Oracle® AutoVue

Integration Guide

Release 21.1.0

F86920-01

September 2023

ORACLE

Oracle AutoVue Integration Guide, Release 21.1.0

F86920-01

Primary Author:

Contributing Author:

Contributor:

Contents

3 Integration Scenarios

4 FAQ

A Feedback

Preface

This document provides an introduction to the different deployment methods and integration technologies for Oracle AutoVue.

Audience

This document is intended for Oracle partners and third-party developers (system integrators) who would want to implement integrations with Oracle AutoVue.

Related Documents

For more information, see the following documents in the documentation set on OTN:

- *AutoVue API and ABV Developer's Guide*
- *AutoVue ISDK Overview and Installation Guide*
- *AutoVue ISDK Technical Guide*
- *AutoVue Web Services Installation and Developer's Guide*

Conventions

The following text conventions are used in this document:

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

1 Introduction

Oracle AutoVue is a versatile enterprise visualization application that can be integrated with a Document Management System (DMS)[1] or deployed as a standalone application. You can modify the functionality of Oracle AutoVue's client in order to create your own customized application or you can trigger actions from a DMS via hotspots. As can be seen from the following list, there are a number of integration options available for Oracle AutoVue.

- **AutoVue Integration Software Development Toolkit**
  The Autovue Integration Software Development toolkit (ISDK) allows you to build a custom integration between Oracle AutoVue and a DMS. The AutoVue ISDK is a Java-based implementation of the Document Management Application Programming Interface (DMAPI) specifications published by Oracle.

  Oracle also provides an AutoVue ISDK-based integration, VueLink, that can be used out-of-the-box for specific environments. As of the current release, Oracle provides a VueLink integration for Oracle WebCenter Content, and for Documentum. For more information on these VueLinks, refer to the "AutoVue Documentation" page on the https://www.oracle.com/technical-resources/documentation/autovue.html.

- **AutoVue Web Services**
  The AutoVue Web Services (WS) package allows you to integrate AutoVue's capabilities into your application regardless of platforms or programming languages.

- **AutoVue Java-based Application Programming Interface**
  The AutoVue Java-based Application Programming Interface is a Java-based toolset that provides tools to modify the functionality of Oracle's AutoVue client. It also allows you to create your own customized Java applications based on AutoVue API components.

- **AutoVue JavaScript Application Programming Interface**

  The AutoVue JavaScript Application Programming Interface is a JavaScript-based toolset that provides tools to launch and interact with AutoVue from a WEB context. It also allows you to create your own customized WEB-based client acting on an AutoVue client.

- **Augmented Business Visualization**
  Oracle's Augmented Business Visualization (ABV) solution creates bi-directional links between parts of a document and an enterprise application. This is done by integrating AutoVue with an enterprise application to query the DMS through actionable hotspots or by highlighting assets. These hotspots can be used to trigger actions or business processes in the DMS from the documents.

The following chapters provide an overview of AutoVue integration technologies and AutoVue deployment methods, as well as sample integration scenarios.

[1] This document uses the term Document Management System (DMS) to refer to Enterprise Resource Planning (ERP) system, Product Lifecycle Management (PLM), and repositories.

2

AutoVue Integration Technologies

This chapter discusses the deployment options for Oracle AutoVue as well as the available integration technologies.

2.1 Oracle AutoVue

Oracle AutoVue can be deployed in two modes:

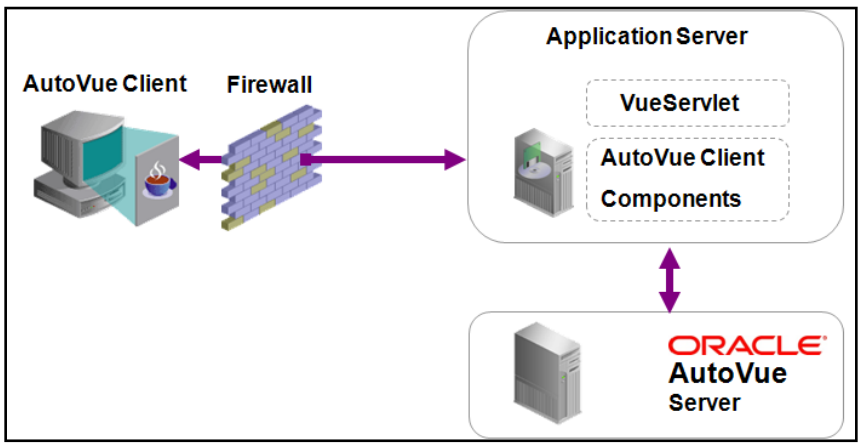
- **Client/Server Deployment**: Oracle AutoVue is installed on a server, to which client machines connect to access and view documents. You can then run Oracle AutoVue as a standalone application or integrate it with a DMS.
- **Desktop Deployment**: Oracle AutoVue is installed on a user's local machine.

> **Note:** The following sections provide an introduction to these deployment modes. For installation, system requirements, and configuration information, refer to their respective *Installation and Configuration Guides*.

2.1.1 Client/Server Deployment

The Oracle AutoVue Client/Server (CS) deployment provides a complete, open and standards-based set of integration tools that allows you to tie AutoVue to any enterprise application. This deployment provides you with a consistent view of data and business objects, and expands workflow automation to document-based processes and can be deployed either as a standalone application or integrated with a DMS.

Figure 2–1 Client/Server Deployment Flow Diagram



2.1.1.1 Deployment Architecture of Oracle AutoVue Client/Server Deployment

Oracle AutoVue CS has several components: the AutoVue server, an application server hosting the VueServlet, JNLP Generator, Rendezvous Servlet, and DMS, a Web server or an application server hosting AutoVue client components, and the AutoVue client. These components are explained in the following sections.

2.1.1.1.1 AutoVue Server The AutoVue server is the core of the AutoVue solution and is comprised of one session server process and multiple document server processes. The session server receives all requests sent to the AutoVue server and then delegates the tasks to a document server.

You can ensure high availability by setting the AutoVue servers in a server farm. In this configuration, the session servers communicate with each other to distribute the load across all the document servers in the server farm.

Take note of the following considerations when determining on a machine to host the AutoVue server:

- The AutoVue server is very CPU, I-O, Memory and Graphics intensive. Ensure that the machine hosting AutoVue server is dedicated to AutoVue and is not being used for other applications.
- If the AutoVue server is running in a virtualized environment, ensure that the proper resources are allocated to the virtual machine hosting it. Note that virtualized environments may have slight degradation in performance compared to running on native hardware.

> **Note:** For more information on the AutoVue server and how to configure it, refer to the “Appendix A: AutoVue Server Configuration” section in the *Oracle AutoVue Client/Server Deployment Installation and Configuration Guide*.

2.1.1.1.2 VueServlet The VueServlet is the main entry point for communication between AutoVue clients and the AutoVue server. When used by external AutoVue clients to communicate with the AutoVue server, the VueServlet must be configured for access through a firewall. Generally, the VueServlet can be deployed on any application server.

> **Note:** For a list of application servers that are certified by Oracle, refer to section "System Requirements" in the *Oracle AutoVue Client/Server Deployment Installation and Configuration Guide*.

When deploying the VueServlet, your deployment steps should generally depend on whether you have integrated AutoVue with a DMS, or whether you are using it in a non-integrated environment. When AutoVue is integrated with a DMS, it is recommended to deploy the VueServlet on the application server (in a different context) that hosts the DMS.

> **Note:** For information on deploying VueServlet in an integrated environment, refer to the “Installing the VueServlet in an Integrated Environment” section of the *Oracle AutoVue Client/Server Deployment Installation and Configuration Guide*.

