

# **Oracle® AutoVue Integration SDK**

Overview and Installation Guide

Release 21.1.0.5

**F86927-06**

February 2026

Overview and Installation Guide, Release 21.1.0.5

F86927-06

Copyright © 1998, 2026, Oracle and/or its affiliates. All rights reserved.

Primary Author:

Contributing Author:

Contributor:

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

---

# Contents

<b>Preface</b> .....	v
----------------------	---

## 1 ISDK Overview

1.1	Introduction .....	1-1
1.2	AutoVue and Repository Integration .....	1-1
1.3	GUI Customization .....	1-3
1.4	Repository Extension .....	1-3
1.5	VueLink .....	1-4
1.6	Optional Components .....	1-4
1.6.1	CAD Connector .....	1-4
1.7	Overview of ISDK Components .....	1-5
1.7.1	Documentation .....	1-5

## 2 System Requirements

2.1	Required Software .....	2-1
2.2	Server .....	2-1
2.2.1	Windows .....	2-1
2.2.2	Linux .....	2-1
2.3	Client .....	2-1
2.4	Application Servers .....	2-1
2.5	Development Tools .....	2-2

## 3 Installation

3.1	Downloading Required Software .....	3-1
3.1.1	Oracle AutoVue .....	3-1
3.1.2	WebLogic Server .....	3-1
3.1.3	Oracle Enterprise Pack for Eclipse .....	3-1
3.1.4	Oracle JDeveloper 11gR1 .....	3-2
3.2	Installing and Configuring .....	3-2
3.2.1	Installing ISDK .....	3-2
3.2.2	Creating a Server Runtime Environment on IDE .....	3-3
3.2.3	Creating Projects on IDE .....	3-4
3.2.4	Configuring ISDK Components .....	3-5
3.3	Configuring Sample Components .....	3-7

## 4 Configuring Sample Projects

4.1	Sample Integration for Filesys DMS .....	4-1
4.1.1	Step 1: Copy the AutoVue Jar Files .....	4-1
4.1.2	Step 2: Configure the AutoVue Server.....	4-1
4.1.3	Step 3: Configure log4j.properties for Debugging .....	4-2
4.1.4	Step 4: Configure RootDir for the Filesys Repository .....	4-3
4.1.5	Step 5: Configure for an Embedded or Pop-Up Window (Optional) .....	4-3
4.1.6	Step 6: Configure the Markup Policy (Optional) .....	4-3
4.1.7	Step 7: Configuring User Control.....	4-4
4.1.8	Step 8: Configure the Picklist.....	4-4
4.1.9	Step 9: Configure the Thumbnail Display .....	4-4
4.1.10	Step 10: Configure for Redirection.....	4-5
4.1.11	Step 11: Configure the Real-Time Collaboration (RTC) Demo.....	4-5
4.1.12	Step 12: Configure the Oracle Enterprise Visualization Framework (OEVF).....	4-6
4.1.13	Step 13: Configure New Sample Data.....	4-8
4.1.14	Step 14: Run the Filesys Project.....	4-14

## 5 Implementation

5.1	ISDK Skeleton Project.....	5-1
5.2	Sample Projects.....	5-3
5.2.1	Sample Integration for Filesys Project .....	5-3
5.3	Implementation .....	5-8
5.3.1	Phase One .....	5-8
5.3.2	Phase Two.....	5-8
5.3.3	Phase Three.....	5-8

## 6 Deployment of ISDK-Based Integrations

6.1	Scaling for High Usage over Distributed Environments .....	6-1
-----	--	-----

## A Updating Existing Integrations to the Java Web Start Client

A.1	Update your Integration .....	A-1
A.1.1	Update the server .....	A-1
A.1.2	Setup the server for SSL Mode.....	A-2
A.1.3	Test AutoVue Sample.....	A-3
A.1.4	Specify Cookies.....	A-4
A.1.5	Update client side code .....	A-4
A.1.6	Customizing AutoVue .....	A-5
A.1.7	AutoVue Constructor Parameters .....	A-5
A.2	Steps for Integration.....	A-6

## B Feedback

B.1	General AutoVue Information .....	B-1
B.2	Oracle Customer Support.....	B-1
B.3	My Oracle Support AutoVue Community .....	B-1
B.4	Sales Inquiries .....	B-1

---

# Preface

The *Oracle AutoVue Integration SDK Design, Installation and Configuration Guide* provides a high-level overview of the Oracle AutoVue Integration Software Development Kit (ISDK) and also describes the procedure for building and running a dynamic Web project in JDeveloper and Eclipse IDEs for Oracle AutoVue.

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 Oracle partners and third-party developers (such as integrators) who want to implement their own 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 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 AutoVue Integration SDK library on OTN:

- ▣ *Technical Guide*
- ▣ *Acknowledgments*
- ▣ *Javadocs*
- ▣ *Security Guide*

## Conventions

The following text conventions are used in this document:

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

# ISDK Overview

This chapter provides a general overview of the AutoVue Integration SDK, and is intended for Oracle partners and third-party developers (such as integrators) who want to create an integration between Oracle AutoVue and a content repository.

## 1.1 Introduction

The AutoVue Integration Software Development Toolkit (ISDK) is intended for third-party developers who want to integrate Oracle AutoVue with their Data Management System (DMS).

**Figure 1–1 AutoVue ISDK work flow**



Oracle AutoVue is a thin client viewing and collaboration solution for enterprise-wide data access.

The ISDK installation package includes four sample projects that can be modified to suit your integration needs. The following sections describe these projects in more detail.

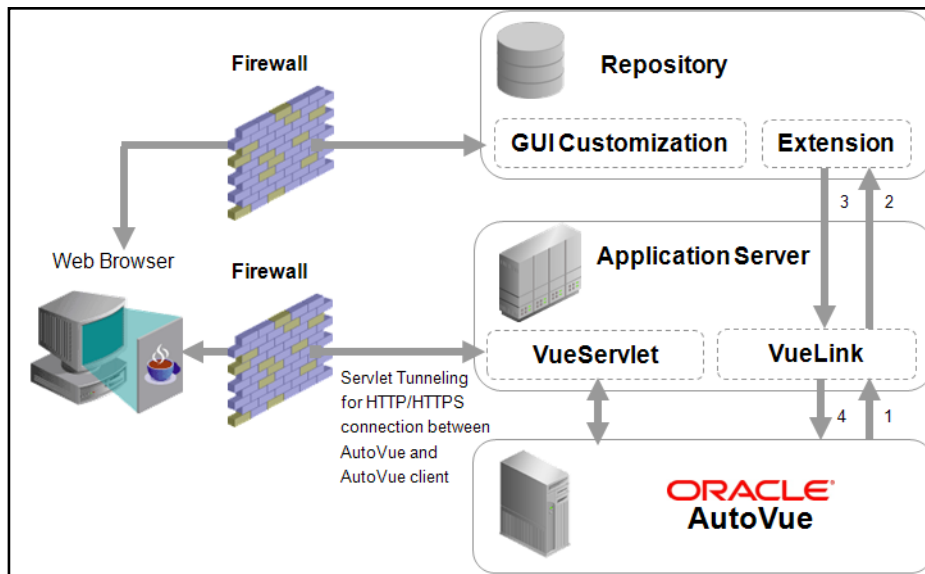
## 1.2 AutoVue and Repository Integration

AutoVue is the key component in Oracle's Enterprise Visualization solutions. AutoVue solutions deliver native document viewing, markup, and real-time collaboration capabilities that streamline the information flow and collaborative processes across the global enterprise. AutoVue solutions help organizations in a variety of industries including Utilities, Industrial Manufacturing, Electronics & High Tech, Engineering and Construction, Aerospace and Defense, Automotive, and Oil & Gas. AutoVue streamlines visualization and collaboration across the global enterprise, improves productivity, reduces errors, and accelerates innovation and time to market. In an enterprise, AutoVue can be part of many business workflows and use cases such as collecting comments and annotations during a design review, recording the actions and results for a maintenance work order, comparing archived documents, and collaborating with other users.

AutoVue can offer its capabilities to many different enterprise systems/repositories such as DMS, PLM, and Content Management Systems (CMS). AutoVue needs to be integrated into these repositories in order to be able to access the documents that are stored in them. There exist many such integrations. For example, there is an integration between AutoVue and WebCenter Content (WCC) and AutoVue and Oracle Agile PLM. The Oracle-developed

integration is known as a VueLink. The VueLink provides an interface that allows communication between the repository and AutoVue in order to retrieve documents and to store data that is generated by AutoVue for those documents (such as annotations). The VueLink is a Java Web application that is hosted on a Java Web application server. The following figure shows how the communication between AutoVue and the repository is done through a VueLink.

**Figure 1–2 Communication between AutoVue and repository/backend system through a VueLink**



**Note:**

- ❏ AutoVue sends a request to the VueLink.
- ❏ VueLink forwards the request to the repository.
- ❏ The repository sends a response back to the VueLink.
- ❏ VueLink forwards the response to the AutoVue server.

After AutoVue gets access to a document and other related data from the repository, it then streams the view of the document to the AutoVue client via the VueServlet. The AutoVue Client provides the user with an interface to manipulate document display and perform other operations.

As shown in the diagram, the repository contains two important components: the repository extension and the GUI customization.

In order for the VueLink to communicate with the repository there needs to be a component on the backend-side whose interface the VueLink understands. This component is known as the repository extension. For more information, refer to the [Repository Extension](#) section.

In a seamless integration, the AutoVue client should be launched from inside the repository user interface. The GUI Customization is applied to the repository user interface. For more information, refer to [GUI Customization](#) section.

The application server component of the diagram includes the VueServlet and the VueLink. The VueServlet is a Java Servlet that acts as a tunnel between the AutoVue server and the AutoVue client. The client makes requests using the HTTP/HTTPS protocol to the VueServlet



and the VueServlet communicates with the AutoVue server using its socket port. All the communication between the AutoVue client and the AutoVue server goes through the VueServlet. The VueServlet is available out of the box with AutoVue and can be easily deployed. Information on the VueLink is provided in [VueLink](#) section.

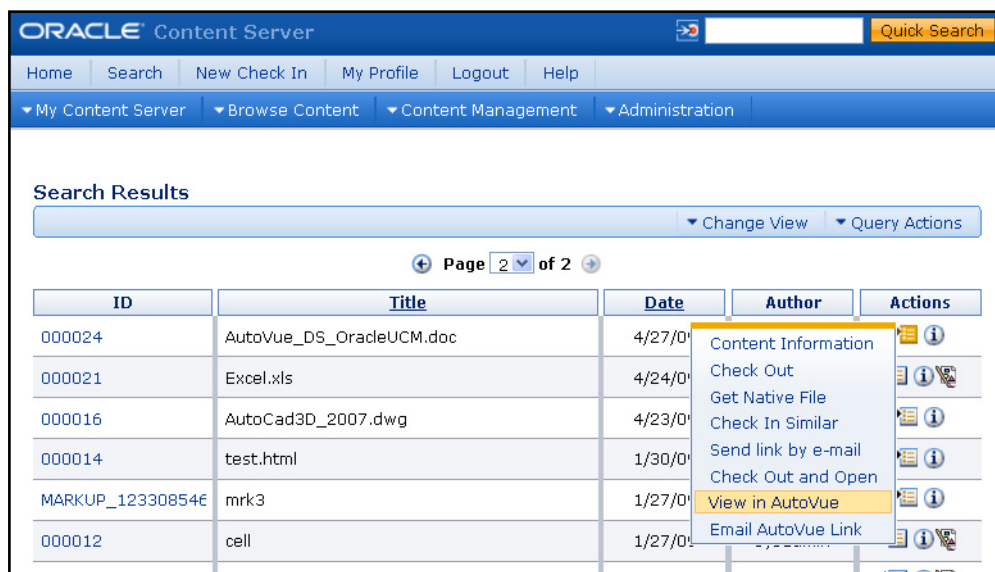
## 1.3 GUI Customization

Launching AutoVue should be implemented through the customization of the repository graphical user interface (GUI). The AutoVue client will be launched in a second window.

Depending on the underlying technology, the implementation of the GUI customization can vary from one environment to another. For example, a very simple implementation may be a hyperlink to an HTML page that loads the AutoVue client. A more sophisticated implementation may involve a repository-based scripting language or APIs. In both cases, you must refer to the repository's documentation for information on how to modify its UI and the available capabilities.

Since the customization can be applied to different places in the repository GUI, its implementation should be looked at from a usability point of view as well as from a technical point of view. A good example is customizing the search results page in the repository GUI. In this example, each document in the Search Results page is associated with a menu item or icon that launches the AutoVue client to view that particular document. A sample GUI customization is shown in the following screenshot. The Search Results page in Oracle Content Server GUI is customized to launch the AutoVue client.

**Figure 1–3 Sample GUI Customization**



## 1.4 Repository Extension

The repository extension is the layer on the repository that the VueLink communicates with. It allows the VueLink to access the repository the same way the end-user accesses the repository through the GUI.

