
}
public static void main(String[] args) {
     TestMain testmain = new TestMain();
}
```

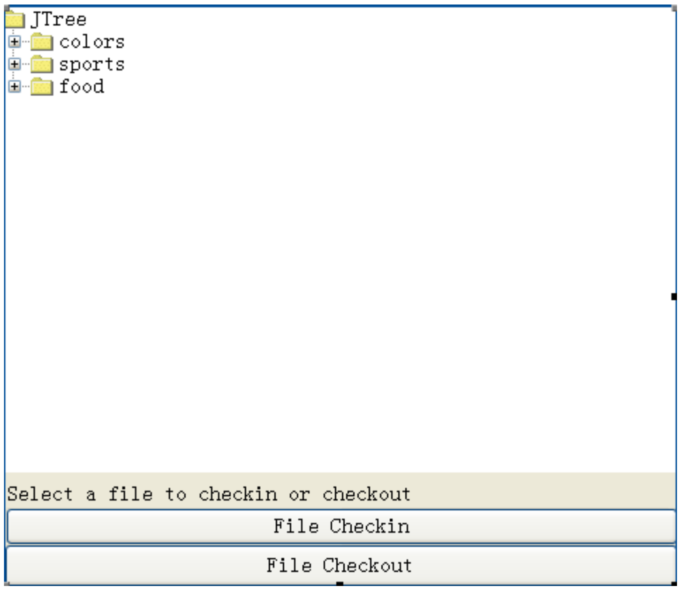
Because we use a Dimension class and a Toolkit class here, import the following two libraries into the TestMain.java file -

```
import java.awt.Toolkit;
import java.awt.Dimension;
```

After that, we will have another dialog with which we can explore the DocumentService capability.

Add a RunDialog class into the project using the steps above (assign the class name as "RunDialog" and Base Class as "JDialog".).

Design the RunDialog as below:



Select a file to checkin or checkout

Add a property for DocumentServiceSoap in the RunDialog class:
DocumentServiceSoap m_docSv = null;

The following is the implementation of the Initialize method for the RunDialog class to get the SecurityHeader object from an external class:

```
public void Initialize(AutodeskDM.Services.Security._02._14._2006.
                                        SecurityHeader svHeader) throws Exception {
            try {
                  DocumentServiceLocator Docloc = new DocumentServiceLocator();
                  m_docSv = Docloc.getDocumentServiceSoap();

                  AutodeskDM.Services.Document._02._14._2006.SecurityHeader svcHeader = new
                              AutodeskDM.Services.Document._02._14._2006.SecurityHeader();
                  String sTicket = svHeader.getTicket();
                  long lUser = svHeader.getUserId();
                  svcHeader.setTicket(sTicket);
                  svcHeader.setUserId(lUser);
                  SOAPHeaderElement shElement = new SOAPHeaderElement(
                              "http://AutodeskDM/Services/Document/02/14/2006/",
                              "SecurityHeader", svcHeader);
                  ((DocumentServiceSoapStub) m_docSv).setHeader(shElement);

            } catch (Exception ex) {
                  ex.printStackTrace();
            }
      }
```

You can implement "OK" button click event in LoginDialog class within which you can pass the SecurityHeader object to the RunDialog class.

```
public void btnOk_actionPerformed(ActionEvent e) {
            try
            {
                 //The following code segment shows how to use the generated SecurityService stub
                SecurityServiceLocator loc = new SecurityServiceLocator();
                SecurityServiceSoapStub port = (SecurityServiceSoapStub)loc.getSecurityServiceSoap();
                String strUserName = m_txtUserName.getText();
                String strPassword = m_txtPassword.getText();
                String strVaultName = m_txtVaultName.getText();
                SecurityHeader securityHeader = port.signIn3(strUserName, strPassword,strVaultName);

                RunDialog dlg = new RunDialog();
                dlg.Initialize(securityHeader);

                 //The following maybe JBuilder specific, you should change it with your IDE.
                dlg.pack();
                this.dispose();
                dlg.setModal(true);

                Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                Dimension frameSize = dlg.getSize();
                if (frameSize.height > screenSize.height)
                        frameSize.height = screenSize.height;
                if (frameSize.width > screenSize.width)
                        frameSize.width = screenSize.width;
                dlg.setLocation((screenSize.width - frameSize.width) / 2,
                                        (screenSize.height - frameSize.height) / 2);

                dlg.setVisible(true);
        } catch (Exception ex)
        {
                ex.printStackTrace();
        }
    }
```

Modify the constructor function of the LoginDialog class as follows:

```
     public LoginDialog() throws HeadlessException {
        super();
        try {
                jbInit();
        } catch (Exception ex) {
                ex.printStackTrace();
        }
    }
```

Import the following libraries into the LoginDialog class:

```
import AutodeskDM.Services.Security._02._14._2006.*;
import java.awt.*;
import javax.swing.JDialog;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import javax.swing.JTextField;
import javax.swing.JPanel;
import javax.swing.*;
import javax.swing.Box;
import java.awt.Dimension;
import javax.swing.BorderFactory;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import com.borland.jbcl.layout.XYLayout;
import com.borland.jbcl.layout.*;
```

And modify the constructor method of the RunDialog class as below:

```
     public RunDialog() throws HeadlessException {
          super();
          try {
               jbInit();
          } catch (Exception ex) {
               ex.printStackTrace();
          }
     }
```

Browsing the folder tree (folder/sub-folders/file versions)

The Vault has a root folder which can have multiple sub-folders and files. Every folder may also have sub-folders and files in it. The files can have multiple versions. The following code shows how to extract this information and display it in a tree structure.

First, add the following code into the Initialize method in the RunDialog class:

```
                Folder rootfolder = m_docSv.getFolderRoot();

               DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode(
                         rootfolder.getFullName());
               refreshFileNode(rootfolder, rootNode);

               DefaultTreeModel model = new DefaultTreeModel(rootNode);
               this.jTree1.setModel(model);
               this.jTree1.updateUI();
               expandAll(this.jTree1,true);
               this.jTree1.repaint();
```

Following code gets the root folder and traverses the children below it and creates tree nodes-

```
       // If expand is true, expands all nodes in the tree.
     // Otherwise, collapses all nodes in the tree.
     public void expandAll(JTree tree, boolean expand) {
          TreeNode root = (TreeNode)tree.getModel().getRoot();

          // Traverse tree from root
          expandAll(tree, new TreePath(root), expand);
     }
     private void expandAll(JTree tree, TreePath parent, boolean expand) {
          // Traverse children
          TreeNode node = (TreeNode)parent.getLastPathComponent();
          if (node.getChildCount() >= 0) {
               for (Enumeration e=node.children(); e.hasMoreElements(); ) {
                    TreeNode n = (TreeNode)e.nextElement();
                    TreePath path = parent.pathByAddingChild(n);
                    expandAll(tree, path, expand);
```

```
                }
            }

            // Expansion or collapse must be done bottom-up
            if (expand) {
                tree.expandPath(parent);
            } else {
                tree.collapsePath(parent);
            }
        }
        private void refreshFileNode(Folder parentFolder,
                                            DefaultMutableTreeNode parentNode) {
            // get the file versions below it
            try {
                ArrayOfFile arrfiles = m_docSv.getLatestFilesByFolderId(
                        parentFolder.
                        getId(), true);
                if (arrfiles != null) {
                    File[] files = arrfiles.getFile();
                    for (int jnx = 0; jnx < files.length; jnx++) {
                        DefaultMutableTreeNode node = new DefaultMutableTreeNode(
                                files[jnx].getName());
                        parentNode.add(node);
                    }
                }

                ArrayOfFolder arrfolders = m_docSv.getFoldersByParentId(
                        parentFolder.
                        getId(), false);
                if (arrfolders != null) {
                    Folder[] folders = arrfolders.getFolder();
                    for (int jmx = 0; jmx < folders.length; jmx++) {
                        DefaultMutableTreeNode node = new DefaultMutableTreeNode(
                                folders[jmx].getFullName());
                        parentNode.add(node);
                        refreshFileNode(folders[jmx], node);
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
```

The Tree structure is represented by many DefaultMutableTreeNodes and each file's name in Vault is the DefaultMutableTreeNode's content.

You also need to import the following libraries into the project:

```
import AutodeskDM.Services.Document._02._14._2006.*;
import AutodeskDM.Services.Security._02._14._2006.*;
import AutodeskDM.Services.Document._02._14._2006.DocumentService.*;

import java.awt.*;
import org.apache.axis.message.SOAPHeaderElement;
import org.apache.axis.*;
import javax.swing.*;
import javax.swing.tree.*;
import java.awt.BorderLayout;
import com.borland.jbcl.layout.XYLayout;
import com.borland.jbcl.layout.*;
import javax.swing.tree.DefaultMutableTreeNode;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import java.util.*;
```

File Download (File CheckOut)

Here CheckOut is downloading the file from the Vault server and making this file writable.

CheckIn is uploading file to the Vault Server and making the file uploaded read-only. The following code shows how to check out a file from the Vault Server.

```
public void btnCheckout_actionPerformed(ActionEvent e) {
    try {
        String strMachine = "localhost";
        String strLocalPath = "c:\\";
        String strComment = "testing java client";
        boolean bDownLoadFlag = true;

         // LAST SELECTED NODE.
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)this.
                                                jTree1. getLastSelectedPathComponent();
        String sFileName = selectedNode.toString();

        ArrayOfString arrName = new ArrayOfString();
        String[] strs = new String[1];
        strs[0] = sFileName;
        arrName.setString(strs);

        ArrayOfFilePathArray arrFilePath = null;
        try
        {
            arrFilePath = m_docSv.getLatestFilePathsByNames(arrName);
        }
        catch (Exception ex) {
            ex.printStackTrace();
            return;
        }

        FilePathArray filePathArr = arrFilePath.getFilePathArray(0);

        ArrayOfFilePath arrFile = filePathArr.getFilePaths();

        FilePath filePath=null;
        try
        {
            filePath = arrFile.getFilePath(0);
```

```
			}
			catch (Exception ex) {
				//System.out.print(Docloc.getServiceName());
				JOptionPane.showMessageDialog(null,"The selected node doesn't represent a file, 
please select another node.","User Selection", JOptionPane.ERROR_MESSAGE);
				return;
			}

			File fileTobeCheckedout = filePath.getFile();
			if (fileTobeCheckedout.isCheckedOut()==true)
			{
				JOptionPane.showMessageDialog(null,"The selected file has been checked out, please 
select another file.","User Selection", JOptionPane.ERROR_MESSAGE);
				return;
			}

			ArrayOfFolder folders = m_docSv.getFoldersByFileMasterId(
					fileTobeCheckedout.getMasterId());
			Folder folder = folders.getFolder(0);

			m_currentFile = m_docSv.checkoutFile(folder.getId(),
										fileTobeCheckedout.getMasterId(),
										strMachine,
										strLocalPath, strComment,
										bDownLoadFlag);

			// Download the file to the disk
			java.lang.Object[] attachments = ((DocumentServiceSoapStub) m_docSv).
									getAttachments();
			if (attachments != null && attachments.length > 0) {
				org.apache.axis.attachments.AttachmentPart attchamentPart =
						(org.apache.axis.attachments.AttachmentPart)
						attachments[0];

				javax.activation.DataHandler rdh = (javax.activation.
						DataHandler) attchamentPart.getDataHandler();

				String receivedfileName = rdh.getName(); //Get the filename.

				if (receivedfileName == null) {
					System.err.println("Could not get the file name.");
					throw new AxisFault("", "Could not get the file name.", null, null);
				}

				java.io.File newFile = new java.io.File(receivedfileName);
				java.io.File destFile = new java.io.File(strLocalPath +
						m_currentFile.getName());
				newFile.renameTo(destFile);
				JOptionPane.showMessageDialog(null,"The file has been checked out, please check it 
in Vault/Ps.","Success information", JOptionPane.INFORMATION_MESSAGE);
			}
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}
```

Add a property to save the selected file in RunDialog class:
File m_currentFile = null;

You need to import the following libraries in the project with this code extension:

```
import java.lang.*;
```

File Upload (File CheckIn)

The files can be checked into Vault by sending it as a DIME attachment to the CheckIn SOAP message. The following code shows how to achieve this task-

```
    public void btnCheckin_actionPerformed(ActionEvent e) {
        try {
            BOM bomData = new BOM();
            AutodeskDM.Services.Document._02._14._2006.ArrayOfLong
                    attachFileIds = null;
            AutodeskDM.Services.Document._02._14._2006.ArrayOfLong
                    dependFileIds = null;

            java.util.Calendar checkInDate = java.util.Calendar.getInstance();