Depending on your peak usage, you may need to have multiple VueServlets that can serve requests to AutoVue. Since the VueServlet is hosted within an application server, you must rely

on the application server's load balancing capabilities or rely on an external load balancer. Note that you must ensure the load balancer is configured for session/cookies stickiness (not the IP address).

2.1.1.1.3 JNLP Generator AutoVue Client can be launched from a WEB environment. You can interact from a WEB environment. It is a Java application that can be started through Java Web Start framework. This requires a JNLP file to be downloaded by the browser in order to trigger the utility javaws through file protocol association. The servlet VueJNLPServlet provided with AutoVue is designed to generate the required JNLP file.

> **Note:** For more information about VueJNLPServlet, JNLP file specifications and its generation, refer to the "Deploying JNLP Components" section of the *Oracle AutoVue Installation and Configuration Guide.*

VueJNLPServlet can be deployed onto any J2EE application server supported for VueServlet. It is an independent self-contained component that does not interact with any other component of AutoVue server. It is provided as a reference implementation for validation and testing purposes.

2.1.1.1.4 AutoVue Client Components AutoVue client components need to be hosted within an application server or a Web server and configured to communicate with the VueServlet, which in turn communicates with the AutoVue server. The process for deploying the client components varies depending on whether you have AutoVue integrated with a DMS or if you are using a non-integrated environment.

> **Note:** For information on deploying AutoVue client components, refer to the "Installing AutoVue Client Components" section of the *Oracle AutoVue Client/Server Deployment Installation and Configuration Guide*.

AutoVue Client

The AutoVue client is a JAVA-based application that can be launched from a Web page (HTML, ASP, and so on), through Java Web Start framework, and is fully customizable. You can modify the graphical user interface (GUI), set up a collaboration session, modify the menu options and toolbars, and so on.

> **Note:** For more information about the customizing the client application and GUI, refer to the "Customizing the AutoVue Client" section of the *Oracle AutoVue Client/Server Installation and Configuration Guide*

2.1.2 Desktop Deployment

Oracle AutoVue Desktop Deployment is a solution for users that want to run AutoVue locally on their individual desktops.

AutoVue provides you the option of customizing your graphical user interface (GUI). By default, the GUI specification is not set and AutoVue uses an internal GUI file for the menus and toolbars.

> **Note:** For information on modifying the GUI file, refer to the "Customizing the GUI" section of the *Oracle AutoVue, Desktop Deployment Installation and Configuration Guide*.

2.2 AutoVue Integration Software Development Kit

The AutoVue Integration Software Development Toolkit (ISDK) enables you to add powerful viewing and markup capabilities to the DMS by interfacing AutoVue with a particular DMS. AutoVue ISDK provides a framework on top of which you can build your own integration between AutoVue and a DMS. This interface, or integration process, is composed of several activities: requirements specification, analysis, design, implementation, testing and maintenance.

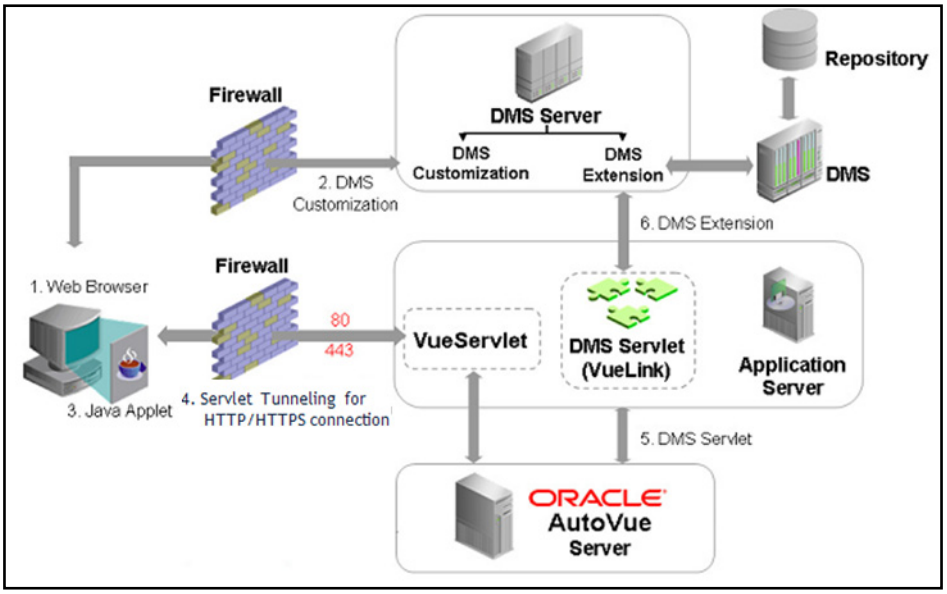
The following sections provide an overview of the ISDK architecture, framework, and sequence flow.