---

**Note:** If the repository already provides a programming interface that gives access to all documents and related data required by the VueLink, then there is no need to develop a custom extension on the repository side for an AutoVue integration.

---

If this interface is not available, then a custom extension must be created using a technology that the repository supports. The extension must support the different requests that come from the VueLink. That is, the VueLink requests are related to retrieving and storing documents and their related data inside the repository. The extension can be built as a Web service or a Java Application Programming Interface (API). A Java API is the preferred approach as the performance is generally better and the overhead is lower in a Java to Java integration than in a Web service integration. This is because once the interface is provided by the repository then a VueLink-type component should be implemented to connect and communicate with it.

## 1.5 VueLink

The VueLink is the integration component that acts as the gateway between AutoVue and the repository. The name VueLink is reserved for these types of Oracle-developed gateway components. Third-party integrators and partners should choose their own trademarks or preferred name for this piece of integration. However, regardless of its name, VueLink-type components enable AutoVue to access documents that are stored inside the repository. It also enables AutoVue to retrieve any data related to these documents from the repository. In addition, any data generated by AutoVue (for example, markups and renditions) can be stored into the repository using this component.

The VueLink is the center piece of an AutoVue integration with a repository. It is able to communicate with AutoVue and with the repository, thereby acting as a translator for each end and isolating AutoVue and the repository from each other's complexity. It is a Java Web application and needs to be deployed on a Java Web application server (such as WebLogic, GlassFish, and Tomcat).

Since the interface between the VueLink and AutoVue is the same for all VueLinks (regardless of the repository they are built for), it is good practice to have an integration framework that has built-in communication with AutoVue and is ready to be used as a starting point for building new integrations with any repository. The AutoVue Integration SDK is designed to fulfill this requirement. For more information, refer to the [Configuring Sample Projects](#) chapter.

## 1.6 Optional Components

Before discussing the AutoVue Integration SDK, the optional components to be used in conjunction with building an integration are presented.

### 1.6.1 CAD Connector

One of the characteristics of CAD models is that often they are not stored in one document. For example, an airplane CAD model consists of many parts (such as wings, wheels, and so on) which in turn have their own subparts. Each part or subpart might be designed and stored in a separate document and referenced in the airplane CAD model document directly or hierarchically. In order to view that airplane CAD model, all parts must be loaded and put together. These separate parts and subparts files are known as external references (XRefs). AutoVue supports loading and viewing documents along with their XRefs documents. However, in an integration, the XRefs support should also be provided at the repository level since they are all stored inside the repository. If the repository provides a mechanism to link

documents to each other as references, then this mechanism can be used to provide XRefs support.

The storing and linking of references in a repository should not be done manually. In order to properly support the XRefs, some software tools should be provided that can import related documents from the CAD authoring software into the repository. An example is a CAD connector. A CAD connector is a software tool that integrates the repository with a CAD authoring software package (such as AutoCAD). It can check-in/check-out a set of related CAD files into/out-of the repository while preserving their relations and linkage.

---

**Note:** This software tool is not an AutoVue integration requirement. It is a facilitator for the repository to organize the XRefs.

---

## 1.7 Overview of ISDK Components

This section describes the various components included in the AutoVue Integration SDK.

### 1.7.1 Documentation

The following AutoVue Integration SDK documentation, with the exception of the *JavaDocs* which is included with the installation, can be found on the Oracle AutoVue Documentation Web site on the Oracle Technology Network (OTN)

<https://www.oracle.com/technetwork/documentation/autovue-091442.html>

▣ *Design, Installation and Configuration Guide*

- This guide provides a high-level overview of the Integration SDK. In addition, it contains information related to the installation, configuration and deployment of projects included in this ISDK in JDeveloper and Eclipse IDE.

▣ *Security Guide*

- This guide contains information related to the security and authentication mechanisms provided in this release of Integration SDK.

▣ *Release Notes*

- Details changes and enhancements made in this release of the ISDK.

▣ *Technical Guide*

- This guide contains in-depth technical information about the integration framework and describes how to implement your own integration based on the sample integration included in this ISDK.

▣ *Acknowledgments*

- This document lists licenses and third-party notices.

▣ *JavaDocs*

- This contains the JavaDocs of the underlying framework contained in the AutoVue Integration SDK. The *JavaDocs* is included the installation of this ISDK.



---

# System Requirements

The recommended system hardware configuration is:

- ▣ A system supporting the JDK/JRE version 8 with at least 8GB of main memory.
- ▣ At least 100MB of free disk space to install the software components and examples.

## 2.1 Required Software

- ▣ Oracle AutoVue 21.1.0.5

## 2.2 Server

The following operating systems have been certified with the Integration SDK:

### 2.2.1 Windows

- ▣ Windows Server 2022 64-bit
- ▣ Windows Server 2019 64-bit
- ▣ Windows 10, 7 64-bit
- ▣ Windows 11, 10 64-bit

### 2.2.2 Linux

- ▣ Redhat Enterprise Linux 7.X<sup>1</sup> (x86\_64) 64-bit, 8.x (x86\_64) 64-bit, 9.X (x86\_64) 64-bit.
- ▣ Oracle Linux 7.X<sup>1</sup> (x86\_64) 64-bit, 8.X (x86\_64) 64-bit, 9.X (x86\_64) 64-bit.

## 2.3 Client

The following Java Virtual Machines have been certified with the Integration SDK:

- ▣ Latest versions of Java (tested on Java 8u481 64 bit) for Filesys Sample and Skeleton.
- ▣ Web browsers supported by Oracle AutoVue 21.1.0.5.

## 2.4 Application Servers

The following application servers are compatible with the Integration SDK:

- ▣ Oracle WebLogic Server 12cR2

---

<sup>1</sup> Support for Linux 7 is deprecated in R21.1.0.5 and will be removed in a future release.

- Any other application server that supports Servlet 4.0.1 may work but are not certified by Oracle

## 2.5 Development Tools

The following IDEs are compatible with the ISDK:

- Oracle Enterprise Pack for Eclipse

---

## Installation

This chapter assumes you are familiar with Java development and with basic Web application development concepts, such as deployment descriptors and WAR archives. Understanding XML language is beneficial, but not mandatory.

The software products listed in [Chapter 2, "System Requirements"](#) must be installed and configured on your system according to the manufacturer's instructions.

### 3.1 Downloading Required Software

Before proceeding with the installation of the AutoVue ISDK, the following software must be installed and configured on your system according to the manufacturer's instructions.

#### 3.1.1 Oracle AutoVue

Download Oracle AutoVue 21.1.0.5 from ARU. The description name is Oracle AutoVue 21.1.0.5. Select the appropriate Media Pack for your development platform.

---

**Note:** Customers need to download GA Patch BUG #37718451 from ARU.

---

#### 3.1.2 WebLogic Server

You can download the Quick Installer for Developers version of the WebLogic Server from the Middleware section of <https://www.oracle.com/technetwork> Software Downloads. The download includes the installation instructions, as well as an executable JAR file that contains the installer.

#### 3.1.3 Oracle Enterprise Pack for Eclipse

Oracle Enterprise Pack for Eclipse (OEPE) is a free set of certified plug-ins, enabling WebLogic developers to support Java EE and Web Service standards. The Oracle Enterprise Pack for Eclipse All-In-One installed includes a pre-configured version of Eclipse and the OEPE plug-ins. The latest version of the installer is available from <https://www.oracle.com/technetwork> Software Downloads in the Developer Tools section.

If you download Eclipse IDE for Java EE Developers from the Eclipse Web site, you must download the Oracle WebLogic Server plug-in separately when creating the server.

### 3.1.4 Oracle JDeveloper 11gR1

You can download Oracle JDeveloper Studio Edition from

<https://www.oracle.com/technetwork/developer-tools/jdev/downloads/jdev11120download-495887.html>.

## 3.2 Installing and Configuring

This section describes the installation and configuration steps for the ISDK.

---

**Note:** If you are planning on deploying the ISDK in a secured environment, you should read the *Oracle AutoVue Integration Software Development Toolkit (ISDK) Security Guide* before installing the ISDK.

---

To install, unzip the media package to extract all necessary files. You must then create a server runtime environment on IDE and create a project. At this point you must manually configure the ISDK component such as the ISDK Skeleton.

Once these steps are complete, and the ISDK is extracted, you can configure the sample projects. For information on configuring the sample projects, refer to [Configuring Sample Projects](#).

### 3.2.1 Installing ISDK

The Oracle AutoVue SDK media pack is shipped as a zipped folder.

1. Enter the location and directory name for the AutoVue Integration SDK. The default location and name for Windows is C:\Oracle\AutoVueIntegrationSDK.
2. Select the components to install. By default, the ISDK Skeleton is selected. To install the sample projects, select Sample Integration (Filesys). The installation summary page appears.
3. The files are extracted to the location specified in step 1.

---

**Note:** The ISDK is installed by default with Secure Sockets Layer (SSL) enabled.

---

The *Quick Start.html* file found in the root folder is a top-level readme file that acts as an entry point to the rest of the ISDK documentation. To view the contents of this file, open it in your browser. After running the installer, all the required files are created under your AutoVueIntegrationSDK installation directory with the following structure:

- The /docs folder contains javadocs. All other ISDK documentation can be found on the Oracle AutoVue Documentation OTN site at <https://www.oracle.com/technetwork/documentation/autovue-091442.html>.
- The /FileSys folder contains four subfolders:
  - The /Repository folder contains filesysRepository.zip which contains sample files used by the Sample Integration for Filesys.
  - The /OEVF folder contains two GUI files used for the OEVF demo.
  - The /WebApplication folder contains a filesys.war file and a /filesys folder. The content in the /filesys folder is the unzipped version of the filesys.war file. The filesys.war can be imported into JDeveloper or Eclipse workspace to demo the Sample



Integration for Filesys and to demo RTC & OEVF functionalities. The project contains source code for sample integration, AutoVue client and third party libraries required by the integration.

- The /ESAPI\_Resources folder contains the OWASP Enterprise Security API properties files: AvESAPI.properties and validation.properties.
- The /ISDKSkeleton folder contains two sub folders:
  - The /WebApplication folder contains an isdk\_skeleton.war file and a /isdk\_skeleton folder. The content in the /isdk\_skeleton folder is the unzipped version of the isdk\_skeleton.war file. The isdk\_skeleton.war can be imported into JDeveloper or Eclipse workspace to create the Integration SDK Skeleton project. Your integration with Java-based backend systems will be developed based on this skeleton project and fulfill the TODO comments in this project.
  - The /ESAPI\_Resources folder contains the OWASP Enterprise Security API properties files: AvESAPI.properties and validation.properties.
- The /etc folder contains a list of files and folders structure contained in this ISDK, and folders containing licenses of third-party software used by the ISDK.

---

**Note:** The AvESAPI.properties and validation.properties files are placed in the folder based on the configuration settings defined by the user. If there is no path defined in the application, the library looks for them inside the esapi folder of the user's home directory. One way is to add the path to the project is: From **Project Properties**, select **Run/Debug/Profile** and then **Edit**. From **Edit**, select **Java Options**, and then add the path to the AvESAPI.properties and validation.properties files (e.g.:  
 -Dorg.owasp.esapi.resources=C:\temp\esapi). Make sure you copy the AvESAPI.properties and validation.properties to the location defined (e.g.: C:\temp\esapi). If there is no path defined in the application, the library looks for them inside the esapi folder of the user's home directory.

---

## 3.2.2 Creating a Server Runtime Environment on IDE

This section describes how to create a server runtime environment on JDeveloper and Eclipse IDEs.

### 3.2.2.1 Create Default Runtime on JDeveloper

JDeveloper has an integrated WebLogic Server (IntegratedWebLogicServer) configured. As a result, you can skip this step if using JDeveloper.

### 3.2.2.2 Create Server Runtime in Eclipse

You can create a server to identify the runtime environment that you want to use to test your Oracle AutoVue project. To create the WebLogic Server, complete the following steps:

---

**Note:** Your Oracle WebLogic Server domain needs to be created in development mode in order to create the server successfully in Eclipse.

---

1. From the **File** menu, select **New**, and then select **Other**.
2. Expand the Server folder, then select **Server**.
3. Click **Next**.

The Define a New Server wizard opens. This wizard lets you define a new server that contains information required to point to a specific runtime environment for local or remote testing, or for publishing to an application server.

4. Select "Oracle WebLogic Server" .%%%
5. Click **Finish**. The Oracle WebLogic Server will now be listed in the Servers view. You can start and stop the Server from this view.
6. Open the Server view to verify that the server has been created. You can click **Servers** or click **Window** from the menu bar, then **Show View** and **Servers** to display the Server view.

### 3.2.3 Creating Projects on IDE

This section describes how to create a project on JDeveloper and Eclipse IDEs.

