Oracle® AutoVue Web Services

Installation and Developer's Guide

Release 21.0.2.9

F10656-02

March 2024

Oracle® AutoVue Web Services Installation and Developer's Guides, Release 21.0.2.9

F10656-02

Preface

The AutoVue Web Services Installation and Developer's Guide is in two parts. The Installation Manual part of the guide describes the installation of AutoVue Web Services on the Windows and Linux platforms and how to configure AutoVue Web Services for a connection with a Document Management System (DMS) repository.

The Developer's Guide describes how to create a Web service client stub for the AutoVue Web Services package, how to use the generated code inside your application, and how to call AutoVue Web Services methods from inside your code.

For the most up-to-date version of this document, go to the AutoVue Documentation Web site on the Oracle Technology Network (OTN) at `https://www.oracle.com/technetwork/documentation/autovue-091442.html`.

Audience

This document is intended for third-party developers (for example integrators) who want to integrate Oracle AutoVue with other application suites and legacy systems.

It is also intended for third-party developers who want to implement SOAP-based integration with AutoVue.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle AutoVue Web Services documentation set on OTN:

- *Oracle AutoVue Web Services Release Notes*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Installation and Configuration Manual

This part describes the installation of AutoVue Web Services and how to configure AutoVue Web Services for a connection with a DMS repository.

Part I contains the following chapters:

1 Introduction

Oracle AutoVue Web Services provides a standard interface that allows for easy integration of Oracle AutoVue with other application suites and legacy systems. With AutoVue Web Services, developers can easily integrate AutoVue's best-in-class enterprise visualization capabilities where they are most needed in the enterprise, regardless of platforms or programming languages. Teams can use AutoVue functionality in a completely transparent way to view and collaborate on business and technical information directly from other applications and enterprise systems, and perform their tasks in a more efficient manner.

AutoVue Web Services represents many AutoVue functionalities such as print, convert, text extraction, and more in the structure defined by Web Service Description Language (WSDL).

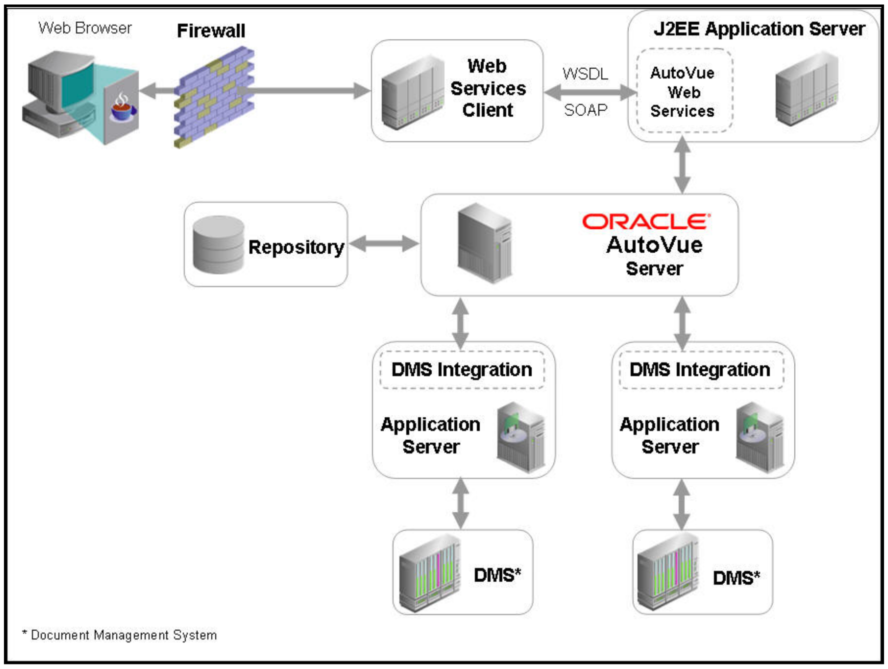
The AutoVue Web Services package acts as a wrapper around Oracle AutoVue client. It exposes certain AutoVue functionalities as Web methods, and translates AutoVue Web Services requests to and from AutoVue (for example, AutoVue messages to AutoVue Web Services responses). Additionally, AutoVue Web Services enables AutoVue to communicate with any third-party application that wants to invoke AutoVue in a Service Oriented Environment (SOE).

The AutoVue Web Services package is designed to work seamlessly with Document Management Systems (DMS) through various DMS integrations. It can also work with local files and Uniform Resource Identifiers (URIs) that are accessible to the host machine.

Note: Not all of AutoVue's functionalities are represented in the AutoVue Web Services package. This is because many of the functionalities require user interaction (for example, online collaboration, digital mockup, and so on) and are not suitable for application-to-application communication; which is the main objective of AutoVue Web Services

The following diagram displays the communication process for AutoVue Web Services:

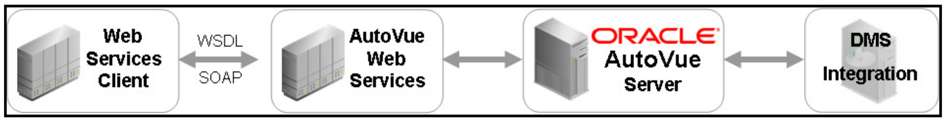
Figure 1–1 Communication Process for AutoVue Web Services



AutoVue Web Services can also invoke operations on files inside Document Management Systems (DMS) repositories. To access the DMS repository, a DMS integration interface is required between the DMS server, AutoVue Web Services, and Oracle AutoVue. This interface enables you to add powerful viewing and markup capabilities to your DMS via a Web browser in an intranet or the Internet.

The following image displays the relationship between Web Services client, AutoVue Web Services, and AutoVue Server:

Figure 1–2 Flow Diagram



2 Installation Prerequisites

Before installing AutoVue Web Services, ensure that AutoVue Server and your J2EE certified Application Server are properly installed and configured on your system according to the manufacturer's instructions.

Note: It is recommended to test both your Application Server and AutoVue Server independently to verify that the installation was successful and that all functionalities are available and produce the expected results.

To run the Web Services installer in graphical mode on Linux, the libXp package must be installed on the machine.

3 System Requirements

- Oracle AutoVue 21.0.2
- Server Operating Systems
  - Windows Server 2008 R2
    - 64-bit (AutoVue running in 32-bit mode)
  - Windows Server 2012 R2
    - 64-bit (AutoVue running in 32-bit mode)
  - Windows Server 2016
    - 64-bit (AutoVue running in 32-bit mode)
  - Oracle Linux 6.X (x86_64), and 7.X (x86_64)
    - 64-bit (AutoVue running in 32-bit mode)
  - Red Hat Enterprise Linux 6.X (x86_64), and 7.X (x86_64)
    - 64-bit (AutoVue running in 32-bit mode)
- A J2EE 7 or later Application Server
  - Oracle AutoVue Web Services is certified with Oracle WebLogic 12cR2.
  - AutoVue Web Services uses Java annotation and other features introduced in Java Platform, Enterprise Edition (Java EE) 5 or later. As a result, AutoVue Web Services can only be deployed on a Java EE certified application server.
- The following VueLinks have been validated with AutoVue Web Services:
  - VueLink 19.3.2 for Documentum/WebTop 6.8 SP2
  - VueLink 20.1 for Oracle UCM/WCC 11.1.1.8
- To test AutoVue Web Services without writing any code, you can use the following:
  - Oracle Web Services Manager (either standalone or part of SOA)
  - SoapUI
  - Oracle JDeveloper that has built-in tools for testing Web Services

Note: If Web Services is installed on a Linux machine, the Linux machine must start at level 5: X11 which is level 3 + display manager.

If you would like to connect to a Linux machine through the X window system (for example, Xming), you must use the one with Mesa 3D capability (for example, Xming-mesa) in order to print a file that contains 3D pages.

AutoVue Web Services has the same dependency as the AutoVue client in terms of the third-party libraries (for example, libGL.so and libGLU.so on Linux). For more information refer to the *Oracle AutoVue Client/Server Installation and Configuration Guide*.

4

Installing and Configuring AutoVue Web Services

This chapter describes the installation and configuration steps for AutoVue Web Services.

To install, run the installer to extract all necessary files and then manually configure AutoVue Web Services. If you are using a DMS integration, you must configure its properties file. Once configuration is complete, create and deploy the AutoVue Web Services WAR file with your application server.

4.1 Installing AutoVue Web Services

The two folders that are included in the AutoVue Media package under the WebServices directory: win32 and linux. Each of these folders contain the AutoVue Web Services installer for the corresponding platform:

- For Windows, go to the *win32* folder and launch the setupwin32.exe file.
- For Linux, go to the *linux* folder and launch the setuplinux.bin file.

Note: It is recommended to install AutoVue Web Services in the default installation directory: C:\Oracle\AutoVueWS on Windows.

4.2 Configuring AutoVue Web Services web.xml

1. From the <AutoVue Web Services Installation Directory>\autovue_webservices\AutoVueWS\WEB-INF directory, open web.xml in a text editor.
2. On Windows operating systems, if the default directory (C:\Oracle\AutoVueWS) is selected during installation, proceed to step 3. If a different installation path is selected, do the following:

   a. Locate the following line of code:

   ```
   <env-entry-value>C:/Oracle/AutoVueWS/autovue_webservices/sample_
   config/log4j.properties</env-entry-value>
   ```

   b. Replace *C:/Oracle/AutoVueWS/autovue_webservices/sample_config/log4j.properties* with the actual full file path for the log4j.properties file.

   Note: On Linux, you have to follow the same procedure as provided in steps a and b.