Note: For a more detailed description of the ISDK, refer to the *AutoVue ISDK Overview and Installation Guide.*

2.2.1 AutoVue ISDK Architecture

The following block diagram shows a typical integration between AutoVue and a DMS using the AutoVue ISDK framework.

Figure 2–2 ISDK integration between AutoVue and a DMS



The following is a description of the integration shown in Figure 2–2, "ISDK integration between AutoVue and a DMS". The numbered steps in the figure correspond to the following steps.

1. Log into the DMS through a Web browser.
2. With DMS customization in place, you are presented with a link labeled **View** next to each file stored inside DMS. This link allows you to view files in the AutoVue viewer.
3. Click **View**. The AutoVue client is launched in a separate window.
4. The AutoVue client communicates with the AutoVue Server through servlet tunneling for HTTP/HTTPS connection (VueServlet).
5. The AutoVue server then communicates with the DMS servlet using a standard HTTP/HTTPS connection.

6. With the DMS extension installed on the server machine, the DMS Servlet is able to talk to the DMS Server to handle any request made by the AutoVue server, such as file fetching.

If you try to view a file having external reference files (XREFs) or font resource files, the DMS Servlet retrieves those files and makes them available to the AutoVue server.

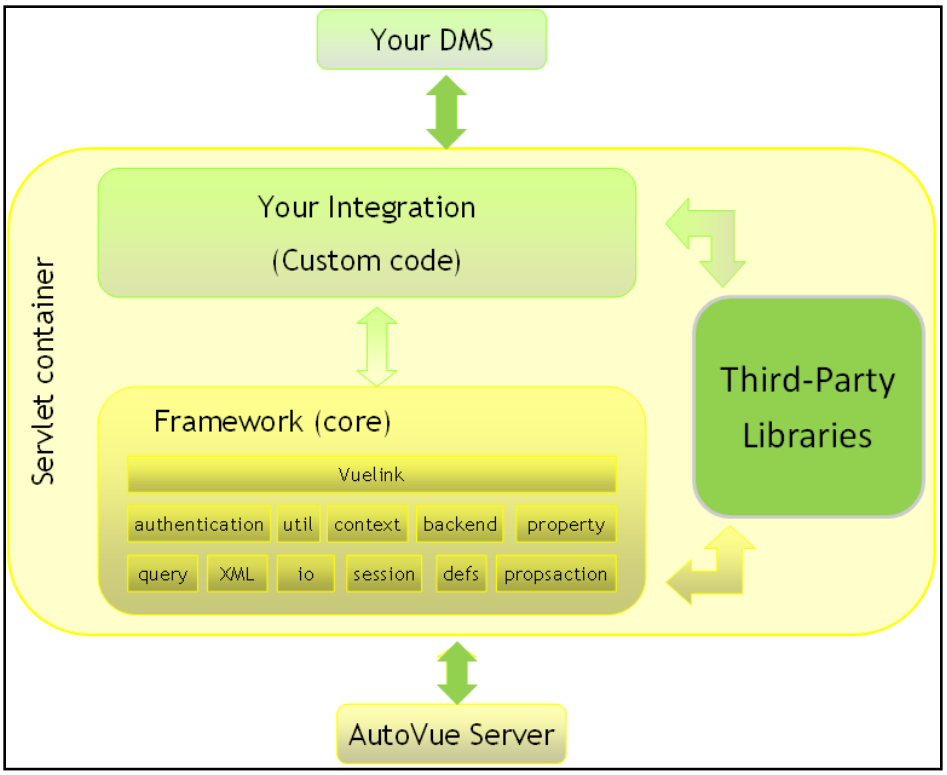
Once the file and all its related XREFs and/or resources are fetched out of the DMS, they are processed by the AutoVue server, which renders the file and streams it to the AutoVue client for display.