#### 3.2.3.1 Projects on JDeveloper

1. Create an application if you do not have one yet. You can create an application by clicking **File** from the menu bar, then select **New**. The New Gallery dialog appears.
2. Select **Applications** under the **General** category from the left panel and then select **Custom Application** from the right panel.
3. Click **OK**. The Create Application dialog appears.
4. Complete the Create Application dialog to create an application with the Application Package Prefix field left empty. Click **Finish** to create the project.
5. Click **File** and then **Import**. The Import dialog appears.
6. Select **WAR File** and then click **OK**. The Create Project from WAR File dialog appears.
7. Browse to the ISDK component folder or sample projects folder and then select a WAR file.
8. In the following Create Project from WAR file dialogs perform the following:
  - ▣ Enter your project name.
  - ▣ Choose a directory to put your project.
  - ▣ Select the WAR file to import. For example, filesys.war.
  - ▣ Verify the location for Root Directory for Web Module.
9. Click **Finish** to finish the creation of your project.
10. In the Project view, browse to verify that your project has been created successfully.
11. Click **Build** to make your project. There should be no compilation error.
12. Check Libraries and Classpath:
  - a. Right-click the project and select Project Properties to bring out the Project Properties dialog
  - b. Click **Libraries and Classpath** in the left panel.
  - c. Check the JSP Runtime and JSF 1.2 are available under the Classpath Entries. If there are not available, you can add them manually in the following steps:

Click **Add Library** in the right panel.

Select **JSP Runtime** under Extension from the pop-up window.

If you are going to deploy the project later to an external WebLogic Server instead of using the IntegratedWebLogicServer, you also need to add JSF 1.2 under Extension from the pop-up window.

Click **OK**.

13. To start the WebLogic Server. You can click on **Run** from menu bar and then click **Start Server Instance** to start or click **On** from the toolbar.

### 3.2.3.2 Projects on Eclipse

1. From the **File** menu select **Import**. The **Import** dialog appears.
2. In the **Import** dialog, expand **Web** and select the WAR file. Click **Next** to launch the WAR Import dialog.
3. To import the sample WAR files, click **Browse**.
4. Browse to the ISDK component folder or filesystem folder and select the WAR file.
5. Provide a name for your Web project. If you have already configured the Oracle WebLogic Server runtime, it will be shown as the Target runtime. If you have not created it yet, click **New** to create a Web project. For more information, refer to "*Creating a Server Runtime Environment in Eclipse*".
6. Click **Next**.
7. Accept the default at the **WAR Import: Web libraries** dialog and click **Finish** to populate the Web project.
8. Click **Yes** if Eclipse asks you to open the "J2EE perspective" for this project.
9. The project should now be created. There should be no compilation errors with the Java code in your project if "Build" is invoked.

## 3.2.4 Configuring ISDK Components

This section provides information on configuring ISDK components.

### 3.2.4.1 Configuring the ISDK Skeleton

The AutoVue Integration SDK Skeleton provides a basic framework for you to build your own integration.

After you complete the steps outlined in [Section 3.2.3, "Creating Projects on IDE,"](#) you must configure the ISDK Skeleton as described in the following steps.

**3.2.4.1.1 Step 1: Copy the AutoVue Jar Files** Copy the following files from the directory <AutoVue Installation directory>\bin to your project's WebContent\applet folder (for Eclipse) or public\_html\jvue folder (for JDeveloper):

- ▣ jvue.jar
- ▣ jogl.jar
- ▣ gluegen-rt.jar
- ▣ jsonrpc4j.jar
- ▣ jsonrpc4j-client.jar
- ▣ slf4j-simple-2.0.17.jar
- ▣ jogl.jnlp
- ▣ 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

Copy the file `vueservlet.jar` and `jsonrpc4j-client.jar` from the directory `<AutoVue Installation directory>\bin` to your project's `WebContent\WEB-INF\lib` folder (for Eclipse) or `public_html\WEB-INF\lib` folder (for JDeveloper).

**3.2.4.1.2 Step 2: Configure the AutoVue Server** 1. From the `WEB-INF` folder of your project, open the `web.xml` file in a text editor.

2. Locate the following block.

```
<servlet id="csi_servlet_2">
<servlet-name>VueServlet</servlet-name>
<servlet-class>com.cimmetry.servlet.VueServlet</servlet-class>
<init-param>
<param-name>JVueServer</param-name>
<param-value>localhost:5099</param-value>
</init-param>
<init-param>
```

3. Update the default location of `JVueServer` "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`).
4. Save your changes.

**3.2.4.1.3 Step 3: Configure log4j.properties for Debugging** The location of `log4j.properties` file is defined in `web.xml`. By default, it is located at `<ISDK Installation Directory>\ISDKSkeleton\WebApplication\isdk_skeleton\WEB-INF\lib` folder.

```
<init-param>
<param-name>log4jInitFile</param-name>
<param-value>/WEB-INF/lib/log4j.properties</param-value>
</init-param>
```

To configure `log4j.properties` for debugging, do the following:

1. Open the `log4j.properties` file with a text editor.
2. Set the location and the filename of your `log4j` logging file, for example, `C:/tmp/filesys.log`.

```
# setting the logging file
log4j.appender.R.File=<Your logs directory>/<logfile>.log
```
3. You can change the level and location of output by modifying this file, for example, `log4j.logger.com.cimmetry.vuelink=DEBUG`.

The following table shows the different levels of logging available.

**Table 3–1 Will Output Messages of Level**

Logger Level	DEBUG	INFO	WARN	ERROR	FATAL
DEBUG	YES	YES	YES	YES	YES

**Table 3–1 (Cont.) Will Output Messages of Level**

Logger Level	DEBUG	INFO	WARN	ERROR	FATAL
INFO	NO	YES	YES	YES	YES
WARN	NO	NO	YES	YES	YES
ERROR	NO	NO	NO	YES	YES
FATAL	NO	NO	NO	NO	YES
ALL	YES	YES	YES	YES	YES
OFF	NO	NO	NO	NO	NO

- ⌘ If you set Logger Level to FATAL, then only output messages of level FATAL are logged in log4j file.
- ⌘ If you set Logger Level to ERROR, then only output messages of level ERROR or FATAL are logged in log4j file.
- ⌘ If you set Logger Level to DEBUG, then output messages of any level are logged in log4j file.

**4.** Save your changes.

For more information on log4j capabilities, refer to log4j documentation.

### 3.3 Configuring Sample Components

If you installed the Sample Integration for Filesys project, then you must import the project into JDeveloper or Eclipse IDE, and then configure, deploy and run the sample integration. To do so, you must follow the steps outlined in the [Deployment of ISDK-Based Integrations](#). You need to make sure that you have all the prerequisite software installed before you start deployment. For the complete list of requirements, specific to your platform, refer to the [System Requirements](#) chapter.

Once you have successfully deployed the Eclipse project, the next step is for you to get familiar with the sample integration. To learn more about the features and functionality provided by the sample integration, refer to the [Configuring Sample Projects](#) chapter in this book.

Once you are familiar with the sample integration, the next step is for you to build your own integration. To do so, refer to the [Implementation](#) chapter in *Oracle AutoVue Integration SDK Technical Guide* and *Javadocs*. The Oracle AutoVue Integration SDK Technical Guide explains the technical details and provides step by step guidance for developing your own integration.



---

## Configuring Sample Projects

The sample projects included with the ISDK provide a good introduction to the many uses of the ISDK. You can take the information provided and apply it to your own integration.

The following sections detail the configuration steps for these sample projects.

### 4.1 Sample Integration for Filesys DMS

This section describes the steps required to run the Sample Integration for Filesys DMS. This project is located in the <ISDK Installation Directory>\AutoVueIntegrationSDK\FileSys folder.

#### 4.1.1 Step 1: Copy the AutoVue Jar Files

Copy the following files from the directory <AutoVue Installation directory>\bin to your project's WebContent\jvue folder (for Eclipse) or public\_html\jvue folder (for JDeveloper):

- ⌘ jvue.jar
- ⌘ jogl.jar
- ⌘ gluegen-rt.jar
- ⌘ jsonrpc4j.jar
- ⌘ jsonrpc4j-client.jar
- ⌘ slf4j-simple-2.0.17.jar
- ⌘ jogl.jnlp
- ⌘ 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

Copy the file vueservlet.jar and jsonrpc4j-client.jar from the directory <AutoVue Installation directory>\bin to your project's WebContent\WEB-INF\lib folder (for Eclipse) or public\_html\WEB-INF\lib folder (for JDeveloper).

#### 4.1.2 Step 2: Configure the AutoVue Server

1. From the WEB-INF folder of your project, open the web.xml file in a text editor.

2. Locate the following block:

```
<servlet id="csi_servlet_2">
<servlet-name>VueServlet</servlet-name>
<servlet-class>com.cimmetry.servlet.VueServlet</servlet-class>
<init-param>
<param-name>JVueServer</param-name>
<param-value>localhost:5099</param-value>
</init-param>
<init-param>
```

3. Update the default location of JVueServer "localhost:5099". You must replace localhost with an IP address/FQDN 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).
4. Save your changes.

### 4.1.3 Step 3: Configure log4j.properties for Debugging

The location of log4j.properties file is defined in web.xml. By default, it is located at WEB-INF/lib folder.

```
<init-param>
<param-name>log4jInitFile</param-name>
<param-value>/WEB-INF/lib/log4j.properties</param-value>
</init-param>
```

To configure log4j.properties for debugging, do the following:

1. Open the log4j.properties file with a text editor.
2. Set the location and the filename of your log4j logging file. For example, C:/tmp/filesys.log.

```
# setting the logging file
log4j.appender.R.File=<Your logs directory>/<logfile>.log
```
3. You can change the level and location of output by modifying this file. For example, log4j.logger.com.cimmetry.vuelink=DEBUG

The following table shows the different levels of logging available.

**Table 4–1 Will Output Messages of Level**

Logger Level	DEBUG	INFO	WARN	ERROR	FATAL
DEBUG	YES	YES	YES	YES	YES
INFO	NO	YES	YES	YES	YES
WARN	NO	NO	YES	YES	YES
ERROR	NO	NO	NO	YES	YES
FATAL	NO	NO	NO	NO	YES
ALL	YES	YES	YES	YES	YES
OFF	NO	NO	NO	NO	NO

- If you set Logger Level to FATAL, then only output messages of level FATAL are logged in log4j file.
- If you set Logger Level to ERROR, then only output messages of level ERROR or FATAL are logged in log4j file.



- If you set Logger Level to DEBUG, then output messages of any level are logged in log4j file.]

4. Save your changes.

For more information on log4j capabilities, refer to log4j documentation.

#### 4.1.4 Step 4: Configure RootDir for the Filesys Repository

1. From the public\_html\WEB-INF folder of your project, open the web.xml file in a text editor.
2. Replace the RootDir param-value. For example, if you have unzipped the Filesys Repository to folder c:\tmp on Windows, the param-value for RootDir will be c:\tmp\filesysRepository.

```
<!-- context parameters are available to all servlets -->
<context-param>
<param-name>RootDir</param-name>
<param-value>Put path to repository here:</param-value>
</context-param>
```

3. Save your changes.

#### 4.1.5 Step 5: Configure for an Embedded or Pop-Up Window (Optional)

AutoVue can be launched in a pop-up window or embedded inside the caller's browser window.

By default, the Filesys demo uses embedded mode and the RTC demo uses pop-up mode.

For the OEVF demo, you can select the mode by providing *embedded=0* or *embedded=1* request parameter in the launching URL. Refer to [jvue/OEVFDemo.html](#).

To change the mode in Filesys demo:

1. Open `jvue/frmApplet.jsp`.
2. Change the line  
`boolean embedded = true;`  
to  
`boolean embedded = false;`

To change the mode in RTC demo:

1. Open `jvue/RTCDemo_init.jsp` and `jvue/RTCDemo_join.jsp`.
2. Change the line  
`boolean embedded = false;`  
to  
`boolean embedded = true;`

#### 4.1.6 Step 6: Configure the Markup Policy (Optional)

The location of MarkupPolicy.xml file is defined in web.xml that controls markup operation. By default, it is located at WEB-INF/lib folder.

```
<init-param>
<param-name>CSI_MarkupPolicyDefLocation</param-name>
<param-value>/WEB-INF/lib/MarkupPolicy.xml</param-value>
</init-param>
```

If you need to update the Markup Policy file, refer to the *Oracle AutoVue User's Manual*. On Windows, the link is <http://localhost/jVue/help/en/AutoVueOnLineHelp.html>. If the link does

not work, check whether there is a virtual directory, jVue, with IIS. It is created during AutoVue server installation.

### 4.1.7 Step 7: Configuring User Control

By default, the Sample Integration for Filesys bundles a file called credential.txt that contains valid user information for authentication. The location of credential.txt file is defined in web.xml.

```
<init-param>
<param-name>CredentialInfoLocation</param-name>
<param-value>/WEB-INF/lib/credential.txt</param-value>
</init-param>
```

To add new users or modify existing user name or password, update credential.txt. Each line of the file contains an entry for a user and its password. The field separator is colon (:).

### 4.1.8 Step 8: Configure the Picklist

This list is for controlling the content of a picklist for Stamp (formerly called Intellistamp) DMS properties. You can remove/modify existing values or add new values for the <Status> and <RelatedInfo> elements in WEB-INF/lib/picklist.xml, but you are not supposed to delete these two elements or add new elements directly under <Data> element.