3. If you want AutoVue Web Services to access a DMS repository (for example, files inside Oracle WebCenter Content or a third-party integration), the value for the environment entry name *vuelinkProtocol* must be specified.

   a. Locate the following code:

```
<env-entry>
<env-entry-name>vuelinkProtocol</env-entry-name>
<env-entry-type>java.lang.String</env-entry-type>
<env-entry-value></env-entry-value>
<injection-target>
<injection-target-class>com.oracle.autovue.services.VueBeanWS</injection-ta
rget-class>
<injection-target-name>vuelinkProtocol</injection-target-name>
</injection-target>
</env-entry>
```

   b. Enter a semicolon (;) separated list of DMS integration protocols between the *<env-entry-value></env-entry-value>* tags. For example:

```
<env-entry-value>
DMS_Integration_1;DMS_Integration_2;...;DMS_Integration_n
</env-entry-value>
```

   Where DMS_Integration_1 to DMS_Integration_n represent protocols for different DMS integrations. Possible DMS integration protocols are as follows:

Table 4–1 DMS integration protocols

Protocol Examples	Description
vuelinkUCM	Represents the protocol of VueLink for Oracle UCM/WCC
vuelinkDCMT	Represents the protocol of VueLink for Documentum
Third_Party_Name	Represents the protocol of a third-party integration

For information on configuring DMS integrations, refer to Configuring DMS Integration Properties

4. Verify that value for *vuelinkPropsDir* points to the directory that contains the DMS connection properties files: DMS_Integration_1.properties, DMS_Integration_2.properties, and so on. To do so, locate the following line of code:

```
<env-entry-value>C:/Oracle/AutoVueWS/autovue_webservices/sample_
config/</env-entry-value>
```

   If different, replace *C:/Oracle/AutoVueWS/autovue_webservices/sample_config/* with the correct full directory path.

5. Specify the value for the environment entry name *destinationDIR.* The destinationDIR directory is the destination location where AutoVue Web Services stores temporary files on the server. All temporary files are deleted when they are no longer required by AutoVue Web Services.

   Note: Ensure AutoVue Web Services has the proper write accesses for the destinationDIR directory.

   a. Locate the following line of code:

```
<env-entry-value>full path to output directory</env-entry-value>
```

   b. Replace *full path to output directory* with the location that the converted files are stored in the Web services server machine (for example, C:/temp).

6. Specify the value for the environment entry name *initialJVueServer*.

   a. Locate the following line of code:

   ```
   <env-entry-value>http://hostname:port/context/servlet/VueServlet</env-entry
   -value>
   ```

   b. Replace *http://hostname:port/context/servlet/VueServlet* with the URL that points to the AutoVue VueServlet that is deployed on your application server. The default is http://localhost:7003/VueServlet/servlet/VueServlet.

7. If you want Web Services to be able to address more requests simultaneously, you can increase the value for the environment entry maxPoolSize. The default value is 1.

8. When there is no VueBean object in the pool, no new request can be processed until one of the VueBean objects becomes available. In such case a new request will be put on hold for a certain period of time, waiting for a VueBean object to be available in the pool. The value for the environment entry maxWait specifies the period of time. The default value is 7200 seconds.

9. The value for the environment entry maxParallelPrintJob specifies the maximum allowed number of parallel print jobs per printer. The default value is 1.

10. The value for the environment entry name maxPrinterJobBuffer specifies the maximum allowed number of jobs in the buffer per printer. This means that the Web Services tries not to overload the printer buffer by sending all jobs to the printer without any feedback from the print buffer. The default value is 5.

11. The value for the environment entry name minMemory specifies the minimum amount of memory in MB required before a VueBean is allowed to open a document. The default value is 128MB.

12. The value for the environment entry name isUploadProtocolEnabled specifies whether Web Services allows upload protocol or not. For security purposes, the default value is FALSE.

4.3 Configuring VueServlet web.xml

1. From the <AutoVue Web Services Installation>\autovue_webservices\VueServlet\WEB-INF directory open web.xml in a text editor.

2. Update the default location of AutoVue server "localhost:5099". You must replace localhost with the host name/IP address of the machine that is running the AutoVue server, and replace 5099 with the socket port number that the AutoVue server is listening to (default is 5099).

Note: To add multiple AutoVue servers, add them as semi-colon separated list (;).

4.4 Configuring DMS Integration Properties

If you specified the DMS integration protocols in Configuring AutoVue Web Services web.xml then the connection properties files with the same name must be created and configured in the location specified by the environment entry *vuelinkPropsDir* in web.xml. The connection properties files in this location are responsible for the connection between AutoVue Web Services and the DMS integration.

The following sections explain how to configure the connection properties files for:

- VueLink for Documentum
- VueLink for Oracle UCM/WCC
- Third-party integrations

4.4.1 VueLink for Documentum

1. Verify that the DMS connection properties file, *vuelinkDCMT.properties,* is in the location specified by the environment entry *vuelinkPropsDir* block in Web Services web.xml file.
2. Open the vuelinkDCMT.properties file in a text editor.
3. Locate the following line of code:

   ```
   DMS=http://appserver:port/context/com.cimmetry.vuelink.documentum.DMS
   ```

4. Replace *appserver:port* with the host and port of the application server that deploys VueLink for Documentum.
5. Replace *context* with the context name of VueLink for Documentum in the application server (for example, webtop).
6. Save the vuelinkDCMT.properties file.

4.4.2 VueLink for Oracle UCM/WCC

Note: Oracle UCM has been renamed as Oracle WebCenter Content (WCC).

1. Verify that the DMS connection properties file, *vuelinkUCM.properties,* is in the location specified by the environment entry name *vuelinkPropsDir* block in Web Services web.xml file.
2. Open the vuelinkUCM.properties file in a text editor.
3. Locate the following line of code:

   ```
   DMS=http://appserver:port/context/DMS
   ```

4. Replace *appserver:port* with the host and port of the application server that deploys VueLink for Oracle UCM/WCC.
5. Replace *context* with the context name of VueLink for Oracle UCM/WCC in the application server (for example, vuelink).
6. Save the vuelinkUCM.properties file.

4.4.3 Third-Party Integrations

With AutoVue Web Services, you can use a third party integration based on AutoVue ISDK between AutoVue and your own DMS repository (for example, Third_Party_Name DMS).

The following steps explain how to invoke AutoVue Web Services for files stored inside the Third_Party_Name DMS repository.

1. In the *<AutoVue Web Services Installation Directory>\autovue_webservices\sample_config* directory, create a file named "Third_Party_Name.properties".
2. Verify that Third_Party_Name DMS integration protocol is defined in the environment entry name *vuelinkProtocol* block in web.xml. Refer to Configuring AutoVue Web Services web.xml.
3. Open the connection properties file in a text editor, and assign the DMS value the URL that points to the third-party DMS integration servlet that is based on AutoVue ISDK. For example, assuming that the application server is on port 8080 of the appSrv1 machine and the path to the DMS servlet is */Third_Party_Name/DMS,* enter the following line of code:

   ```
   DMS=http://appSrv1:8080/Third_Party_Name/DMS
   ```

4. If used by the Third_Party_Name DMS repository, set the DMS arguments (DMSArgs). For example:

   ```
   DMSArgs=Arg1;Arg2
   Arg1=value
   Arg2=value
   ```

   AutoVue Web Services passes these DMSArgs to your Third_Party_Name DMS servlet.
5. Save the Third_Party_Name.properties file.

4.5 Configuring Web Services over HTTPS/SSL

We recommend that you configure your application server to allow HTTP connections over Secure Socket Layer (SSL). For more information on how to set up AutoVue Web Services to run in a SSL environment, refer to HTTPS/SSL in the Developer's Guide.

Note: If you choose not to run AutoVue Web Services over SSL, any data (including any user credentials) sent to AutoVue Web Services will be in clear text and not encrypted. It is recommended to use SSL for secure communication.

4.6 Creating and Deploying the WAR File

1. From the *<AutoVue Web Services Installation>\autovue_webservices\* directory, run the *createWarfile.bat/createWARfile.sh* file.

   The AutoVue Web Services WAR file, AutoVueWS.war, and VueServlet WAR file, VueServlet.war, are created in the *<AutoVue Web Services Installation>\autovue_webservices\* directory.
2. Deploy the WAR file with your application server.

   If you are using Oracle WebLogic, refer to Deploying AutoVue Web Services on Managed Server of Oracle WebLogic.

   If you are using another application server, refer to your application server's documentation for information on deploying a WAR file.
3. Verify the deployment. For more information, refer to Verification.

Note: If you wish to modify the AutoVue Web Services configuration after it is deployed, refer to Modifying AutoVue Web Services Configuration After Deployment.

4.7 Modifying AutoVue Web Services Configuration After Deployment

If you want to modify web.xml after you deploy AutoVue Web Services to the application server, do the following:

1. Undeploy AutoVue Web Services.
2. Restart your application server.
3. Modify web.xml.
4. Create the WAR file.
5. Deploy the WAR file to your application server.