2.2.2 AutoVue ISDK Framework

The AutoVue ISDK framework handles all the plumbing for parsing XML requests received from the AutoVue Server, as well as constructing XML responses sent back to the AutoVue server. This framework provides the foundation you need to build your own integration so that you do not have to implement your integration from scratch.

The following block diagram shows the internal structure of a typical integration with a DMS.

Figure 2–3 Integration internal structure



The AutoVue ISDK bundles some third-party Java libraries needed by the framework. These libraries are also available for you to call from your own code.

Your integration is responsible for interacting with your DMS. Depending on what type of SDK your DMS provides, such interaction can be as easy as calling your DMS Java libraries.

2.2.3 AutoVue ISDK Sequence Flow

When a user selects a document to view, the AutoVue server makes several requests to the DMS servlet. The DMS servlet provides a response for each request. The scenario of the

exchanges established between the AutoVue server and the DMS servlet are summarized as follows:

1. The AutoVue server asks for the private key. This request is handled by VueLink core.
2. The AutoVue server asks for the user name (CSI_UserName).
3. The AutoVue server asks for the document ID (DocID) of the selected document. This is done through the **Open** action, which obtains the DocID from the DMS.
4. The AutoVue server asks for some properties of the document, such as document name, document size and date of the last modification. The reason is that the AutoVue server maintains a cache of the document and needs to know if it already has the exact same version of the document in its cache. In this case, AutoVue uses the cached copy rather than redownloading the document.
5. AutoVue fetches the document through the **Download** action.
6. AutoVue downloads XREFs.
7. AutoVue queries for any markups associated to the document.

2.2.4 VueLink

The VueLink is an Oracle-developed ISDK for specific integration environments. VueLink provides an interface that allows communication between a DMS and AutoVue in order to retrieve documents and to store data that is generated by AutoVue for those documents (for example, annotations). The VueLink is a Java Web application that is hosted on a Java Web application server.

For a list of currently available VueLinks, refer to the "AutoVue Documentation" page on the Oracle Technology Network (OTN) https://www.oracle.com/technetwork/documentation/autovue-091442.html.

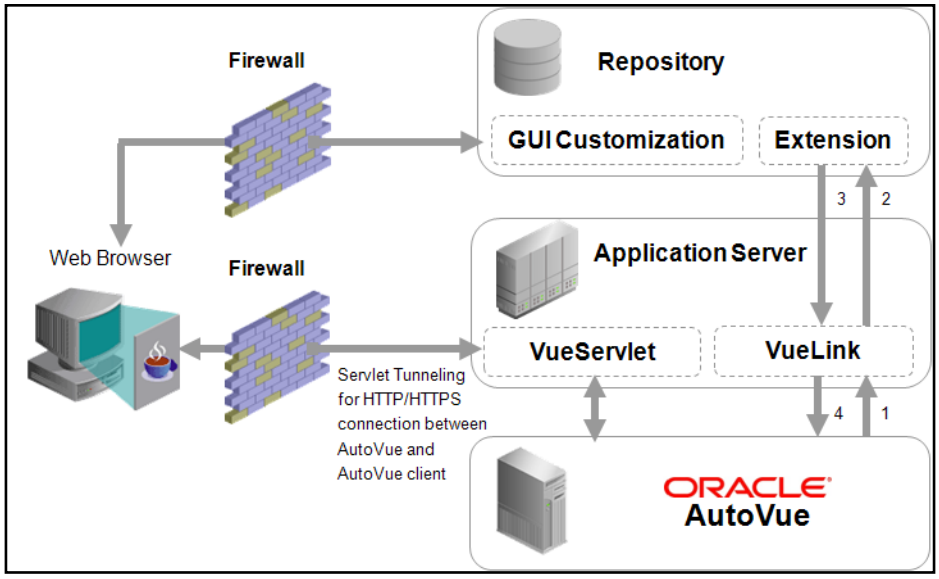
This section provides an overview of the VueLink architecture and how to customize the graphical user interface (GUI).