### 4.1.9 Step 9: Configure the Thumbnail Display

If you want to show thumbnails based on BMP renditions when browsing the Filesys Repository, you can do the following configuration.

1. For Windows operating systems, create a virtual directory on Internet Information Services (IIS) for the Filesys repository. For example, if you have unzipped the Filesys repository to folder c:\tmp on Windows, you can create a virtual directory with alias filesysRepository and the location path c:\tmp\filesysRepository. Suppose IIS is available at the default port 80.
2. For Linux system, if Apache Server is available, do the following configuration.
  - ⌘ Open Apache's httpd.conf file.
  - ⌘ Locate the line: DocumentRoot "/var/www/html".
  - ⌘ Copy this line and comment out the original one.
  - ⌘ Change the copied line to, for example, DocumentRoot "/home/ucm/tmp"

Suppose your Filesys repository is unzipped to /home/ucm/tmp folder and your /home/ucm/tmp/filesysRepository folder allow executing file as program. If your DocumentRoot has already been used, you need to put your Filesys repository under the existing DocumentRoot folder in order to preview thumbnails.

  - ⌘ Save the file and restart Apache Server.
3. Replace the param-value for RootURL in web.xml. This URL is mainly used for thumbnail displaying. However, you must enter a URL (for example, http://localhost) even if thumbnail displaying is not intended. With the configuration sample in Step 1, the param-value for RootURL will be http://localhost/filesysRepository. Note the case sensitivity of IIS.

```
<!-- This URL is only needed to construct thumbnail URLs -->
<context-param>
<param-name>RootURL</param-name>
<param-value>http://localhost/filesysRepository</param-value>
```

```
</context-param>
```

#### 4.1.10 Step 10: Configure for Redirection

To test the redirection functionality in Filesys, you need to install IDE and deploy the Filesys sample project on two machines (a main server and a remote server) and complete the generic configuration and other configurations based on your needs. You must then perform the following configurations for redirection:

1. On the main server machine, change the folder permission for the filesys repository to Full Control for all users.
2. On the remote server machine, create a network mapping drive to the Filesys repository directory on the main server machine. In Filesys demo, both remote server and the main server use the same Filesys repository data.
3. On the main server, modify web.xml to comment out the blocks *RemoteVueLink*, *RemotejVueServer* and *RemoteVueServlet*. Specify the param-values for these three parameters.

**Table 4–2 Param-names and param-values for web.xml**

Param-name	Description and param-value
RemoteVueLink	URL to the remote VueLink. The param-value is http://host:port/context/servlet/FilesysVuelink where host is the remote host name or IP address, port is the remote IDE's server runtime port number, context is the Filesys project name on the remote IDE.
RemotejVueServer	Hostname or IP address of the remote AutoVue server. The remote server can use another AutoVue server instead of the one running on the main server.
RemoteVueServlet	URL to the remote VueServlet. The param-value is http://host:port/context/servlet/VueServlet

For Example:

```
<context-param>
<param-name>RemoteVuelink</param-name>
<param-value> http://sremote:7001/ISDK_Remote/servlet/FilesysVuelink</param-value>
</context-param>
<context-param>
<param-name>RemotejVueServer</param-name>
<param-value>sremote</param-value>
</context-param>
<context-param>
<param-name>RemoteVueServlet</param-name>
<param-value>http://sremote:7001/ISDK_Remote/servlet/VueServlet</param-value>
</context-param>
```

#### 4.1.11 Step 11: Configure the Real-Time Collaboration (RTC) Demo

The following section describe how to configure the RTC demo:

#### 4.1.11.1 Verify the RTC Demo

Make sure the WEB-INF/lib/credential.txt has an entry for user "rtc" and "rtc1". Although every valid user can initiate and join a meeting, by default the meeting is initiated as user "rtc" and joined by user "rtc1" and the AutoVue applet is named after the username.

Prior to running the demo, you must do the following:

- Uncomment the two users (rtc and rtc1) from the WEB-INF/lib/credential.txt file and change the default passwords.
- Update the password parameter in the jvue\RTCDemo\_init.jsp file for user rtc:

```
request.getSession().setAttribute("password", "rtc");
```

Update the password parameter in the jvue\RTCDemo\_join.jsp file for user rtc1:

```
request.getSession().setAttribute("password", "rtc1");
```

Updating these parameters avoids an Authentication dialog when initializing or joining a RTC meeting when using the ISDK RTC demo.

#### 4.1.11.2 Create or Update the meetingfiles.txt

Verify that the meetingfiles.txt file under your <Filesys repository>/Meeting folder exists. If this file does not exist, you need to create it manually.

If you want to change the files shown in the Meeting File drop down list when initiating a RTC meeting from RTCDemo\_init.jsp page similar to the following figure, then you need to update the meetingfiles.txt file.

Each entry in the meetingfiles.txt file represents one meeting file; it starts with "/" and reflects one viewable document file in the Filesys repository.

To select another file to collaborate on during a meeting, from the AutoVue menu bar, the meeting controller can click **File**, **Open URL**, and then **DMS Browse**. The Meeting folder shows files already defined in meetingfiles.txt. The new collaborated file is appended to meetingfiles.txt.

After the host closes a RTC meeting by clicking **Collaboration** and then **Close Collaboration Session**, the chat transcript is saved to the Meeting folder.

### 4.1.12 Step 12: Configure the Oracle Enterprise Visualization Framework (OEVF)

The following sections describe how to configure the OEVF:

#### 4.1.12.1 Define OEVFInfoLocation in web.xml

By default, ISDK filesys bundles a file called oevf.xml which defines the mapping of document IDs with assetIDs and workflowIDs. The default location of oevf.xml is under the folder WEB-INF/lib. If you move the file to another location, then you need to specify the full path for the parameter *OEVFInfoLocation* in web.xml.:

```
<!--
# the location of xml file which contains all the info about assetIDs, workflowIDs
# and full path of the latest revision in FileSys DMS
-->
<init-param>
<param-name>OEVFInfoLocation</param-name>
<param-value>/WEB-INF/lib/oevf.xml</param-value>
</init-param>
```

#### 4.1.12.2 Update oevf.xml

This step is required if you want to establish new or update existing mappings of document IDs with assetIDs and workflowIDs.

The root element of the oevf.xml file is `<data>`. The direct elements under `<data>` are `<file>` elements that contain the definition for files. Each `<file>` element represents one file. If you want to add mapping relationships for a new file, then you need to add a new `<file>` entry.

A `<file>` element can include multiple `<revision>` elements that represent the multiple revisions of the file. If you want to add a new revision section to an existing file, then you must add one new `<revision>` entry.

Each `<version>` element includes a `<docID>`, `<assetIDs>`, `<workflowIDs>` and `<version>` elements. The value for `<version>` element is the revision number. The value for `<docID>` element is the relative path to a file in the Filesys data repository. It starts with `"/`. For example, `/2D/MicroStation.dgn/MicroStation.dgn(2)/MicroStation.dgn`.

The `<assetID>` elements can contain multiple `<assetID>` elements and the `<workflowID>` elements can contain multiple `<workflowID>` elements. You can add or delete an assetID that is associated with one revision of a file by adding or deleting element a `<assetID>` element. You can add or delete a workflowID, that is associated with one revision of a file by adding or deleting a `<workflowID>` element.

#### 4.1.12.3 Update OEVFDemo.html

This step is needed to add new or modify existing test cases for OEVF.

The launching OEVF URL defined inside `<a>` tag calls `"/jvue/frmApplet.jsp"` page combined with some of the following parameters.

**Table 4–3 OEVF URL parameters**

URL Request Parameter	Value and Description
aID	A Value defined for <code>&lt;assetID&gt;</code> element in oevf.xml.
docID	A value defined for <code>&lt;docID&gt;</code> element in oevf.xml.
wID	A value defined for <code>&lt;workflowID&gt;</code> element in oevf.xml.
embedded	V0 or new such parameter: AutoVue applet appears in a new window. 1: AutoVue applet is embedded in the caller's browser window.
goBack	Work together with embedded=0. 0 or no such parameter: The caller's browser displays an empty page with the launching OEVF URL. 1: The caller's browser displays the OEVFDemo.html page.
guiFile	Name of the AutoVue GUI to be used.

You can pass in only aID, only wID, aID with wID, aID with dID, wID with dID, aID with wID and dID in addition with embedded or goBack or guiFile param. Refer to *OEVFDemo.html* for the meaning of different combinations.

#### 4.1.12.4 Copy the OEVF GUI files to AutoVue

Copy assetView.gui and assetEdit.gui files from inside the ISDK installation AutoVueIntegrationSDK/FileSys/OEVF folder to the folder `<AutoVue Installation Directory>/bin/Profiles` folder. If the Profiles folder does not exist, create one before copying.

### 4.1.13 Step 13: Configure New Sample Data

You can add new data to the existing sample Filesys repository. It is recommended not to rename the folder name or file name, or delete existing data, because the sample data is preconfigured to demonstrate certain functionalities (for example, for RTC Demo and OEVF demo). To add new data to the existing repository, refer to [Add new data to the document repository](#).

#### 4.1.13.1 Add new data to the document repository

With Filesys DMS application you can add new data to the document repository by manually creating the data structure or by using a utility class `com.cimmetry.vuelink.filesys.dms.util.FilesysDataStructureCreator` coming with filesys.

**4.1.13.1.1 Create data structure manually** To add a new document into the repository, such as "my.dwg", follow these steps:

1. Browse to the <filesys data repository unzipped folder>/filesysRepository folder.
2. You can create a new folder in parallel to "2D", "3D" and "Meeting" folder or create a new folder inside "2D" or "3D" folder. Suppose you want to add the file inside "2D" folder. You can name the folder using the file's name that you want to view, that is, "my.dwg". This is the convention for sample data.
3. Under this folder, create a new folder for the first revision of the file. Name it "my.dwg(1)" and put the file "my.dwg" inside the folder.
4. If you have another revision for "my.dwg" file, then you can create "my.dwg(2)" folder under "my.dwg" folder and put the second revision of the file inside.
5. If the file "my.dwg" has XRefs, then create an "xrefs" folder under the base file's folder "my.dwg(1)" and put all the XRefs files there.

**4.1.13.1.2 Add data from IDE** To add the new data to your document repository, you must execute the `main()` method of `com.cimmetry.vuelink.filesys.dms.util.FilesysDataStructureCreator` class.

#### **Example 4-1** *main() method*

```
public static void main(String[] args) {
    BasicConfigurator.configure();
    //String[] params = {"-url", "C:/temp/filesysRepository/ECAD", "-b", "C:/program
files/jVue/html/samples/ECAD/PAD//PADS_ILEARN.pcb"};
    //params = {"-url", "C:/temp/filesysRepository/2D", "-b", "C:/program
files/jVue/html/samples/2D/MicroStation.dgn" -v 3};
    FilesysDataStructureInfos data = new FilesysDataStructureInfos();
    try{
        data.constructStructure(args);
        FilesysDataStructureCreator struct = new FilesysDataStructureCreator(data);
        struct.createStructure();
    }catch(FileNotFoundException fex){
        m_logger.error(fex);
        System.exit(0);
    }catch(Exception ex){
        m_logger.error(ex);
        System.exit(0);
    }
```

You need to pass in arguments for the `main()` method. These arguments indicate types, versions and locations of the files to add in the repository. You can add several types of documents to the repository such as: base documents, XRefs, markups and conversions files. We use options **<-option>** to indicate the document type. Here is the complete list the options:

-url: location of filesys repository  
 -b: base file  
 -v: version number  
 -x: xrefs files  
 -m: master markups files  
 -n: normal markups files  
 -c: consolidated markups files  
 -tiff: TIFF conversion file  
 -pdf: PDF conversion file  
 -meta: metaFile

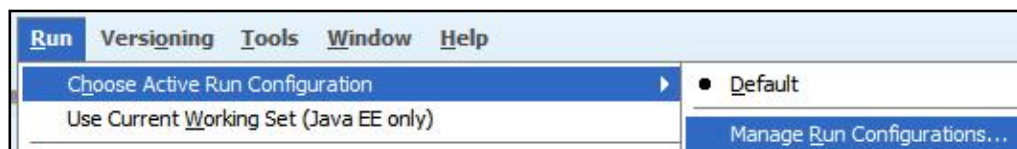
For the first sample argument in the figure above, the url is c:\temp\filesysRepository\ECAD (You don't have to specify the exact destination location of a file, you have just to specify the repository location) and the base file is PADS\_ILEARN.pcb and is located in C:\program files\jVue\html\samples\ECAD\PAD folder.

For the second sample argument, the url is c:\temp\filesysRepository\2D and the base file is MicroStation.dgn and is located in C:\program files\jVue\html\samples\2D folder. The version number is 3.