4.8 Verification

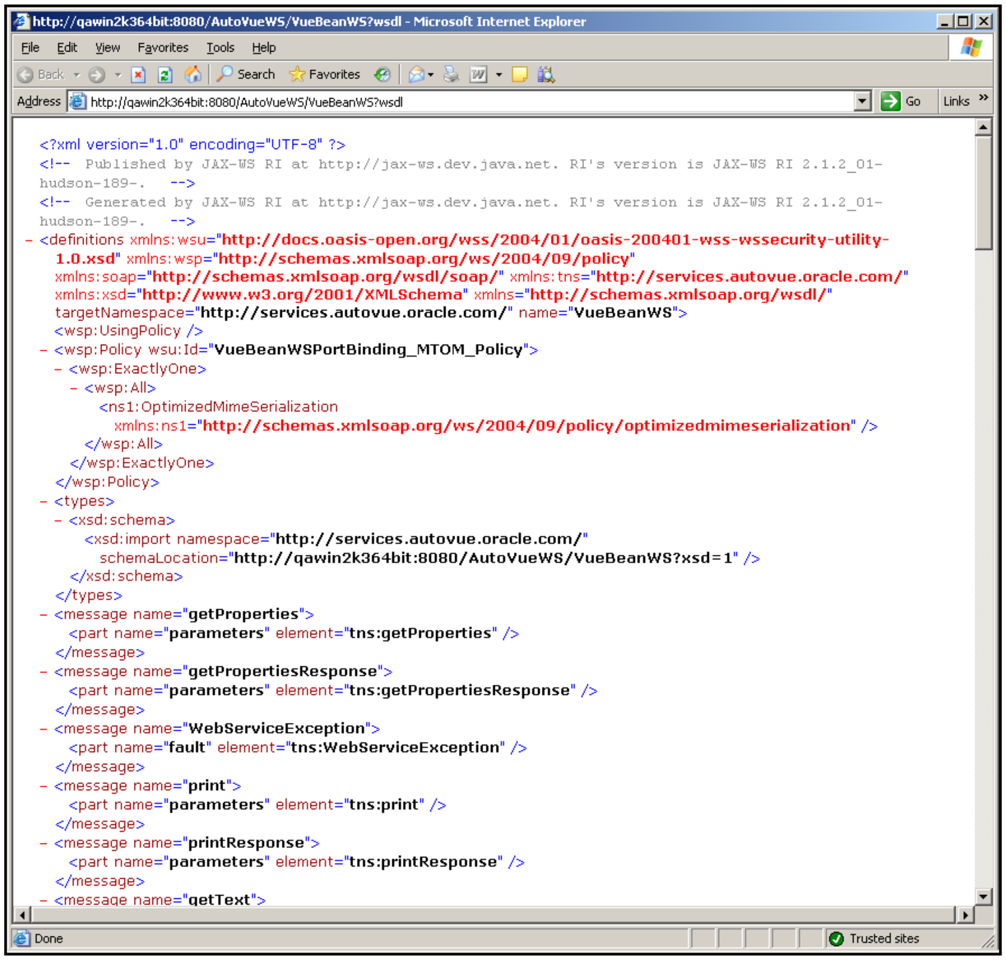
To verify that AutoVue Web Services is running properly, launch your Web browser and enter the URL pointing to *index.jsp*. To verify that the VueServlet is running properly, launch your Web browser and enter the URL pointing to the VueServlet. The following examples show how to verify that AutoVue Web Services and VueServlet are running properly.

Example 4–1 Verify AutoVue Web Services deployment with URL pointing to index.jsp

1. Enter the following URL in the Web browser: *http://[host:port]/AutoVueWS/index.jsp* The Java Environment System Properties are displayed.
2. Select **Click here to view WSDL.** The formatted XML for WSDL displays.

   The Figure 4–1 displays a sample indicating that AutoVue Web Services is running properly. If you do not receive a similar response, refer to the chapter – Installing and Configuring AutoVue Web Services.

Figure 4–1 AutoVue Web Services sample



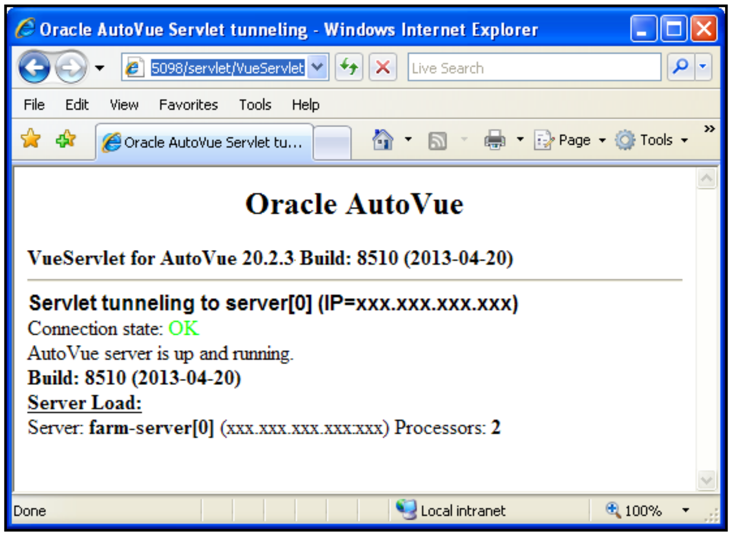
http://qawin2k364bit:8080/AutoVueWS/VueBeanWS?wsdl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://qawin2k364bit:8080/AutoVueWS/VueBeanWS?wsdl Go Links
Done Trusted sites