Note: For a more detailed description of VueLink components, refer to the *AutoVue ISDK Overview and Installation Guide*.

2.2.4.1 VueLink Architecture

The following figure shows how the communication between AutoVue and the DMS/repository is done through a VueLink.

Figure 2–4 AutoVue and DMS Integration



The integration flow is as follows:

1. AutoVue sends a request to the VueLink.
2. VueLink forwards the request to the repository.
3. The repository sends a response back to the VueLink.
4. VueLink forwards the response to the AutoVue server.

Once AutoVue gets access to a document and other related data from the DMS, it then streams the view of the document to the AutoVue client via the VueServlet.

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. You can choose your 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 DMS. The VueLink also enables AutoVue to retrieve any data related to these documents from the DMS. In addition, any data generated by AutoVue (for example, markups and renditions) can be stored into the DMS using this component.

2.2.4.1.1 GUI Customization The AutoVue client can be embedded into the repository GUI or it can be launched in a separate window. In either case, an action must be defined in the repository GUI that invokes the AutoVue client. This action can be assigned to a user interface (UI) button, an icon, or a menu item inside the repository UI.

Note: For more information on VueLinks and GUI customization, refer to the *AutoVue ISDK Overview and Installation Guid*e.

2.3 AutoVue Web Services

The AutoVue Web Services (WS) package enables AutoVue to communicate with a third-party application that wants to invoke AutoVue in a Service Oriented Environment (SOE). It exposes certain AutoVue functionalities as Web methods, and translates AutoVue Web Services

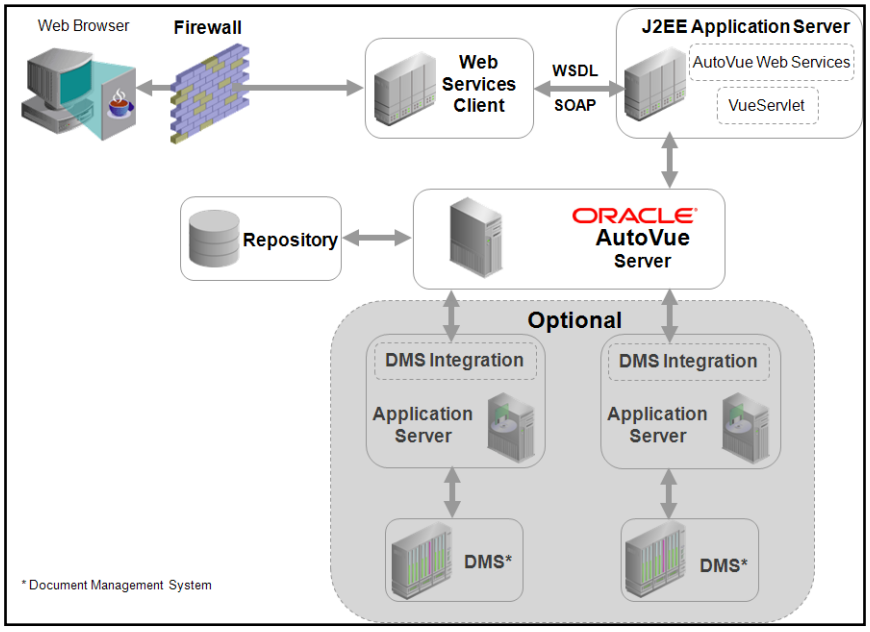
requests to and from AutoVue (for example, AutoVue messages to AutoVue Web Services responses).