**4.1.13.1.3 Adding Data from JDeveloper** On JDeveloper IDE, to run the FilesysDataStructureCreator class, you need to complete the following steps:

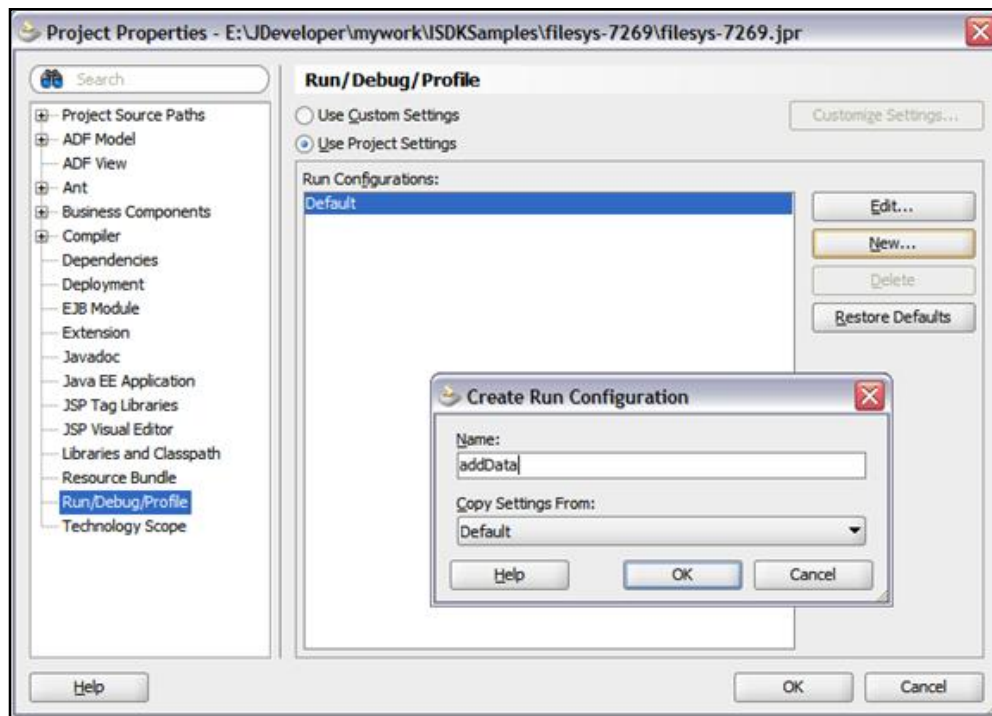
1. Select the **filesys** project
2. From the **Run** menu, select **Choose Active Run Configuration**, and then select **Manage Run Configurations...**

**Figure 4–1 JDeveloper Manage Run Configurations**



3. Click **New...** at the right side of the Project Properties windows, name it **addData** and then click **OK**. addData appears under the Run Configurations.

Figure 4–2 JDeveloper addData Run Configurations

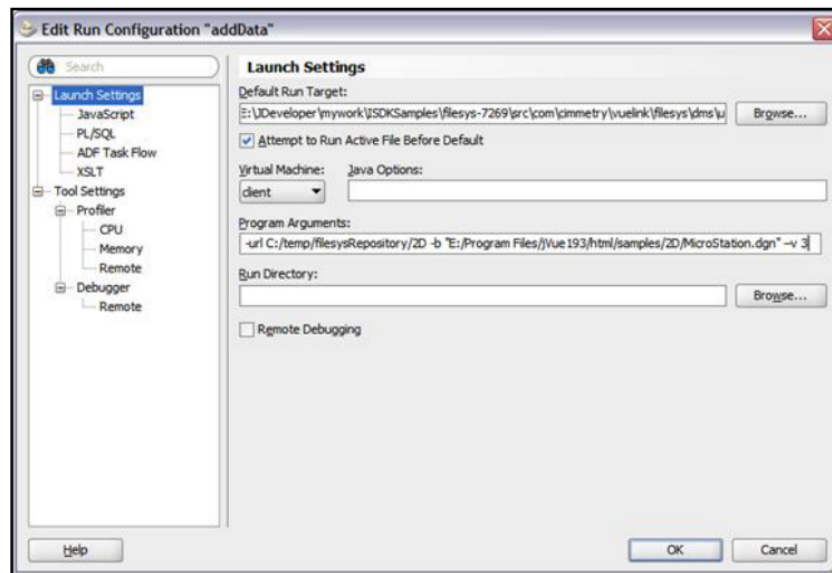


4. Select **addData** and click **Edit**.
5. Browse to set the Default Run Target to be `com.cimmetry.vuelink.filesys.dms.util.FilesysDataStructureCreator` and input Program Arguments. Click **OK** to exit. Two sample program arguments are

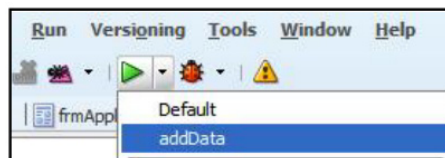
```
-url C:/temp/filesysRepository/EDA -b "C:/Program
Files/jVue/html/samples/EDA//PADS/PADS_ILEARN.pcb"
and
```

```
-url C:/temp/filesysRepository/2D -b "C:/Program
Files/jVue/html/samples/2D/MicroStation.dgn" -v 3
```



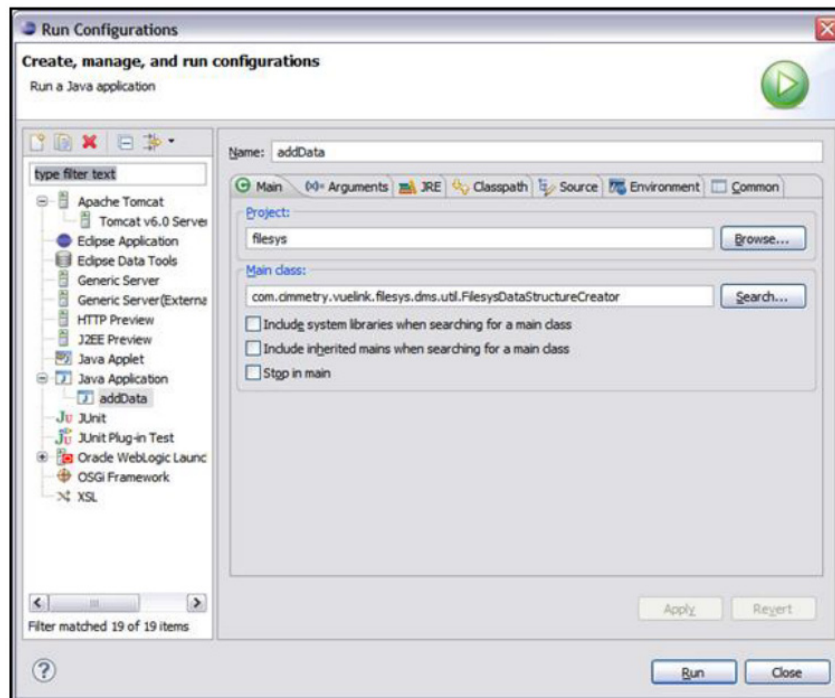
**Figure 4–3** JDeveloper create addData run configuration 2

6. Run addData to create new file structure in the filesys repository.

**Figure 4–4** JDeveloper run addData

**4.1.13.1.4 Adding Data from Eclipse** On Eclipse IDE, to run the FilesysDataStructureCreator class, you need to complete the following steps:

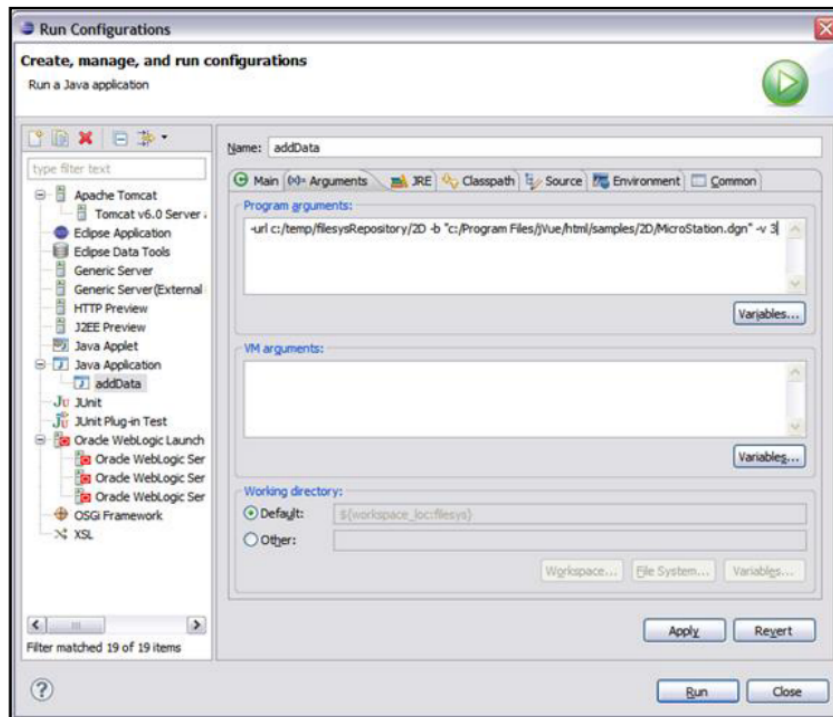
1. Select the **filesys** project.
2. From the **RUN** menu click on the **RUN...** artifact.
3. Select the **Java Application** item.
4. Right click and select **New** from the context menu.
5. Enter **addData** in the Name field.
6. Search the class to execute (the class must have public static main method)  
com.cimmetry.vuelink.filesys.dms.util.FilesysDataStructureCreator

**Figure 4–5 Eclipse create addData run configuration 1**

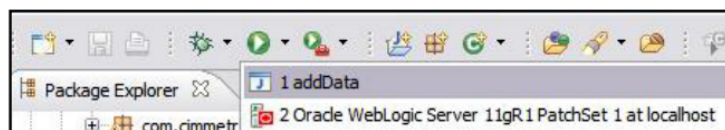
7. Click the **Arguments** tab. You can add **Program Arguments** to create a file. Two sample arguments are:

```
-url C:/temp/filesysRepository/EDA -b "C:/Program
Files/jVue/html/samples/EDA//PADS/PADS_ILEARN.pcb"
and
```

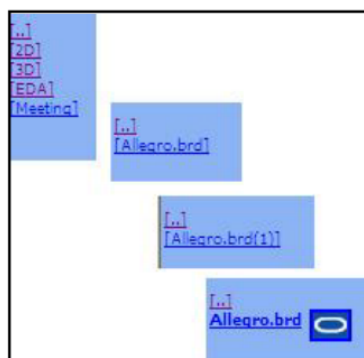
```
-url C:/temp/filesysRepository/2D -b "C:/Program
Files/jVue/html/samples/2D/MicroStation.dgn" -v 3
```

**Figure 4–6 Eclipse create addData run configuration 2**

8. You can click **RUN** directly from the above to create new file structure in the filesys repository or you can click **Apply** and **Close**, then click on addData artifact from the **RUN** icon.

**Figure 4–7 Eclipse run addData**

If you provide the first sample argument when running the addData, from your filesys demo application, you can navigate to check the following new repository structure under the EDA folder.

**Figure 4–8 Repository structure for PADS\_ILEARN.pcb**

If you provide the second sample argument when running the addData, from your filesystems demo application, you can navigate to check the following new repository structure.

**Figure 4–9 Repository Structure for the third version of MicroStation.dgn**



#### 4.1.14 Step 14: Run the Filesys Project

The following steps describe how to run the Sample Integration for Filesys project:

1. Run the AutoVue Server.
2. Deploy project and start WebLogic server on Eclipse:
  - ⌘ Go to the Servers view by clicking Servers.
  - ⌘ Right-click on the Oracle WebLogic Server and then click **Add and Remove**. In the Add and Remove dialog, select your project from the left panel, click **Add** to add the project to the right panel, and then click **Finish** to exit.
  - ⌘ Click to start the server.
3. Start WebLogic server and deploy project on JDeveloper:
  - a. From menu bar select **Run** and then select **Start Server Instance** (IntegratedWeblogicServer) to start the WebLogic Server.
  - b. Right-click the project, click **Deploy** and click your project's name. The Deploy dialog appears.
  - c. From the Deploy dialog, select **Deploy on Application Server**, click **Next**, then select **IntegratedWeblogicServer**.
  - d. Accept the default setting and click **Next**.
  - e. At the last page, click **Finish**.
  - f. Note down the host IP and port number from the server's Deployment log. For example, the following may appear in the log: *<Channel "Default" is now listening on 10.10.1.1:7101 for protocols...>*. In this case, note down *10.10.1.1:7101*.
4. Launch a Web browser and enter the URL address `http://<localhost :port>/<context>` to launch the home page for ISDK Demo. For example:
  - ⌘ For Eclipse, the URL can be `http://<localhost> :7001/filesys`.
  - ⌘ For JDeveloper, the URL can be `http://10.10.1.1:7101/ISDKSamples-filesys-context-root`.
5. If you run into an issue when launching the project, verify that the FilesysVueLink and VueServlet servlets are running properly using the following URLs:
  - ⌘ `http://<host:port>/context/servlet/FilesysVuelink`
  - ⌘ `http://<host:port>/context/servlet/VueServlet`

Replace the *<host:port>* using your own host name, WebLogic server's port.