            String strComment = "testing java client";
            boolean bKeepCheckOutFlag = false;

             // LAST SELECTED NODE.
            DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)this.
                                                  jTree1.
                                                  getLastSelectedPathComponent();
            String sFileName = selectedNode.toString();

            if (m_currentFile == null) {
                ArrayOfString arrName = new ArrayOfString();
                String[] strs = new String[1];
                strs[0] = sFileName;
                arrName.setString(strs);
                ArrayOfFilePathArray arrFilePath = m_docSv.
                        getLatestFilePathsByNames(arrName);
                FilePathArray filePathArr = arrFilePath.getFilePathArray(0);

                ArrayOfFilePath arrFile = filePathArr.getFilePaths();

                FilePath filePath=null;
                try {
                    filePath = arrFile.getFilePath(0);
                } catch (Exception ex) {
                    //System.out.print(Docloc.getServiceName());
                    JOptionPane.showMessageDialog(null,
                       "The selected node doesn't represent a file, please select another node.",
                                                  "User Selection",
                                                  JOptionPane.ERROR_MESSAGE);
                    return;
```

```
		}

		m_currentFile = filePath.getFile();
	}
	if (m_currentFile.isCheckedOut()==false)
	{
		JOptionPane.showMessageDialog(null,
				"The selected file has NOT been checked out, No need to check in it.",
							"User Selection",
							JOptionPane.ERROR_MESSAGE);
		return;
	}
	java.io.File newFile = new java.io.File("c:\\" + sFileName);

	javax.activation.DataHandler dh = new javax.activation.DataHandler(
			new javax.activation.FileDataSource(newFile));

	((DocumentServiceSoapStub) m_docSv).addAttachment(dh);
	((DocumentServiceSoapStub) m_docSv)._setProperty(org.apache.axis.
			client.Call.
			ATTACHMENT_ENCAPSULATION_FORMAT,
			org.apache.axis.client.Call.
			ATTACHMENT_ENCAPSULATION_FORMAT_DIME);

	ArrayOfFileAssoc arrFileAssoc = m_docSv.getFileAssociationsById(
			m_currentFile.getId(), AssociationRelationEnum.Children, false, true);

	// Any file other than DesignVisualiztion type of file has association information
	if (arrFileAssoc != null){
		attachFileIds = new utodeskDM.Services.Document._02._14._2006.ArrayOfLong();
	      dependFileIds = new AutodeskDM.Services.Document._02._14._2006.ArrayOfLong();
		for (int j = 0; j < arrFileAssoc.getFileAssoc().length; j++) {
			FileAssoc assoc = arrFileAssoc.getFileAssoc(j);
			if (assoc.getTyp() == AssociationType.Attachment) {
				long[] longs = new long[1];
				longs[0] = assoc.getCldFile().getId();
				attachFileIds.set_long(longs);
			} else {
				long[] longs = new long[1];
				longs[0] = assoc.getCldFile().getId();
				dependFileIds.set_long(longs);
			}
		}
	}
	// Perform the check in
	m_currentFile = m_docSv.checkinFile(m_currentFile.getMasterId(),
							strComment, bKeepCheckOutFlag,
							checkInDate, dependFileIds,
							attachFileIds, bomData,
							m_currentFile.getName(),
							(FileClassification)
							m_currentFile.getFileClass(),
							false);
	newFile.delete();
	JOptionPane.showMessageDialog(null,"The file has been checked in, please check it in
Vault/Ps.","Success information", JOptionPane.INFORMATION_MESSAGE);
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```

This concludes the code required for this sample Java Client application.

Conclusion

This whitepaper illustrates how to start writing a Vault client application using Java – this includes signing into a Vault Server, browsing all folders and files and performing file CheckIn and CheckOut between the Vault Server and the disk.

In general, writing Vault client applications using Java is simple and straight forward. The first step is to generate the client stubs for services. These can then be directly used without any modifications (with the exception of SignIn3() method). Then create your Java project and import those client stubs, sign into the Vault server and make any required calls to the Vault API Services.

I hope that this is a good start to get you going with your Java Vault Client implementation and wish you success with your application!

- *Barbara*