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2_01-
hudson-189-.  -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2_01-
hudson-189-.  -->
- <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
    1.0.xsd" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://services.autovue.oracle.com/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://services.autovue.oracle.com/" name="VueBeanWS">
    <wsp:UsingPolicy />
  - <wsp:Policy wsu:Id="VueBeanWSPortBinding_MTOM_Policy">
    - <wsp:ExactlyOne>
      - <wsp:All>
          <ns1:OptimizedMimeSerialization
            xmlns:ns1="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization" />
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://services.autovue.oracle.com/"
          schemaLocation="http://qawin2k364bit:8080/AutoVueWS/VueBeanWS?xsd=1" />
      </xsd:schema>
    </types>
  - <message name="getProperties">
      <part name="parameters" element="tns:getProperties" />
    </message>
  - <message name="getPropertiesResponse">
      <part name="parameters" element="tns:getPropertiesResponse" />
    </message>
  - <message name="WebServiceException">
      <part name="fault" element="tns:WebServiceException" />
    </message>
  - <message name="print">
      <part name="parameters" element="tns:print" />
    </message>
  - <message name="printResponse">
      <part name="parameters" element="tns:printResponse" />
    </message>
  - <message name="getText">
```

Example 4–2 Verify AutoVue Web Services deployment with URL pointing to the VueServlet

1. Start the AutoVue server.
2. Enter the following URL in the Web browser: http://[host:port]/VueServlet/servlet/VueServlet The following screenshot displays a sample indicating that the VueServlet is running properly. If you do not receive a similar response, refer to the chapter – Installing and Configuring AutoVue Web Services.

Figure 4–2 VueServlet sample


Oracle AutoVue Servlet tunneling - Windows Internet Explorer
5098/servlet/VueServlet
Live Search
File Edit View Favorites Tools Help
Oracle AutoVue Servlet tu...
Page
Tools
Done
Local intranet
100%

Oracle AutoVue

VueServlet for AutoVue 20.2.3 Build: 8510 (2013-04-20)

Servlet tunneling to server[0] (IP=xxx.xxx.xxx.xxx)
Connection state: OK
AutoVue server is up and running.
Build: 8510 (2013-04-20)
Server Load:
Server: **farm-server[0]** (xxx.xxx.xxx.xxx:xxx) Processors: **2**

5

Uninstalling AutoVue Web Services

To uninstall AutoVue Web Services, perform the following steps:

1. Manually delete the *AutoVueWS.war* and *VueServlet.war* files.
2. From the *<AutoVue Web Services Installation Directory>_uninst* directory, run *uninstaller.exe/uninstaller.bin* to delete the installation folder.
3. Undeploy the WAR files from the application server.
4. Undeploy AutoVue Web Services and VueServlet from the application server.
5. If applicable, delete the previous manually copied log4j.properties and DMS integration connection properties files.

6

Upgrading AutoVue Server

If there is an upgrade for AutoVue server, you must also update the jvue.jar, jogl.jar, gluegen-rt.jar, jsonrpc.jar, gluegen-rt-natives-macosx-universal.jar, jogl-natives-macosx-universal.jar, gluegen-rt-natives-windows-amd64.jar, jogl-natives-windows-amd64.jar, gluegen-rt-natives-linux-amd64.jar, jogl-natives-linux-amd64.jar, jogl-natives-windows-i586.jar, and gluegen-rt-natives-windows-i586.jar. To do so:

1. Replace your old versions of *jvue.jar, jogl.jar*, *jsonrpc4j.jar* and *gluegen-rt.jar* located in the *<AutoVue Web Services Installation Directory>\autovue_ webservices\AutoVueWS\WEB-INF\lib* directory with the new release from the patch. Replace your old version of vueservlet.jar located in the <AutoVue Web Services Installation Directory>\autovue_webservices\ VueServlet \WEB-INF\lib directory with the new release from the patch.
2. From the *<AutoVue Web Services Installation Directory>\autovue_webservices\* directory run the *createWarfile.bat/createWARfile.sh* to create new WAR files.
3. Deploy the WAR files to the application server.

For information on deploying the war file, refer to Creating and Deploying the WAR File.

7 Configuring Oracle Web Services Manager to Secure AutoVue Web Services

Oracle Web Services Manager can be configured to provide security for AutoVue Web Services. For configuration information, refer to the Oracle Web Services Manager documentation on the Oracle Technology Network (OTN):

`https://docs.oracle.com/middleware/1212/owsm/index.html`

Part II

Developer's Guide

This part of the AutoVue Web Services Installation and Developer's Guide describes how to create a Web service client stub for the AutoVue Web Services package, how to use the generated code inside your application, and how to call AutoVue Web Services methods from inside your code. This manual also serves as a good starting point for developers and professional services to become more familiar with the technical details of this package.

AutoVue Web Services is intended for system integrators or developers who want to integrate Oracle AutoVue with their applications. AutoVue Web Services is written in Java and based on Java API for XML Web Services (JAX-WS). Clients that consume AutoVue Web Services can be written in any language such as Java or .NET as long as they understand WSDL and communicate using Simple Object Access Protocol (SOAP).

Part II contains the following chapters:

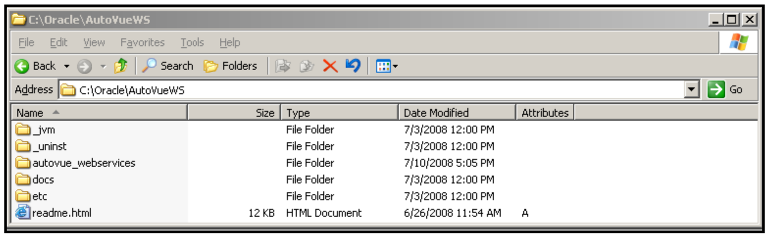
8 AutoVue Web Services

With AutoVue Web Services, developers can easily integrate AutoVue's best-in-class enterprise visualization capabilities where they are most needed in the enterprise, regardless of platforms or programming languages.

8.1 Getting Started

After you run the installer for AutoVue Web Services on your machine, several folders are created. The following screenshot displays the folder structure that is created when installed in the default installation directory (C:\Oracle\AutoVueWS) on Windows:

Figure 8–1 Folder Structure of AutoVue Web Services



Name	Size	Type	Date Modified	Attributes
_jvm		File Folder	7/3/2008 12:00 PM	
_uninst		File Folder	7/3/2008 12:00 PM	
autovue_webservices		File Folder	7/10/2008 5:05 PM	
docs		File Folder	7/3/2008 12:00 PM	
etc		File Folder	7/3/2008 12:00 PM	
readme.html	12 KB	HTML Document	6/26/2008 11:54 AM	A

The *readme.html* file acts as an entry point to the remaining documentation for AutoVue Web Services. To view the contents of the file, open it from a Web browser.

The following is a brief description of what is contained in each folder after the installation of AutoVue Web Services:

- The /docs folder contains the JavaDocs. All other documentation can be found on the Oracle Technology Network (OTN) `https://www.oracle.com/technetwork/documentation/autovue-091442.html`.
- The /Javadocs folder contains JavaDocs for Oracle AutoVue Web services.
- The */autovue_webservices* folder contains files needed to generate AutoVueWS.war for deployment into J2EE application server:
  - *AutoVueWS:* A staging folder for generating AutoVueWS.war file.
  - *sample_config:* Contains configuration files used by AutoVueWS.war.
  - *createWARfile.bat:* Batch file which generates AutoVueWS.war and VueServlet.war on Windows OSes.
  - *VueServlet:* Staging folder for generating VueServlet.war.

  - *createWARfile.sh:* Shell scripting which generates AutoVueWS.war and VueServlet.war on Linux OSes.

An optional folder – sample_client is also created in the <Av Web Services Install Dir> if you select the AutoVue Web Services Sample Client Code check box during installation of Web Services.

- *sample_client:* Contains sample AutoVue Web Services client code which demonstrates a persistent retry as long as the server is busy or when there is not enough memory.
- The */etc* folder contains the following files:
  - **version.txt:** Version information
  - **fileslist.txt:** List of files and folders structure contained in this release
  - **3rdParty:** This folder contains licenses of the included software components developed by 3rd party companies. It has the following subfolders:
    - **apache:** This folder contains licenses of the included software developed by the Apache Software Foundation (http://www.apache.org/)
    - **jogamp:** This folder contains licenses of the included JOGL software developed by jogamp community (http://jogamp.org/)
- The */_jvm* and */_uninst* folders are used for uninstalling AutoVue Web Services.

Initially, you must generate the AutoVueWS.war Web application module and deploy it into your Application Server.

You also need to ensure that you have all prerequisite software installed before you start deployment. For a complete list specific to your platform, refer to the chapter – System Requirements.

Once you have successfully deployed the AutoVueWS.war Web application, you should familiarize yourself with the available Web services. For more information on the features and functionalities provided by each Web service, refer to the chapter – Using AutoVue Web Services.

To test AutoVue Web Services without writing any code, you can use Oracle Web Services Manager. For version information refer to System Requirements.

You can also create your own Web services proxy client that consumes AutoVue Web Services. For more information refer to Generating Client Proxy Using WSimport.

8.2 DMS Integration

Through a document management system (DMS) integration, AutoVue Web Services can invoke operations on files located in DMS repositories (such as Oracle WebCenter Content, third-party integrations, and so on). Each DMS integration is associated to a connection properties file that manages the interaction between AutoVue Web Services and the DMS integration.

Note that each DMS integration has an associated properties file. For example:

- VueLink for Oracle UCM/WCC is assigned *vuelinkUCM.properties*
- VueLink for Documentum is assigned *vuelinkDCMT.properties*
- A third-party application is assigned *vuelinkTHIRD_PARTY.properties*

For more information on DMS integration and configuring connection properties files, refer to Configuring DMS Integration Properties.

8.3 Overview of Components

AutoVue Web Services contains two main components: JavaDocs and AutoVue Web Services Module.

8.3.1 AutoVue Web Services Module

AutoVue Web Services Module consists of the following components: AutoVueWS.jar, AutoVue components, third-party libraries, and batch utility.

8.3.1.1 AutoVueWS.jar

This is the main library that is used as the end point for AutoVue Web Services. It is responsible for processing all incoming Simple Object Access Protocol (SOAP) messages and building responses to AutoVue Web Services clients.

- AutoVue Web Services uses standard Web descriptor file (web.xml) for storing configuration parameters and log4j for logging messages into an output log file. You can specify the location and filename of the logs file in property.logpath and appender.rolling.fileName, respectively, in the log4j.properties file

8.3.1.2 AutoVue Components

AutoVue Web Services is built as a wrapper around the AutoVue client. The following component of Oracle AutoVue is bundled with AutoVue Web Services:

- AutoVue Client (jvue.jar)
- AutoVue Web Services does not bundle the AutoVue server. As a result, you must download it separately.

8.3.1.3 Third-Party Libraries

AutoVue Web Services bundles the following third-party open source libraries:

- Commons-pool-1.5.4.jar
- Log4j-api.jar
- Log4j-core.jar
- Jogl.jar
- Gluegen-rt.jar
- jsonrpc4j.jar
- gluegen-rt-natives-macosx-universal.jar
- jogl-natives-macosx-universal.jar
- gluegen-rt-natives-windows-amd64.jar
- jogl-natives-windows-amd64.jar
- gluegen-rt-natives-linux-amd64.jar
- jogl-natives-linux-amd64.jar
- jogl-natives-windows-i586.jar
- gluegen-rt-natives-windows-i586.jar

8.3.1.4 Batch Utility

CreateWARfile.bat/createWARfile.sh is a batch utility file that generates a Web Archive (WAR) file for easy deployment into your Application Server. Before running this utility, ensure that your settings inside various configuration files (web.xml, log4j.properties, and so on) are correct.

8.3.2 List of AutoVue Web Services

The Table 8–1 is a summary of methods provided by AutoVue Web Services. For more information, see AutoVue Web Services Methods.

Table 8–1 Summary of Methods

Web Services	Description
getText	This text extraction Web method returns visible text inside a given document. This method is not supported for 3D formats. Metadata is not included by this method (for example, EDA entity information, layer and block names, and so on).
getProperties	This file level metadata extraction Web method returns metadata and properties for a given file. From the Operation list, select **getProperties** and wait for the page to refresh. This method only needs a valid URI. Authorization is needed only if the URI cannot be accessed without it.
getXRefs	This External References (XRefs) Web method returns a list of XRefs associated to a given file.
getPartTree	This part tree extraction Web method returns a list of parts contained in a given file. This method is only supported for 3D formats. For example, in the case of a 3D assembly, this Web service returns a list of parts and sub-assemblies referenced by the 3D assembly.
getPartProperties	This part level metadata extraction Web method returns metadata for a given part in a given file. This method is only supported for 3D formats. For example, in the case of a 3D assembly, this Web service returns properties of a particular part referenced by the 3D assembly.
print	This printing Web method sends a given file to a printer for printing.
packetPrint	Prints a group of documents (known as packet) one at a time, along with the auto-generated cover page and summary page.
convert	This conversion Web method converts a given file into another format such as BMP, PDF, or TIFF.
getLayerInfo	This Web method returns a list of all available layers of a given document.
getPrinterNameList	This utility Web method returns a list of available printers.
getPaperList	This utility Web method returns a list of different papers supported by a given printer.

8.4 How AutoVue Web Services Works

After the AutoVue Web Services package is deployed, the VueBeanWS.wsdl (AutoVue's WSDL interface) provides the required gateway to client applications. The client applications can identify the available Web methods and their input/output parameters through the WSDL, and generate the required communication proxy. Next, automatically generate the Web Services client stub for AutoVue from the tools provided by the programming languages (all AutoVue Web Services methods are defined in a single WSDL).

Once the client stub is created in a specific programming language it can be reused by applications in that language and a few lines of code are needed to call any AutoVue Web Services method through the client proxy.

Several AutoVue Web Services methods accept options (for example, print and convert) as an optional input parameter. The structures of these options are defined in a XML Schema Definition (XSD) that is linked to the WSDL and are generated in the client stub code automatically. The client application instantiates these options and sets their variables to desired values and invokes the AutoVue Web Services method.

A successful call to an AutoVue Web Services method returns the desired output. An error message is displayed if there is an issue with the input parameters.

Note: The output types of AutoVue Web Services methods vary from one to another. Regardless, all custom output structures are defined in the XSD and are generated automatically once the client stub is generated.

---

9 Using AutoVue Web Services

This chapter discusses how to use AutoVue Web Services and the steps involved to create client proxy.

9.1 How to use AutoVue Web Services

The first step in using AutoVue Web Services is to create a client proxy in your desired language. Then, after installation and deployment of AutoVue Web Services, look for the URL that points to the WSDL (for example, http://host:port/AutoVueWS/VueBeanWS?wsdl). This URL is needed for any utility that you use to create your client stub.

9.1.1 Java Client Proxy

There are two steps in generating a Java client proxy:

1. Generating Client Proxy Using WSimport
2. Importing and Using Client Proxy

9.1.1.1 Generating Client Proxy Using WSimport

To generate the Java client proxy, you can simply call *wsimport,* which is bundled with JDK, from the command line with the *-keep* option and pass the WSDL's URL.

For example:

```
wsimport -keep http://host:port/AutVueWS/VueBeanWS?wsdl
```

This will provide the following output:

```
parsing WSDL...
generating code...
compiling code...
```

After returning back to the command line, a new Java package is created in the current location.

The directory structure of the package should be *com\oracle\autovue\services.* All client proxy codes are generated inside this directory.

For detailed information regarding available wsimport options refer to the following link:

https://docs.oracle.com/javase/7/docs/technotes/tools/share/wsimport.html

9.1.1.2 Importing and Using Client Proxy

The next step is to import and instantiate the generated package inside your client code.The following code demonstrates how to call an AutoVue Web Services method:

```
import com.oracle.autovue.services.*;

public class AutoVueWSClient {

   public static void main(String[] args) throws Exception {

   //create service
       VueBeanWS_Service service = new VueBeanWS_Service();

       //create proxy
       VueBeanWS proxy = service.getVueBeanWSPort();

       //call autovue ping Web method
       System.out.print (proxy.ping("Hello from Java") );
 }
}
```

The first line, `import com.oracle.autovue.services.*;` imports the AutoVue client stub package generated by the wsimport tool. To call a Web method, you need to first instantiate the *VueBeanWS_Service* object. From this object, instantiate the *VueBeanWS* class which is the proxy for calling all AutoVue Web Services methods.

The simplest AutoVue Web Services method to run--which is also a good method for testing--is the *ping* method. It verifies that AutoVue Web Services is running and responding correctly.

After running the above code, you should receive an output similar to the following:

```
Server Date/Time: //some number showing current time at the server side
AutoVue Client Build Number: //...
AutoVue Client Build Date: //...
```

Note: Optionally, you can pass a string value to the ping method.

For a detailed description on calling AutoVue Web Services methods, refer to Testing AutoVue Web Services.

9.1.2 .NET Client Proxy

This section contains the following:

- Generating Client Proxy using WSDL
- Importing and Using Client Proxy in Microsoft Visual Studio

9.1.2.1 Generating Client Proxy using WSDL

The .NET environment also provides a tool for creating an AutoVue Web Services client proxy, *wsdl.exe*. To generate the AutoVue Web Services client proxy, from the command line, you can simply pass WSDL's URL to wsdl.exe.

For example:

```
wsdl.exe http://host:port/AutoVueWS/VueBeanWS?wsdl
```

The tool generates a file, *VueBeanWS.cs,* in the same location as *VueBeanWS?wsdl.*

Note: If you want the information being sent between a client and AutoVue Web Services to be secure, you should enter the HTTPS protocol in the URL instead of HTTP. For more information, refer to HTTPS/SSL.

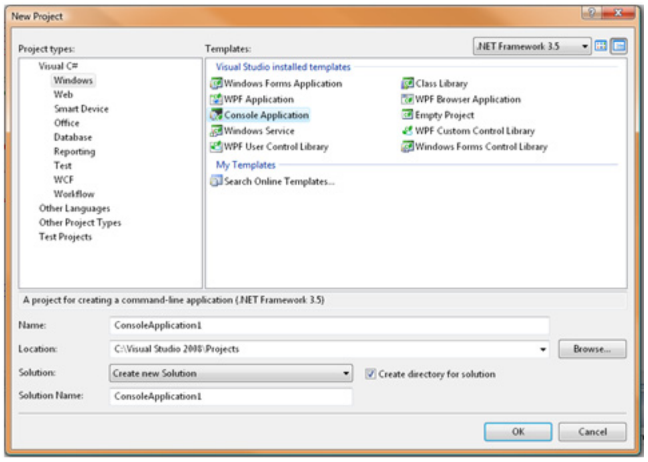
Since Microsoft Visual Studio is the primary IDE for .NET development, you can also use it to create and use the AutoVue Web Services client proxy. For more information on using Microsoft Visual Studio for generating a client proxy, depending on your environment, refer to Importing and Using Client Proxy in Microsoft Visual Studio.

9.1.2.2 Importing and Using Client Proxy in Microsoft Visual Studio

You can generate the AutoVue Web Services client proxy without using the command line in Visual Studio.

1. After starting Visual Studio, create a new console application (optionally choose the name AutoVueWSClient). As shown in the following figure, C# is the preferred coding language.

Figure 9–1 AutoVueWSClient



.NET Framework 3.5

A project for creating a command-line application (.NET Framework 3.5)

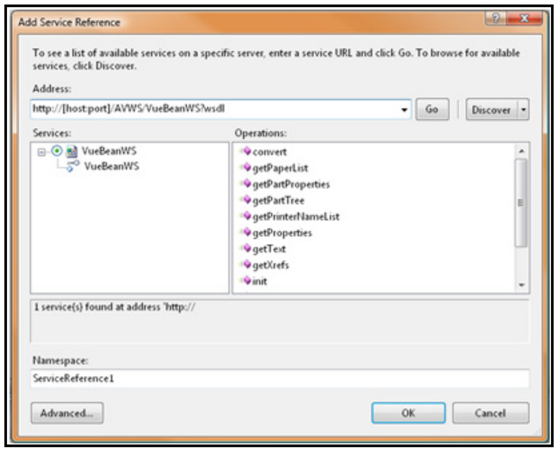
Name: ConsoleApplication1
Location: C:\Visual Studio 2008\Projects
Solution: Create new Solution
[✓] Create directory for solution
Solution Name: ConsoleApplication1

2. In the newly created project, from the Solutions Explorer window, right-click on **References**, and then select **Add Service References**.

   The Add Service Reference window appears.

3. Enter the AutoVue WSDL's URL in the URL field, then click **Go**. The VueBeanWS Web service and its Web methods are displayed in the Add Service Reference window.

Figure 9–2 Add Service Reference


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

http://[host:port]/AVWS/VueBeanWS?wsdl

Services:

- VueBeanWS
  - VueBeanWS

Operations:

- convert
- getPaperList
- getPartProperties
- getPartTree
- getPrinterNameList
- getProperties
- getText
- getXrefs
- init

1 service(s) found at address 'http://

Namespace:

ServiceReference1

4. Optionally, provide a new Namespace (for example, AutoVueWS), and then click **OK**. The proxy code is generated and added to your project.

5. On the Solution Explorer window double-click on the **app.config** file to be opened for editing.

6. Inside the file, locate `messageEncoding="Text"` which is part of the attributes for this binding: `<binding name="VueBeanWSPortBinding"` under the `<basicHttpBinding>` .

7. Change the value of the attribute to `Mtom`. That is, `messageEncoding="Mtom"`

8. Optionally, increase the values of the `maxBufferSize` and the `maxReceivedMessageSize`. This is useful when using the `convert()` method for converting large files, because the conversion result is returned in binary format attached to the response. At this point, you can import the proxy to your application (for example, Program.cs) and call the AutoVue Web Services methods. The following code demonstrates a sample C# code that calls the ping Web method:

```
using System;
using AutoVueWSCSClient.AutoVueWS;
namespace AutoVueWSCSClient
{
  class Program
  {
    static void Main(string[] args)
       {
           try
           {
                VueBeanWSClient vuebean = new VueBeanWSClient();
                Console.Write(vuebean.ping("Hello from C#"));
           }
           catch (Exception e)
           {
                Console.Write(e);
           }
       }
  }
}
```

As with the ping Web method, you can call all other VueBeanWS Web methods by passing them input parameters.

For more information on input/output parameters, refer to the AutoVue Web Services methods descriptions in Testing AutoVue Web Services.

9.1.3 HTTPS/SSL

Security plays an important role in communication between applications. When it comes to Web services, this issue is even more critical. As a result, it is highly recommend to only use HTTPS protocols to call AutoVue Web Services.

To run and use AutoVue Web Services over SSL, you must first deploy AutoVue Web Services on a secure server, import the server certificate to your client environment, and then generate and use the client proxy in the same manner as described in Importing and Using Client Proxy. Additionally, for SSL, you must use a secure connection over HTTPS to generate and use the client code (for example, https://host:port/AutVueWS/VueBeanWS?wsdl).

If you are using Oracle Weblogic to deploy AutoVue Web Services, you can use the self-signed certificate that comes with the application server out of the box:

1. Export the certificate from Oracle Weblogic into a file. You can do so through a Web browser.
2. Import the certificate into your client machine.
3. Follow the instructions in Importing and Using Client Proxy to generate and use the client proxy.

Note: Make sure you provide the HTTPS address of the WSDL.

10 AutoVue Web Services and DMS Integration

In addition to standard protocols supported by AutoVue Server (such as http:// and ftp://) AutoVue Web Services architecture allows flexible communication with DMS integrations in the same way as passing a URI. As a result, the client can send information about a document that is inside a DMS repository to AutoVue Web Services. Additionally, if an existing DMS integration is already set up, AutoVue Web Services can communicate with the DMS integration and access the document in order to process the client's request.

As with standard protocols such as http and ftp, the AutoVue Web Services administrator defines a custom protocol for each DMS integration and assigns a properties file on the AutoVue Web Services server that contains connection information for that specific DMS integration.

For example, if a DMS integration protocol is defined with the name *DMS_Integration_1*, then a *DMS_Integration_1.properties* file contains the location information and any other static data that is needed to communicate with an existing DMS instance. Client code can easily call AutoVue Web Services and pass a valid DMS document ID, as well as use the term *DMS_Integration_1* as prefix (for example, DMS_Integration_1://dID=12345). Once AutoVue Web Services finds a match between the DMS integration protocol name in the request and a defined custom protocol (in this case, DMS_Integration_1), it treats the rest of the string as a document ID and passes it to the DMS instance.

Note: The name of the DMS integration protocol is arbitrary and can be configured in AutoVue Web Services. However, both associated properties files on the server and client code must use the same name.

Note: It is important for the administrator to use meaningful names for DMS Integration protocols to avoid any confusion on the client side. For example, *vuelinkUCM://,* and so on. Additionally, if more than one instance of the same DMS integration is setup with AutoVue Web Services, a numbering scheme is suggested. For example, vuelinkUCM1://, vuelinkUCM2://, and so on.

Because each DMS integration and related DMS repository follow different standards of addressing documents, the structure of the document ID varies from one DMS integration to another. It is important to follow the string representation of document IDs that are defined in this section.

The following sections demonstrate string representations of the document ID for these supported DMS integrations (assuming a custom DMS integration protocol is setup and registered with AutoVue Web Services by the server administrator):

10.1 VueLink for Oracle UCM/WCC

The string representation of a document ID in VueLink for Oracle WCC is as follows:

```
dID=some_id_number[&Markup_BasedID=some_id_number][&Format=some_format]
[&Extension=some file ext]
```

Where:

dID: The valid document ID of the desired document.

Markup_BasedID: The valid document ID of the base document (only meaningful and needed when the document ID belongs to an AutoVue Markup).

Format: The format of the document according to what is defined inside the Oracle WCC (optional, but it is needed when the document ID belongs to an XRef folio).

Extension: The filename extension of the document according to what is defined inside the Oracle WCC (optional, but needed when the document ID belongs to an XRef folio and Format is not included).

The following are examples of URI values when invoking AutoVue Web Services for an Oracle WCC document (assuming protocol name is vuelinkWCC):

```
vuelinkWCC://dID=227&Markup_BasedID=228
vuelinkWCC://dID=350&Extension=slddrw
vuelinkWCC://dID=270&Format=Application/dwg&Extension=dwg
vuelinkWCC://dID=253&Extension=xcsr
```

10.2 VueLink for Documentum

The string representation of a document ID in VueLink for Documentum is as follows:

```
WebTopURL?userName=some_name&docbase=some_docbase_name&sessionid=webtop_seession_
id&objectid=some_object_id&rendition=some_file_format
```

WebTopURL: The URL for webtop.

userName: A valid webtop UserName.

docbase: A valid docbase name.

sessionid: A valid webtop session ID.

objectid: A valid ID of an object in the above docbase.

rendition: A valid Documentum format.

The following are examples of URI values when invoking AutoVue Web Services for a Documentum document (assuming protocol name is vuelinkDocumentum):

```
vuelinkDocumentum://http://[host:name]/Webtop6?userName=Administrator&docbase=demo
&sessionid=s7&objectid=0901869f80002565&rendition=unknown
```

10.3 Third-Party Integration

You must construct a document ID for a file stored inside Third_Party_Name DMS using the Third_Party_Name protocol.

For example: Third_Party_Name://Third_Party_NameDocID=123&Format=dwg

Note:

- The prefix used in your document ID must match your properties filename (in this case, Third_Party_Name).
- To properly access files stored inside your Third_Party_Name DMS repository, the syntax of your document ID should match one that is understood by your DMS integration servlet.
- Invoke AutoVue Web Services on the document ID.

  For example: getXrefs()/getText()

10.3.1 AutoVue ISDK Integration Example

The string representation of document ID in AutoVue ISDK (filesys) is as follows:

```
RootURL/some_repository/some_file_name/some_file_name(some_version)/ some_file_
name
```

RootURL: The value defined for parameter RootURL in web.xml of ISDK (filesys).

some_repository: A valid repository name which the file belongs to.

some_file_name: A valid file name that exists in the repository.

some_version: A valid version number for the file.

The following are examples of a URI value when invoking AutoVue Web Services for an ISDK document (assuming that the protocol name is vuelinkISDK):