Replace context with the context for Filesys project on IDE.

If VueLink and VueServlet are running properly, the URLs load and display their respective version and build information, and in the case of the VueServlet, whether the connection state is OK. If you do not get a successful response, perform the following verifications:

- ▣ Verify that the AutoVue server is running.
- ▣ Verify that your project is installed deployed correctly.
- ▣ Verify that web.xml is configured properly.
- ▣ Verify that your application server is running and functioning properly.



---

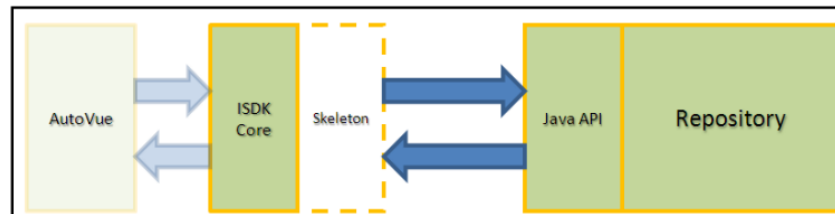
## Implementation

To speed up the integration and provide the integrators with a starting point, the ISDK includes a skeleton package.

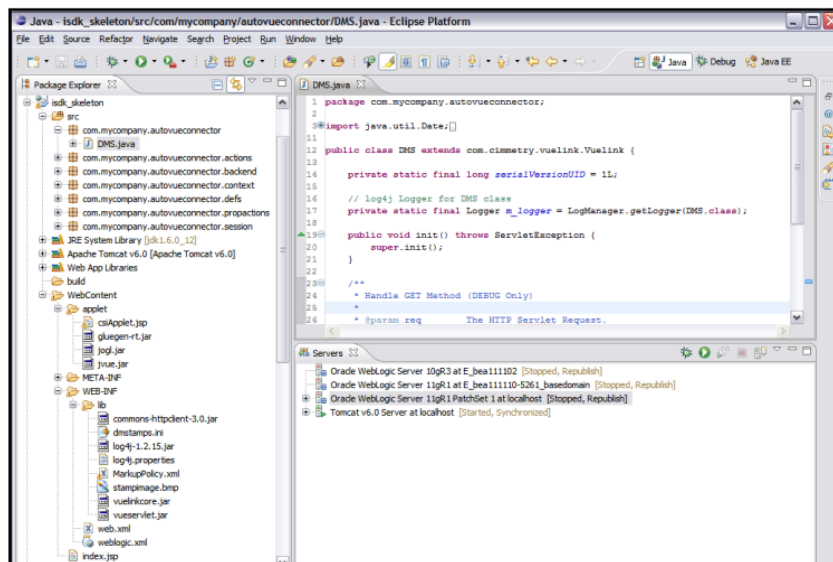
### 5.1 ISDK Skeleton Project

The ISDK Java skeleton package has the structure for building a new VueLink. The skeleton comes with a set of *TODO* comments in places where the integrators need to add their code. The ISDK Java skeleton implementation means adding code to the skeleton codebase so that it can communicate with the repository's Java API as shown in the [Figure 5–1](#):

**Figure 5–1** ISDK Java skeleton implementation



This project is located under the ISDKSkeleton/WebApplication folder and is available as single WAR (isdk\_skeleton.war) as well as separate files. It can be imported into JDeveloper or Eclipse workspace.

**Figure 5–2 Eclipse workspace**

It contains four main subcomponents:

- ⊗ Framework (VueLink Core): vuelinkcore.jar
- ⊗ Integration Skeleton
- ⊗ AutoVue Components:
  - AutoVue Client (jvue.jar, jogl.jar, gluegen-rt.jar, jsonrpc4j.jar)
  - VueServlet tunneling servlet (VueServlet.jar)
- ⊗ Third-Party Libraries:
  - log4j-api.jar
  - log4j-core.jar
  - log4j-slf4j2-impl.jar
  - slf4j-simple-2.0.17.jar
  - jsonrpc4j-client.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

Refer to [Sample Projects](#) for description about the two subcomponents: Framework and AutoVue Components.

It includes skeleton classes containing TODO tasks to realize the following functionalities:

- ⊗ Document viewing
- ⊗ Retrieve document attributes
- ⊗ Create, save and review markups



- ▣ Compare document versions
- ▣ Convert documents to other formats
- ▣ Returning External References (XREFS)
- ▣ Browse DMS repository
- ▣ Search DMS repository
- ▣ Support for Stamps markup entity
- ▣ Support for Set Property action
- ▣ Support for AutoVue authorization mechanism
- ▣ Support for integration between Online meeting managements and AutoVue Real-Time Collaboration (RTC)
- ▣ Support for markup save alert before applet close
- ▣ Enhanced framework to support Oracle Enterprise Visual Framework (OEVF)

The Integration Skeleton contains the following packages:

- ▣ com.mycompany.autovueconnector
- ▣ com.mycompany.autovueconnector.actions
- ▣ com.mycompany.autovueconnector.backend
- ▣ com.mycompany.autovueconnector.context
- ▣ com.mycompany.autovueconnector.defs
- ▣ com.mycompany.autovueconnector.propactions
- ▣ com.mycompany.autovueconnector.session

It includes a sample Front-UI file - applet/csiApplet.jsp - which contains HTML code for launching AutoVue applet (<applet> tag). When you develop your DMS extension on the DMS Server, you can customize this sample file.

Refer to the *Oracle AutoVue Integration SDK Technical Guide* about steps to design your integration based the ISDK Skeleton.

## 5.2 Sample Projects

The installation of the ISDK includes the sample project:

- ▣ Sample Integration for Filesys DMS

These projects provide a good introduction to the many uses of the ISDK. The following sections provide an introduction to each project. For full configuration information, refer to [Configuring Sample Projects](#).

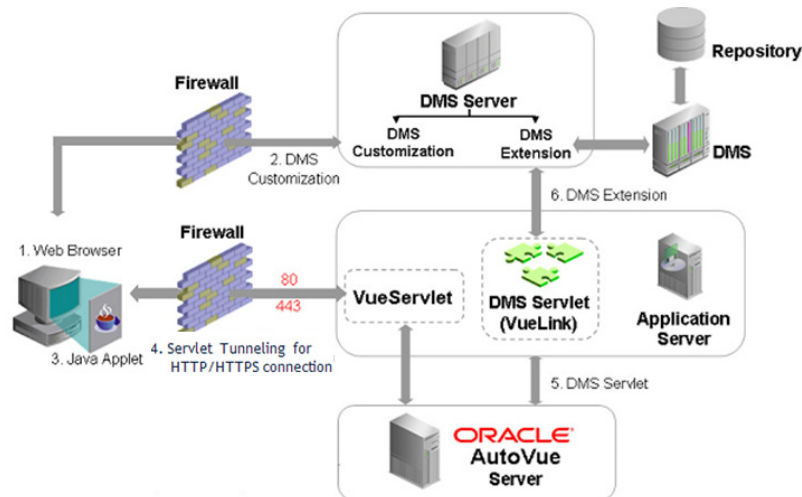
### 5.2.1 Sample Integration for Filesys Project

The AutoVue Integration SDK bundles a sample integration into a JDeveloper-based or Eclipse-based project. This project is located under the FileSys\WebApplication folder and is available as single web archive (WAR) file `filesystem.war` and separate files. This provides you with an option of either importing the project from single WAR file into your workspace or manually creating a project and adding individual pieces to it.

The following high-level architectural diagram shows how various components included in the Sample Integration for Filesys are related to each other as well as to others. For detailed

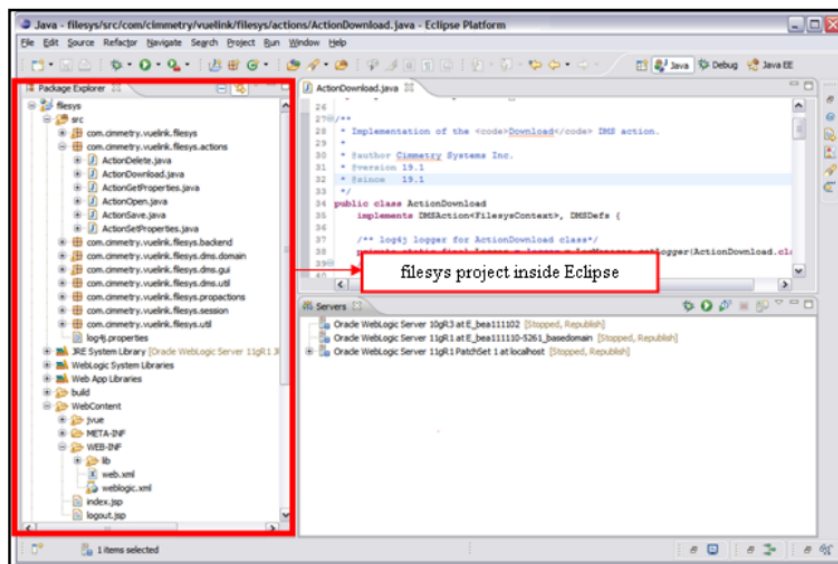
description about this architectural diagram, refer to *Oracle AutoVue Integration SDK Technical Guide*.

**Figure 5–3 High-level architectural diagram of Sample Integration for Filesys components**



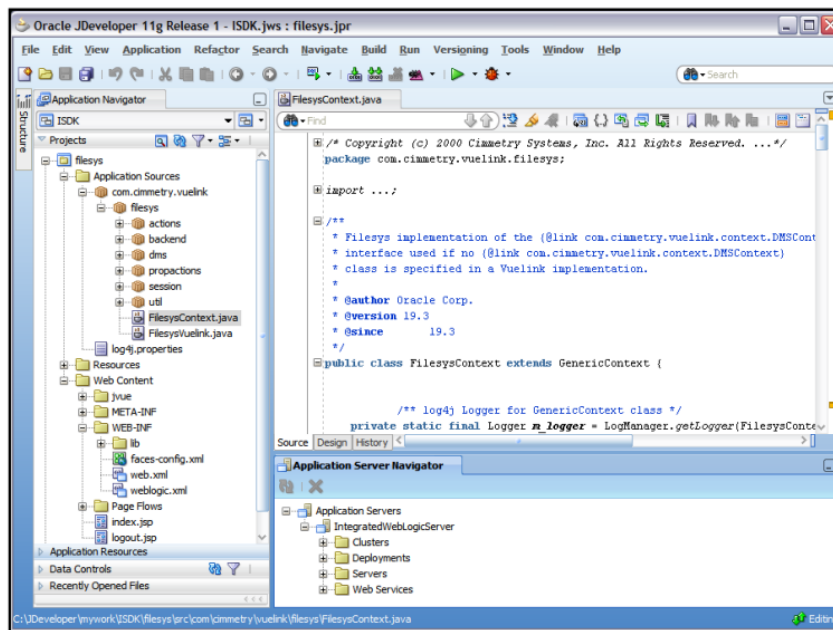
The following is a sample integration project on Eclipse.

**Figure 5–4 Sample integration project on Eclipse**



The following is a sample integration project on JDeveloper:

Figure 5–5 Sample integration project on JDeveloper



The project contains four main subcomponents:

- ❑ Framework (VueLink Core)
- ❑ Sample Integration
- ❑ AutoVue Components
- ❑ Third-Party Libraries

The following sections describe each of these subcomponents.

### 5.2.1.1 Framework (VueLink Core)

The framework is implemented based on Java Servlet API provided by SUN as part of J2EE. The main class is called *com.cimmetry.Vuelink* which extends *javax.servlet.http.HttpServlet*. Servlets do not run on their own, they require a servlet engine such as Oracle WebLogic Server.

The framework links AutoVue system with a third-party DMS. Integration framework receives requests from AutoVue, obtains information from the DMS, and then builds a response back to AutoVue.

The framework provides plumbing for parsing XML requests received from AutoVue Server as well as constructing XML responses sent back to AutoVue Server. The framework uses XML parser libraries included in your servlet container.

The framework defines a set of interfaces and classes that facilitate the integration task. The framework is packaged into *vuelinkcore.jar* and contains many packages including following:

- ❑ *com.cimmetry.vuelink*
- ❑ *com.cimmetry.vuelink.authentication*
- ❑ *com.cimmetry.vuelink.backend*
- ❑ *com.cimmetry.vuelink.context*
- ❑ *com.cimmetry.vuelink.defs*
- ❑ *com.cimmetry.vuelink.io*

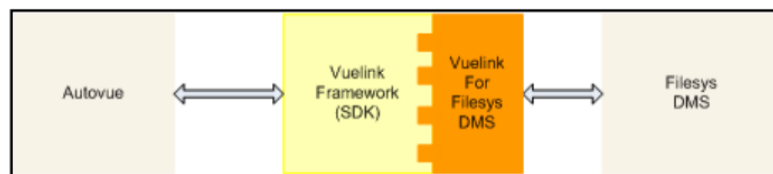
- ▣ com.cimmetry.vuelink.property
- ▣ com.cimmetry.vuelink.prosaction
- ▣ com.cimmetry.vuelink.query
- ▣ com.cimmetry.vuelink.session
- ▣ com.cimmetry.vuelink.util
- ▣ com.cimmetry.vuelink.xml

The framework uses log4j for logging messages into application server log file or console.

### 5.2.1.2 Sample Integration

The ISDK includes a sample integration of a simplified file system management (Filesys). This sample includes VueLink for filesys DMS and aims to act as a starting point for developing your own integration as well as familiarizing yourself with the integration framework.

**Figure 5–6 Sample integration framework**



The Filesys DMS comes with a database repository that is preloaded with some sample files in 2D/3D formats. To simplify things, the structure of this repository is based on local file system. Markups and renditions are stored back into this content repository.

The sample integration demonstrates how you can add basic and advanced functionalities to your own integration including:

- ▣ Document viewing of native formats
- ▣ Retrieve document attributes
- ▣ Create, save and review markups
- ▣ Browse DMS repository
- ▣ Search DMS repository
- ▣ Compare document versions
- ▣ Convert documents to other formats
- ▣ Support for Stamps markup entity
- ▣ Support for Set Property action (with Pick List support)
- ▣ Enhanced framework to support Oracle Enterprise Visual Framework (OEVF)
- ▣ Support for markup save alert before applet close
- ▣ Support for browser Pop-up blocker notification
- ▣ Support for AutoVue authorization mechanism (encrypted Authorization block and password)
- ▣ Improved performance with support for distributed file servers
- ▣ Support for integration between Online meeting managements and AutoVue Real-Time Collaboration (RTC)

- ▣ Support for saving/deleting on-line master markups based on default markup policy
- ▣ Support for read-only markups
- ▣ Bundled demos for OEVA and RTC

The sample integration for Filesys contains many packages including the following:

- ▣ `com.cimmetry.vuelink.filesys`
- ▣ `com.cimmetry.vuelink.filesys.actions`
- ▣ `com.cimmetry.vuelink.filesys.backend`
- ▣ `com.cimmetry.vuelink.filesys.dms`
- ▣ `com.cimmetry.vuelink.filesys.dms.gui`
- ▣ `com.cimmetry.vuelink.filesys.propactions`
- ▣ `com.cimmetry.vuelink.filesys.session`
- ▣ `com.cimmetry.vuelink.filesys.util`

The sample integration includes a front-end UI which allows users to navigate the Filesys DMS data structure. This UI consists of:

- ▣ A default home page: `index.jsp`. It provides links for Filesys demo, RTC demo and OEVF demo.
- ▣ RTC demo pages: `RTCDemo.jsp`, `RTCDemo_init.jsp`, `RTCDemo_join.jsp`
- ▣ OEVF demo pages: `OEVFDemo.html`, `OEVFDemoDes.html`
- ▣ Filesys demo pages: `jVue.html`, `frmApplet.jsp` and a single servlet called `com.cimmetry.vuelink.filesys.dms.gui.ListDirServlet`. The main filesys demo page contains two frames as shown in the following figure.
- ▣ The frame on the left displays the structure of Filesys DMS data. The content of this frame is displayed by `ListDirServlet` which allows you to navigate the Filesys DMS by expanding folders and selecting documents to view.
- ▣ The frame on the right displays the AutoVue applet using `frmApplet.jsp`. When you click a document in the frame on the left, it displays in the frame on the right.

### 5.2.1.3 AutoVue Components

The ISDK bundles following two components of Oracle AutoVue:

- ▣ AutoVue Java Client
  - Referenced by `frmApplet.jsp` which contains HTML and Javascript code to initiate the Web Start of the client.
- ▣ `VueServlet` tunneling servlet (`vueservlet.jar`)
  - This servlet is used to allow AutoVue Java connects to AutoVue Servlet.

The AutoVue Integration SDK does not bundle AutoVue Server. You need to download and install it separately. Refer to the [Installation](#) chapter for more information.

**5.2.1.3.1 Third-Party Libraries** The Sample Integration for Filesys bundles third-party open-source libraries needed by the framework. For information, refer to the *Acknowledgments* document.

## 5.3 Implementation

The ISDK Java skeleton should be used when a Java API is available in the repository.

The implementation steps are dependent on the ISDK Java skeleton being used. However, the expected functionality of the integration can be understood in three phases that range from the most basic (phase one) to the more advanced (phase three) capabilities. The following sections discuss these integration phases.

### 5.3.1 Phase One

The requirement for phase one is viewing the document. To view the document, the integration should cover the Open and Download actions and a subset of GetProperties (get name, size, last modified date and multi-content values) actions.

### 5.3.2 Phase Two

Phase two of the integration adds the following capabilities:

- ▣ Save, update and delete markups (annotations) inside the repository
- ▣ Compare a document with other versions of the same document
- ▣ Download the external references (XRefs) of a document from the repository (if applicable)
- ▣ Save (and reuse) the renditions of a document into the repository
- ▣ Add the repository attributes to the print output in the header/footer sections

For this to happen, the integrators should add implementation for the Save and Delete and a subset of GetProperties related to listing versions, listing markups, listing XRefs, listing renditions and listing all attributes of a document.

As mentioned in [Optional Components](#), the repository should support XRefs and the development of a CAD connector for the repository may be required.

### 5.3.3 Phase Three

Phase three of the integration adds the following capabilities:

- ▣ Search and browse the repository through the AutoVue client UI
- ▣ Use the AutoVue Intellistamp with the repository attributes

AutoVue Intellistamp is one of the AutoVue advanced markup features. For more information, refer to the *Oracle AutoVue User's Manual*.

For these features, integrators must add an implementation for SetProperties and the remaining subset of GetProperties that are defined to retrieve these information: search/browse UI, search/browse query results, and the collaboration-related data from the repository.

---

## Deployment of ISDK-Based Integrations

Once the development of an ISDK-based component is complete, it should be deployed on a Java Web application server.

The deployment may involve some configuration depending on its complexity. For example, if multiple instances of integrations are being used in a server farm, the deployment must be scaled for high usage. For more information, refer to [Scaling for High Usage over Distributed Environments](#). Additionally, it may be required to support proper failover when deploying in a distributed environment.

For technical information on deploying the ISDK and supported Web application servers, refer to the *Oracle AutoVue Integration SDK Technical Guide*.

### 6.1 Scaling for High Usage over Distributed Environments

Depending on the number of concurrent users, the type and size of documents that users typically view, and whether files are to be loaded natively or from streaming files, it may be required to deploy your ISDK-based integration in a server farm. Additionally, it may be required to deploy it in distributed environments. In order to support proper failover in a distributed environment, the HTTP session needs to be replicated across all cluster nodes. In the event that a node fails, a second cluster node takes over and continues to process the requests. Seamless failover is when the user's actions are not disrupted and no authorization dialog is requested during this process.

Depending on the type of DMS integration and connection, login or session information may need to be remembered. For the information to be replicated across nodes, the objects attached to the HTTP session need to be serializable.

For example, in the FileSys sample integration, the `com.cimmetry.vuelink.filesys.FilesysContext` class manages this aspect. The back-end session objects may not always be serializable. The `FilesysContext` stores the username and password strings in the session. This allows the node that is taking over another session to reinitialize the connection with the back-end. The `DMSSession` class is provided by the ISDK to wrap the HTTP session variable. It provides the `setAttribute()` and `getAttribute()` methods to handle the storing of serializable objects to be saved and replicated. For more information, refer to the *Oracle AutoVue Integration SDK Technical Guide*.

Consider the following when serializing objects:

- ❖ The DMS provides a session ID: If the back-end DMS provides a session ID, this session ID can be serialized into the `DMSSession` object. This way when a cluster node fails, the new node can pick up the replicated `DMSSession` and use the stored session ID to continue communicating with the back-end DMS.
- ❖ The DMS connection object is not serializable: If the connection object cannot be serialized into the `DMSSession`, the information needed for recreating this object should be

serialized and replicated. This way, when the active node fails, the second node retrieves this information and recreates the DMS connection object.

If seamless failover is not possible, an authorization exception can be thrown to request the user login information. This way, the user retains the ability to save any markups that they have created provided that they can enter a valid username and password.

Non-serializable objects should be added to the `DMSSession` to increase performance and allow caching of data between requests. However, they need to be declared as transient in order not to break the session replication during failover. These transient objects will need to be regenerated once the session was migrated to a different cluster node.

For information on scaling AutoVue servers for high usage and seamless failover, refer to the "Scaling AutoVue for High Usage" section of the *Oracle AutoVue Planning Guide*.



---

# Updating Existing Integrations to the Java Web Start Client

As an alternative to browser based Java Applets, the new solution is based on "Java Web Start" utility. Java Web Start provides a facility for launching Java-based applications from web browsers, running out of the browser as a separate process. The browser uses file download and file association capabilities to start Web Start automatically. In place of the HTML APPLET tag and attributes, a new "Java Network Launching Protocol" (JNLP) XML file allows developers to specify where the application can be obtained, how it should be launched, and what the initial parameters are. The sections covered in this chapter will help you migrate your Applet based deployment to the new version of AutoVue.

## A.1 Update your Integration

This section will help you migrate your Applet based deployment to the new version of AutoVue. Do the following for updating your integration to the new solution of AutoVue:

1. **Update the server:** Deploy the different AutoVue artifacts on the server side.
2. **Setup the server for SSL mode:** This is required only if you need to run AutoVue under HTTPS protocol.
3. **Test the AutoVue sample:** Test the sample to ensure you have properly deployed all the artefacts on the server side.
4. **Specify Cookies:** Specify cookies that are necessary for authenticating the different components necessary for a seamless integration. These cookies must be specified in the `VueJNLPServlet` so that AutoVue can correctly integrate with your application (single sign on).
5. **Update the client side code:** Update the Client code (customization) to use the new AutoVue *JavaScriptAPI* provided in *autovue.js*.
6. **Security:** Ensure your deployment is secure and follows all the security guidelines of your organization.
7. **Customize your code:** If needed, customize your integration solution to fulfill your specific needs.

### A.1.1 Update the server

Add `VueJNLPServlet` descriptor and mapping to the deployment descriptor file – `web.xml` of the `VueServlet` container. Then setup `VueJNLPServlet` initialization parameters in it. Following are the servlet initialization parameters:

- **URL-Dir:** This parameter refers to the *URL* of AutoVue Client folder with respect to the context root in your Java Servlet container. You may use the integrated jetty servlet

container AutoVue ships with. Make sure your root context is used consistently across your entire deployment. From now on, your root context is *"/AutoVue"*, which would be the value of this *"URL-Dir"* parameter.

---

**Note:** An integrator specific implementation will most certainly not need this parameter defined, as the context root is usually known by the servlet and does not need to be different from the one present inside the JNLP.

---

- **Cookies:** Semicolon ';' separated list of cookie names identifying the cookies to transfer to AutoVue at the start-up:

The rest of initialization parameters are usual AutoVue Client Parameters (also called *"Applet Parameters"*, usually). Refer to the section – [Client Parameters](#) for more information about these parameters. Following is the sample code to be included in the deployment descriptor file *web.xml*:

```
<servlet id="VueJNLPServlet">
<servlet-name>VueJNLPServlet</servlet-name>
  <servlet-class>com.cimmetry.servlet.VueJNLPServlet</servlet-class>
  <init-param>
    <param-name>URL-Dir</param-name>
    <param-value>/AutoVue</param-value>
  </init-param>
  <init-param>
    <param-name>Cookies</param-name>
    <param-value>JSESSIONID;COOKIE2;COOKIE3...</param-value>
  </init-param>
  ...
  <load-on-startup>0</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>VueJNLPServlet</servlet-name>
  <url-pattern>/servlet/VueJNLPServlet</url-pattern>
  <url-pattern>/servlet/VueJNLPServlet/*</url-pattern>
</servlet-mapping>
```

#### Validation:

- Start AutoVue Server
- Start AutoVue Servlet Container (*Jetty*, *Tomcat* or *WebLogic*)
- Connect with a browser to the *URL* of *VueJNLPServlet* and *CodebaseHost* as an argument indicating the codebase URL:

As an example, for Jetty:

```
http://localhost:5098/servlet/VueJNLPServlet?CodebaseHost=http://
localhost:5098/AutoVue
```

If you follow the steps provided above, AutoVue Client should start.

## A.1.2 Setup the server for SSL Mode

When running the AutoVue Client with the rendez-vous servlet, there is no need for the localhost certificate, as was required in previous releases of AutoVue 21.0.2. When running with the rendez-vous servlet, configuring SSL support is specific to the application server being

used to host the AutoVue servlets. See the documentation for the chosen application server to configure it for SSL operation.