```
vuelinkISDK://http://localhost/filesysRepository/2D/AutoCAD.dwg/AutoCAD.dwg(1)/Aut
oCAD.dwg
vuelinkISDK://http://localhost/filesysRepository/3D/Hard Drive.CATProduct/Hard
Drive.CATProduct(1)/Hard Drive.CATProduct
```

Note:

- It is important that the prefix used in your document ID matches your properties filename (in this case, vuelinkISDK).
- No authentication is required in order to access files in AutoVue ISDK (filesys).
- Invoke AutoVue Web Services on the document ID.

  For example: getXrefs()/getText()

11 Oracle Web Services Manager

Oracle Web Services Manager (OWSM) is a component of the Oracle SOA suite. It can be used as a proxy for your AutoVue Web Services, and you can assign it different policies and rules to access AutoVue Web Services. Additionally, you can monitor accesses to your AutoVue Web Services and review different statistics and logs that are provided by OWSM.

OWSM can also be used to perform a simple test of AutoVue Web Services. It can generate an input from any AutoVue Web Services methods and invoke them through a Web browser. AutoVue Web Services can be easily tested using the Oracle Web Services Manager Test tool.

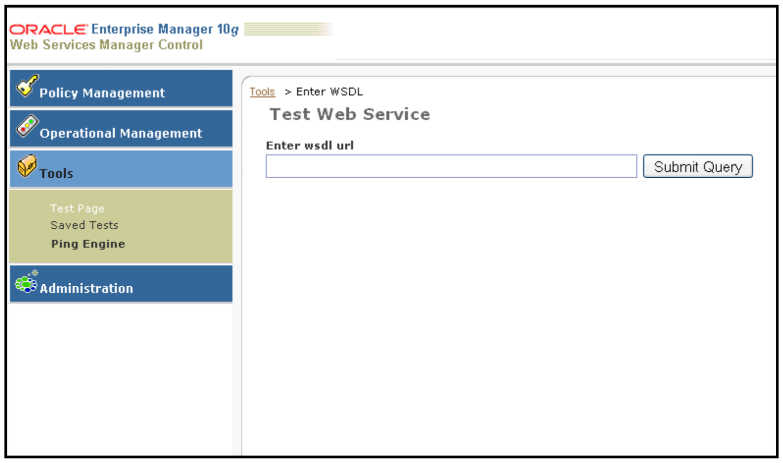
For more information about Oracle Web Services Manager, refer to the following URL:
`https://docs.oracle.com/middleware/1212/owsm/index.html`

12 Testing AutoVue Web Services

Oracle SOA Suite is used to test AutoVue Web Services.

1. Start Oracle SOA Suite.
2. As shown in the following screen shot, from the Web Services Manager Control page, click **Tools,** and then click **Test Page.**

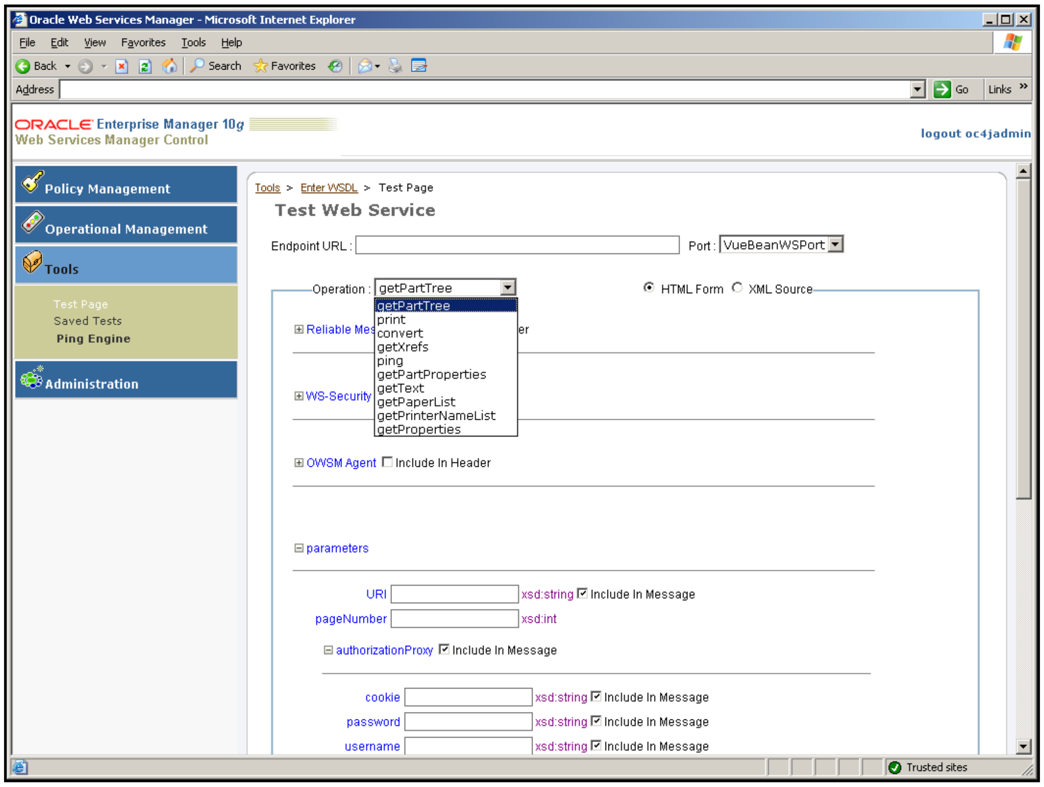
Figure 12–1 Oracle Enterprise Manager - Tools



3. Enter the AutoVue Web Services WSDL's URL in the **Enter wsdl url** text box. For example, `http://AVWSHost:7011/AutoVueWS/VueBeanWS?wsdl`
4. Click **Submit Query.**

   As shown below, the Test Web Service page reloads with an input form that is ready to invoke one of the AutoVue Web Services methods.

Figure 12–2 Oracle Web Services Manager



Tools > Enter WSDL > Test Page

Test Web Service

Endpoint URL : Port : VueBeanWSPort

Operation : getPartTree

getPartTree
print
convert
getXrefs
ping
getPartProperties
getText
getPaperList
getPrinterNameList
getProperties

[✓ HTML Form] XML Source

⊞ Reliable Mes... er

⊞ WS-Security

⊞ OWSM Agent ☐ Include In Header

⊟ parameters

URI		xsd:string	☑ Include In Message
pageNumber		xsd:int	

⊟ authorizationProxy ☑ Include In Message

cookie		xsd:string	☑ Include In Message
password		xsd:string	☑ Include In Message
username		xsd:string	☑ Include In Message

5. From the **Operation** list, select a Web method.

   For information on the available Web methods, refer to AutoVue Web Services Methods.

12.1 AutoVue Web Services Methods

The following table provides a summary of the available AutoVue Web Services methods. After selecting the method and entering the required information, click **Invoke** to send the request to the Web Services provider.

Note: If the method calls a file from inside the DMS repository that requires authentication, you must provide the required credentials.

Table 12–1 Web Methods

Web Method	Description
getPartTree	This part tree extraction Web method returns a list of parts contained in a given file. This is applicable only for 3D formats. For example, in case of 3D assembly, this Web service returns a list of parts and sub-assemblies referenced by the 3D assembly. From the Operation list, select **getPartTree** and wait for the page to refresh. **1.** To invoke this service, enter a valid URI in the **URI** text box. **2.** If the URI is a VueLink DocID, or an address that needs authentication, you should also enter the username/password and/or cookie depending on what is required. **3.** In the **pageNumber** field, enter a value less or equal to the number returned by getProperties. **Note**: The dmsArguments section is optional and is only needed if required by a VueLink. To add more DMS arguments, click the plus symbol.
print	This printing Web method sends a given file to a printer for printing. From the Operation list, select *print* and wait for the page to refresh. The print options for the print Web method are divided into three groups: **WSPrintOptions** This option provides the following options: - Specify page range. - Choose one of the available paper sizes on the target printer. These values can be retrieved by calling getPaperList and passing the printer name. - Select printOrientation {ORIENTATION_PORTRAIT, ORIENTATION_LANDSCAPE, ORIENTATION_AUTO} - Select printPageType {PAGES_ALL, PAGES_CURRENT, PAGES_RANGE} - Specify printer name. The available values can be retrieved by calling getPrinterNameList. - Flag indicating whether the blank pages should be skipped. - Flag indicating whether force all colors to black. - Specify the layers to print for specified pages. Page number, layer id and layer name are mandatory. If no layer information is provided, then the file's default layer settings define which layers to print. **WSPrintHeader** This option allows you to specify the position of the text to be added to the header and footer of the printed page (left, right, and/or center). **WSPrintWaterMark** This option provides the following options: - Specify the text to be added as watermark to the printed page. - Select the orientation of the watermark {DIAGONAL, HORIZONTAL, VERTICAL} **openAllMarkups** A boolean flag that indicates if the markups of the document must be printed with the document.

Table 12–1 (Cont.) Web Methods

Web Method	Description
packetPrint	Prints a group of documents (known as packets) one at a time, along with the auto-generated cover page and summary page. From the Operation list, select **packetPrint** and wait for the page to refresh. The following is the list of packetPrint input parameters: **URIs** A list of URIs that belong to same packet. (mandatory) **PacketID** A string representing the ID of the packet. (mandatory) **PacketIDLocation** One of the six possible locations for packetID to appear on every page. From the printoutLocation enum, it is a combination of (top/bottom) + (left/center/right). It is an optional input. The PacketID is printed on the cover page and summary page regardless. **FileIDLocation** One of the six possible locations for File ID (file number in the packet) to appear on every page. From the printoutLocation enum, it is a combination of (top/bottom) + (left/center/right). It is an optional input. **WSPacketPrintOptions** This option applies to all documents in the packet. It is an optional parameter and provides the following: - Choose one of the available paper sizes on the target printer. These values can be retrieved by calling getPaperList and passing the printer name. - Select printOrientation {ORIENTATION_LANDSCAPE, ORIENTATION_AUTO, ORIENTATION_PORTRAIT} - Specify printer name. The available values can be retrieved by calling getPrinterNameList. If this option is not specified, then the default printer is used. - Flag indicating whether to force all colors to black (grayscaled). - Specify the layers to print for specified pages. Page number, layer id and layer name are mandatory. **WSPrintHeader** This option allows you to specify the text to be added to the header and footer of the printed page. **WSPrintWaterMark** This option provides the following options: - Specify the text to be added as watermark to the printed page. - Select the orientation of the watermark {DIAGONAL, HORIZONTAL, VERTICAL} **openAllMarkups** A boolean flag that indicates if the markups of the document must be printed with the document.
getXrefs	This External References (XRefs) Web method returns a list of XRefs associated to a given file. From the Operation list, select **getXrefs** and wait for the page to refresh. This method only requires a valid URI. Authorization is needed only if the URI cannot be accessed without it.

Table 12–1 (Cont.) Web Methods

Web Method	Description
getLayerInfo	This Web service allows a user to obtain information about all available layers of a given document. From the Operation list, select **getLayerInfo** and wait for the page to refresh. This method only needs a valid URI. Authorization is needed only if the URI cannot be accessed without it.
getPartProperties	This part level metadata extraction Web method returns a list of all metadata for a given part in a given document. This method is only supported for 3D formats. For example, in the case of a 3D assembly, this Web method returns properties of a particular part referenced by the 3D assembly. **1.** From the Operation list, select **getPartProperties** and wait for the page to refresh. **2.** This method needs a valid URI and a valid entityID. The valid entityIDs are retrieved by calling the getPartTree method and passing the same URI. Authorization is needed only if the URI cannot be accessed without it. **3.** In the **pageNumber** field, enter a value less or equal to the number returned by getProperties.
getText	This text extraction Web method returns visible text inside a given document. This method is not supported for 3D formats. Metadata is not included by this method (for example, EDA entity information, layer and block names, and so on). From the Operation list, select **getText** and wait for the page to refresh. This method only needs a valid URI. Authorization is needed only if the URI cannot be accessed without it.
getPaperList	This utility Web method returns the paper sizes for a given printer that are available to AutoVue. From the Operation list, select **getPaperList** and wait for the page to refresh. This method only needs a valid printer name. Valid printer names can be retrieved by calling getPrinterNameList.
getPrinterNameList	This utility Web method returns a list of available printers. From the Operation list, select **getPrinterNameList** and wait for the page to refresh. This method does not need an input parameter.

Table 12–1 (Cont.) Web Methods

Web Method	Description
convert	This conversion Web method converts a given file into another format such as BMP, PDF, or TIFF. It only supports one page at a time. From the Operation list, select **convert** and wait for the page to refresh. This method can be called without including the option section. In this case, the default options use the bitmap version of the document in its original size. If you set *openAllMarkups* to TRUE, Web Services retrieves all markups and includes them in the converted output. In a non-integrated these markups are retrieved from the Markups folder of the AutoVue server. When integrated with a DMS, all markups returned by the DMS are included in the converted output. If you include **convertOption**, you can: ■ Specify the color depth value. ■ Select the output format {BMP, PDF, TIFF} ■ Specify the page (only one page at a time is supported). Note that with PDF format, regardless of the page setting, all pages are converted together. ■ Select the *convert* scale {TYPE_SIZE, TYPE_SCALE} ■ Specify the *height* and *width* in pixels (if TYPE_SIZE Scale is selected). ■ Specify the *scaleFactor* and *stepsPerInch* (if TYPE_SCALE is selected). ■ Specify if it is a rendition to be saved back to the repository. If set to TRUE, then no convert data is returned to the caller and it is sent to the repository.
getProperties	This file level metadata extraction Web method returns metadata and properties for a given file. From the Operation list, select **getProperties** and wait for the page to refresh. This method only needs a valid URI. Authorization is needed only if the URI cannot be accessed without it.

12.2 AutoVue Web Services API

The JavaDoc index (located in AutoVue webservices default installation directory C:\Oracle\AutoVueWS\docs) provides a complete reference to all classes and APIs inside the AutoVue Web Services package. The **com.oracle.autovue.services** package contains all classes and sub-packages of AutoVue Web Services. All the AutoVue Web methods are defined inside the **VueBeanWS** class of this package.

The sub-package **com.oracle.autovue.services.options** includes all classes that represent custom input options for different AutoVue Web methods such as convert and print.

The sub-package **com.oracle.autovue.services.types** includes all classes that represent custom outputs for different AutoVue Web methods such as getText, getXrefs, and so on.

The sub-package **com.oracle.autovue.services.pool** includes pooling mechanisms used inside the AutoVue Web Services package.

A

Appendix A - Sample Client Code in Java

The following sample client code in Java calls all of the AutoVue Web methods with a predefined URL.