### A.1.3 Test AutoVue Sample

A new *HTML* sample page (**av\_jnlp.html**) illustrating a basic AutoVue integration and similar to the usual sample page *autovue.html* is delivered within the files of the new solution.

In order to test this client sample, you should:

- ❏ Start AutoVue Server
- ❏ Start AutoVue Servlet Container (*Jetty*, *Tomcat* or *WebLogic*)
- ❏ Connect with a browser to the *URL* of the new *HTML* sample page *av\_jnlp.html*. As an example, for *Jetty*, the *URL* is

```
http://localhost:5098/AutoVue/av_jnlp.html
```

As soon as the page loads, *Java Web Start* is triggered and launches AutoVue Client standalone outside of the browser. You can then test the file links and the scripting *API*. It should work exactly as it used to do for the Applet sample *autovue.html*, except that AutoVue Client is not embedded in the browser as an Applet.

The new *HTML* sample *av\_jnlp.html* has similar options as *autovue.html*, and they can be customized. Following are the options:

- ❏ CL\_PRTS
- ❏ INIT\_PARAMS
- ❏ ONINIT
- ❏ ONINITERROR
- ❏ USER\_DATA

All these options are explained in detail in the following sections.

#### A.1.3.1 CL\_PRTS

CL\_PRTS should be set to null when using rendez-vous communication.

#### A.1.3.2 INIT\_PARAMS

Similar to "*Applet Parameters*", INIT\_PARAMS is the list of AutoVue Client parameters that is set at initialization stage. They should be provided into a Java Script Object wrapping the parameters to pass. An example follows:

```
INIT_PARAMS = {'FILENAME': '<myFileURL>', 'LOCALE': 'fr_CA'}
```

#### A.1.3.3 USER\_DATA

Custom object to provide within the *start* API and receive within the failure callback **ONINITERROR** below.

#### A.1.3.4 ONINIT

This is a JavaScript callback to register. It will be called when AutoVue starts and listens to JSON-RPC requests in order to handle JavaScript calls. By default it is set to the function **onAvStartup** provided in *av\_jnlp.html* sample. This callback is similar to the one used in the earlier solution and has the same name.

### A.1.3.5 ONINITERROR

This is a *JavaScript* callback to send within the *start* API. It will be called in case AutoVue does not start. As for AutoVue Applet, a suggestion is given in *av\_jnlp.html* (**onAvInitError**) which prompts the user 3 times to retry, then; suggests sending an e-mail to the server administrator notifying him about this failure. This callback is similar to the one used in the earlier solution and has the same name. It must follow the prototype described in the *start* API.

## A.1.4 Specify Cookies

AutoVue Client is not an applet anymore. It cannot access the cookies directly. Any browser cookies that will be needed by the AutoVue Client must be provided at launch time. A new AutoVue Client parameter is introduced to address this. The new parameter "**COOKIES**" holds a list of cookies in a key-value format (*name=value*) separated by a semicolon ";".

The server administrator should specify a semicolon ";" separated list of cookie names passed as init parameter of *VueJNLPServlet*, named *COOKIES* as well. This init parameter supports two special values:

- **true:** Pass all the browser cookies of the domain to AutoVue (this must be combined with filtering in the *VueJNLPServlet* and encryption to ensure a secure deployment).
- **false:** Do not pass any cookie to AutoVue

Basically, add cookies' names to the server configuration file *web.xml*:

```
<servlet id="VueJNLPServlet">
  <servlet-class>com.cimmetry.servlet.VueJNLPServlet</servlet-class>
  <init-param>
    <param-name>Cookies</param-name>
    <param-value>JSESSIONID;...</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
```

The required cookies' values will be collected by the "*GET*" method of *VueJNLPServlet* during the client start-up and passed to the client within the new Client parameter (*COOKIES*).

## A.1.5 Update client side code

In order to update the client side code, you have to do the following:

- Remove AutoVue Applet tag and related code, after that include *autovue.js* as follows:

```
<script type="text/javascript" src="graphics/autovue.js"></script>
```

Instantiate an *AutoVue* object within a *JavaScript* block as in *av\_jnlp.html* (refer to the section – [AutoVue Constructor Parameters](#) for more information about the constructor parameters):

```
var myAvApp = new AutoVue(<VueJNLPServlet Host>, <Codebase Host>, ...);
```

- Call the **start** method to launch an AutoVue Client (refer to the *JavaScript API* section of the *Oracle AutoVue API Guide* for more information about the function parameters):

```
myAvApp.start(onInit, onFail, user_data);
myAvApp.setFile(<URL of a file to load in AutoVue>);
```

Update the calls to AutoVue Applet scripting *API* using the new *AutoVue* Object. Refer to the *JavaScript API* section of the *Oracle AutoVue API Guide* for a complete description of AutoVue scripting *API* supported by *AutoVueJavaScript* Object. An example follows:

**Validation:** Start AutoVue Server, AutoVue Servlet Container and launch AutoVue Client using your integration solution. It should work as before and AutoVue will start in a separate window.

---

**Note:**

- If you call the Scripting API directly without starting AutoVue, the *JavaScriptAutoVue* Object will start AutoVue client automatically for you before invoking the scripting API.
  - You can recycle this *JavaScriptAutoVue* Object across your *HTML* pages. The object will remember its previous state. For example, if you had previously connected to AutoVue, the JavaScript will remember the port used for that. The JavaScript will re-connect to a new AutoVue application using the previously defined parameters in case AutoVue was closed.
- 

## A.1.6 Customizing AutoVue

The new solution provides customization capabilities similar to AutoVue applet solution. Following are the customizable options:

- Client Parameters
- AutoVue Constructor Parameters

The sections - Client Parameters and AutoVue Constructor Parameters discuss the customizable options.

### A.1.6.1 Client Parameters

The new solution provides the following mechanism to set Client Parameters (usually set in the applet and known as "**Applet Parameters**"):

- Client Setting

**Client Setting:** Pass the parameters to the constructor of *AutoVueJavaScriptObject*. Refer to the section – [AutoVue Constructor Parameters](#) for a description of expected parameters format.

## A.1.7 AutoVue Constructor Parameters

Additional options are specific to the new *API* and the new solution. They are available in *autovue.js*. They can be passed as arguments to the constructor of *AutoVueJavaScript* Object:

```
var myAvApp = new AutoVue(JNLP_HOST, CODEBASE_HOST , CLIENT_PORTS, INIT_PARAMS,
ECNRYPT_COOKIES, VERBOSITY, STARTUP_DELAY);
CSI_ClbSessionSubject=Subject;CSI_ClbSessionType=public|private;CSI_
ClbUsers=user1,user2,x;
```

The [Table A–1](#) provides the list of AutoVue Constructor Parameters.

**Table A–1    AutoVue Constructor Parameters**

Parameter	Description	Default Value
JNLP_HOST	Specifies the URL of VueJNLPServlet deployed within AutoVue Servlet container. For example, for an AutoVue fresh installation, this URL would be <code>http://localhost:5098/servlet/VueJNLPServlet</code> . This parameter is mandatory. It is needed to connect to VueJNLPServlet servlet that generates the JNLP file required to trigger Java Web Start. AutoVue Client cannot start without the required JNLP file.	
CODEBASE_HOST	Specifies the URL of AutoVue Client codebase ( <code>jvue.jar</code> , <code>jsonrpc4j.jar</code> , <code>jogl.jar</code> , <code>gluegen_rt.jar</code> ). For example, this URL would be <code>http://localhost:5098/AutoVue</code> . This parameter is also mandatory. Without it, Java Web Start will not be able to find AutoVue Client's codebase in order to launch it.	
CLIENT_PORTS	This parameter is obsolete and should always be set to null.	No default. The default value could also be null.
INIT_PARAMS	These replace the AutoVue applet parameters. This is the list of AutoVue Client parameters to be set at an initialization stage. They should be provided wrapped into a JavaScript object.	null
ENCRYPT_COOKIES	This option has been deprecated, and is not supported when using the rendez-vous connection between the browser and the AutoVue Client. It should be set to false in rendez-vous based environments.	true
VERBOSITY	This parameter specifies how <code>autovue.js</code> should output error messages: <ul style="list-style-type: none"> <li>⌘ None</li> <li>⌘ Browser Console</li> <li>⌘ JavaScript alert popup</li> <li>⌘ Both Browser Console and JavaScript alert popup</li> </ul>	1
STARTUP_DELAY	AutoVue is launched using <i>Java Web Start</i> . The start-up process can take some time to complete since the java classes ( <code>jars</code> ) have to be downloaded to the client machine and the browser may prompt the user before starting any download. At the same time, AutoVue JavaScript Object tries to establish communication with AutoVue Client through JSON-RPC to detect when it is ready to handle scripting methods. This variable specifies the required delay before assuming a start-up failure of AutoVue Client (For example, AutoVue Server not running). It is set by default to 30 seconds.	30

## A.2 Steps for Integration

For partners who have developed an integration to link AutoVue to a file repository (typically using the *ISDK* toolkit), the recommendation is to follow the incremental process provided in [Table A–2](#) for migrating the implementation to the new *Web Start* based AutoVue Client.

**Table A–2 Steps for integrating with the new Web Start-based AutoVue Client**

Step	Tasks	Expected Result
AutoVue Client Launch	<p>The initial goal is to have the AutoVue Client launched in the integrated environment through the <i>Web Start</i> framework.</p> <p>Actions:</p> <ul style="list-style-type: none"> <li>▣ Install and configure <i>VueJNLPServlet</i> and the template file it uses on the application server of your environment.</li> <li>▣ Validate <i>JNLP</i> file produced by <i>VueJNLPServlet</i>.</li> <li>▣ Verify that AutoVue Client can be launched from client machines.</li> </ul>	An empty AutoVue Client can be launched from the embedding environment. Local files can be opened from the user interface of AutoVue.
Basic File Viewing	<p>The second step is to enable the viewing of files from the document repository. The implementation work here will involve passing authentication and document identification information to the AutoVue client.</p> <p>Actions:</p> <ul style="list-style-type: none"> <li>▣ Adapt viewer launch page in existing integration to invoke <i>VueJNLPServlet</i> with parameters that are currently added as applet parameters.</li> <li>▣ Validate that viewer can be launched with files chosen by the user through the user interface of the embedding system.</li> </ul>	Files can be selected from the document repository and viewed in the AutoVue client.
Migrate Additional Functionality	<p>Most integrations provide additional document related functionality through the use of the <i>LiveConnectAPI</i> of AutoVue. These functions may include markup, printing, comparison, etc. The <i>JSON-RPCAPI</i> provides the same functionality as the <i>LiveConnectAPI</i>.</p> <p>Actions:</p> <ul style="list-style-type: none"> <li>▣ Identify all <i>LiveConnectAPI</i> usage in the current integration. Generate a checklist of user operations that are built through these capabilities.</li> <li>▣ Convert integration code that was using the AutoVue applet's <i>LiveConnectAPI</i>s to invoke the functionality through the <i>JSON-RPC</i> interface. We recommend using the provided AutoVue <i>JavaScript</i> class to simplify this process.</li> <li>▣ Validate that all checklist items have been successfully migrated. Provide feedback to Oracle for any functionality that cannot be migrated.</li> </ul>	Revised integration should match functionality of original integration to a very high degree. Any deficiencies have been reviewed and a remediation plan established.

**Table A–2 (Cont.) Steps for integrating with the new Web Start-based AutoVue Client**

Step	Tasks	Expected Result
Refine Security	<p>The migration of the AutoVue integration from applet-based to <i>Web Start</i> may allow for an improvement in the overall system security. If applicable, the following actions may be taken:</p> <ul style="list-style-type: none"> <li>⌘ Revise security implementation to make use of <i>VueJNLPServlet</i> cookies capabilities.</li> <li>⌘ Restore <i>HTTPOnly</i> attribute on authentication related cookies if it had been removed for AutoVue applet use.</li> </ul> <p>Enable cookie encryption feature:</p> <ul style="list-style-type: none"> <li>⌘ Decide on an approach for generating the required key-pair information.</li> <li>⌘ Enable cookie encryption in <i>VueJNLPServlet</i> invocation.</li> </ul>	<p>Security related cookies are protected through <i>HTTPOnly</i> attribute. Security information in <i>JNLP</i> files is secured to a level appropriate for the operating environment.</p>



If you have any questions or require support for AutoVue, please contact your system administrator. If the administrator is unable to resolve your issue, please contact us using the links below.

### B.1 General AutoVue Information

---

Web Site	<a href="https://www.oracle.com/applications/autovue/">https://www.oracle.com/applications/autovue/</a>
----------	---

---

### B.2 Oracle Customer Support

---

Web Site	<a href="https://www.oracle.com/support">https://www.oracle.com/support</a>
----------	---

---

### B.3 My Oracle Support AutoVue Community

---

Web Site	<a href="https://community.oracle.com/hub/">https://community.oracle.com/hub/</a>
----------	---

---

### B.4 Sales Inquiries

---

E-mail	<a href="https://www.oracle.com/corporate/contact/global.html">https://www.oracle.com/corporate/contact/global.html</a>
--------	---

---