```
import java.io.FileOutputStream;
import java.util.List;
import com.oracle.autovue.services.*;

public class AutoVueWSClient
{
public static void main(String[] args) throws Exception{
  //Create Service
  VueBeanWS_Service service = new VueBeanWS_Service();

  //Create proxy
  VueBeanWS proxy = service.getVueBeanWSPort();

  //Call AutoVue ping Web method.
  System.out.print (proxy.ping("hello") );

  String URL =
"https://www.oracle.com/us/products/applications/autoVue/057065.pdf";
  //Call the convert Web method.
  try{
ConvertOption option = new ConvertOption();
option.setFormat(Format.BMP);
option.setPage(1);
option.setScaleType(ScaleType.TYPE_SIZE);
option.setHeight(600);
option.setWidth(800);
byte[] file = proxy.convert(URI, option, null, false);
FileOutputStream fos = new FileOutputStream("c:/temp/output1.bmp");
fos.write(file);
fos.close();
 }
  catch(Exception e){
  e.printStackTrace();
}
//Call the getPrinterNameList Web method.
List<String> printers = proxy.getPrinterNameList();
for (String printer : printers) {
System.out.println("Printer Name: "+printer);
System.out.println("Available Papers on this Printer");
//Call the getPaperList Web method
List<String> papers = proxy.getPaperList(printer);
for (String paper : papers) {
  System.out.println("Paper Name: "+paper);
  }
 //Call the getProperties Web method.
List<MetaProperty> properties = proxy.getProperties(URI, null);
for (MetaProperty prop : properties) {
System.out.println( prop.getName() + "=" +prop.getValue());
//Call the getText Web method.
List<SearchText> texts = proxy.getText(URI, null);
for (SearchText text : texts ) {
System.out.println("\nPage Number:"+ text.getPageNumber());
List<String> txts = text.getTexts();
for (String txt : txts) {
System.out.print(txt);
}
}
//Call the getXrefs Web method
List<XrefsInfo> xrefs = proxy.getXrefs(URI, null);
for (XrefsInfo xref : xrefs ) {
System.out.println("Name:"+xref.getDocName() + " " + "docID:" + xref.ge
DocID());
}
//Assuming URI is a 3D document. Call the getPartTree Web method.
int pageNum = 4;
PartTreeResult parts = proxy.getPartTree(URI,pageNum,null);
//Call getParts Web method.
List<PartInfo> info = parts.getParts();
for (PartInfo part : info) {
System.out.println("Part Name :"+part.getName() + " - Part ID:" +
part.getID()+" - Part Type:" + part.getType());
List<PartMetaProperty> metaProps = proxy.getPartProperties(URI, pageNum,
part.getID(), null);
for (PartMetaProperty meta : metaProps) {
System.out.println( meta.getName() + "=" meta.getValue());
     }
   }
}
```

A.1 Web Services Sample Client Code for Printing

AutoVue Web Services provides a sample Web Services client code, SampleClient.java, which demonstrates how to call Web Services' print() method. It is located under the <AutoVue Web Services Installation Directory>\autovue_webservices\sample_client directory. You can make the following modification according to your needs:

- Specify the username and password if the file has restricted access. For example, this is needed when storing a file in DMS.
- Specify more print options, watermark options, and header/footer options.
- If an error message containing the string ERROR_00 appears when the client calls the Web Services print() method, then the Web Services cannot process the request due to following reasons:
  - Server is too busy. No VueBean is available to process the request.
  - Not enough memory is available for a VueBean to open a file.

    To resolve this issue, the client must call the *print()* method later. In the SampleClient.java file, the client waits for one minute (60000 milliseconds) to call again.

- The following error messages appear when layer information is invalid:
  - ERROR_005: The page specified in LayersInfo object does not have layer.
  - ERROR_006: The layer information specified in LayersInfo object is not correct.
  - ERROR_007: There is no layer information in LayersInfo object for the specified page number.

A.2 Packet Printing

AutoVue Web Services provides a sample Web Services client code, SamplePacketPrintClient.java, which demonstrates how to call Web Services' `packetPrint()` method. The sample client code is located under <AutoVue Web Services Installation Directory>\autovue_webservices directory.

Note: In the `packetPrint()` method only the list of the documents and the packet ID are mandatory. Other parameters are optional. If you want the packetID to be printed on every page of all documents, then you must specify a print out location. Same is true for file counter (the file ID of each document in the packet is printed if a location is specified).

If no packet print option is defined or if no printer is set in that object, then the default printer on the AutoVue Web Services machine is used automatically.

The output of the `packetPrint()` method includes an auto-generated cover page at the beginning of the packet print out and a summary page at the end.The summary page includes the success/fail status of each document in the packet. For this reason, the `packetPrint()` method does not return until the last document in the packet is processed.

B

Deploying AutoVue Web Services on Managed Server of Oracle WebLogic

In order to achieve better performance, it is recommended to deploy AutoVue Web Services and the VueServlet on different servers of the same WebLogic domain.

The following steps illustrate how to deploy AutoVue Web Services on a managed server of Oracle WebLogic. It is assumed that you have already created managed servers for deploying AutoVue Web Services and VueServlet before starting the following steps. For more information on how to create a managed server, refer to Oracle WebLogic documentation.

1. Access the administration console of the Oracle WebLogic.
2. Enter the administration user name and password.
3. In the left pane, click **Deployments**.
4. If you already have AutoVue Web Services and VueServlet applications deployed, undeploy them by selecting **Delete**.
5. To deploy the newly assembled AutoVue Web Services, click **Install**.
6. Select the WAR file for deployment by navigating to the <AutoVue Web Services Installation Directory>\autovue_webservices\ directory and select AutoVueWS.war.
7. Click **Next**.
8. Select **Install this deployment as an application**.
9. Click **Next**.
10. Select which target server you want to deploy AutoVue Web Services on.
11. Click **Next**.
12. Modify the default value as you want, and then click **Finish**.

For more information on verifying the deployment of AutoVue Web Services, refer to Verification

Repeat the steps 3 through 12 in order to deploy VueServlet.war on Oracle WebLogic.

C
Troubleshooting

This appendix contains information on how to troubleshoot common errors.

SoapUI Client Consumes AutoVue Web Services

If the soapUI client consumes AutoVue Web Services, you will receive the following error message:

```
Error getting response; java.net.SocketTimeoutException:Read timed out
```

To resolve this issue, change the Socket Timeout setting:

1. From the **File** menu, select **Preferences,** select **HttpSettings,** and then select **Socket Timeout.**
2. Assign a larger number to Socket Timeout value.

Error: "Failed to access the WSDL at: ..."

This error means either that the application server that hosts AutoVue Web Services is not running or that the client code is not pointing to the right location. Make sure that the application server is running and, if necessary, regenerate the client proxy code as described in the Generating Client Proxy Using WSimport.

Error: "Internal object is null. Make sure AutoVue server is running."

If you encounter this error message, make sure AutoVue server is running and listening to the correct port, and that any firewall between AutoVue server and AutoVue Web Services is configured to allow communication. The AutoVue server host:port should match the value defined in the web.xml file inside the AutoVue Web Services package ("initialJVueServer").

Error: "Cannot get file: File not found."

This error can occur in different scenarios:

- If an authentication is required to access the document, make sure they are defined in your client code and bundled to your request.
- If a DMS integration protocol is involved, make sure the protocol name in the URI and the one defined in the AutoVue Web Services configuration are exactly the same. Verify the connection information inside the properties file that is associated with the DMS integration is running and accessible from the Web Services package.

  Additionally, ensure that the document ID is valid and that the user has permission to access the document.

Error: "Cannot get metadata for entity" when calling getPartProperties() method

If you get the "Cannot get metadata for entity" message when entering the correct entity ID for the getPartProperties() Web method, you must call the getPartTree() method to get a new entity ID. This is because the first call to the *getPartTree()* method loads the file from the native file and also triggers the generation of a streaming file. Additionally, when you call the *getPartProperties()* method, the file is loaded from the streaming file. The entity IDs for files that load from the native file are different from those that load from the streaming file.

Error: Installer Crash on Linux or Logs Errors

If the Linux installer crashes or creates error logs, verify that the environment variable LC_ALL is set to en_US. If you cannot change the variable or if it does not resolve the problem, then run the installer in console mode as an alternative.

For example: `./setupLinux.bin -console`

D Feedback

If you have any questions or require support for AutoVue please contact your system administrator.

If at any time you have questions or concerns regarding AutoVue, please contact us.

D.1 General AutoVue Information

Web Site	`http://www.oracle.com/us/products/applications/autovue/index.html`
Blog	`http://blogs.oracle.com/enterprisevisualization/`

D.2 Oracle Customer Support

Web Site	`http://www.oracle.com/support/index.html`

D.3 My Oracle Support AutoVue Community

Web Site	`https://communities.oracle.com/portal/server.pt`

D.4 Sales Inquiries

E-mail	`https://www.oracle.com/corporate/contact/global.html`