Note: This section provides an overview of the AutoVue WS architecture. For information on how to implement AutoVue WS into your integration, refer to the *AutoVue Web Services Installation and Developer's Guide*.

2.3.1 AutoVue Web Services Architecture

The following diagram displays the communication process for AutoVue WS.

Figure 2–5 AutoVue Web Services communication process

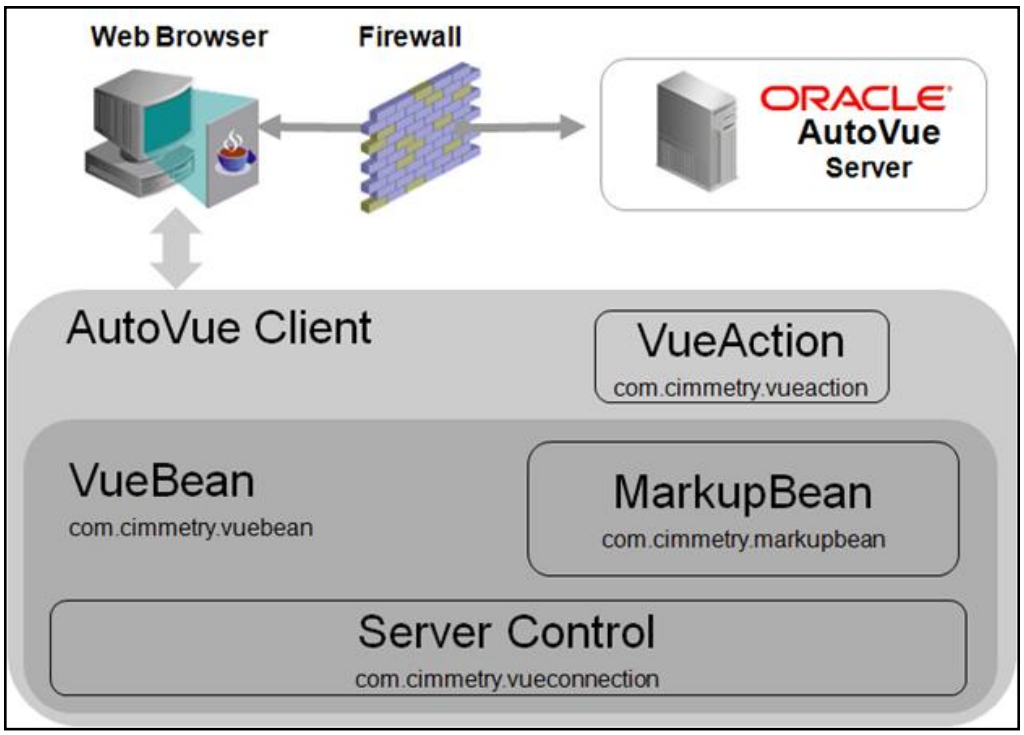


1. The Web Services client communicates with the Web Services/Application Server via Web Service Definition Language (WSDL)/Simple Object Access Protocol (SOAP).
2. All requests are then sent to the AutoVue Server.
3. If there is a the DMS extension installed on the server machine, the DMS Servlet is able to talk to the DMS Server to handle any request made by the AutoVue server, such as file fetching from the DMS.

2.4 AutoVue Java-based Application Programming Interface

The AutoVue Java-based Application Programming Interface (API) is an umbrella term for the APIs that the AutoVue client is built upon, with the VueBean API being the core component of the architecture. The following figure exposes these API packages and classes.

Figure 2–6 AutoVue API Packages and Classes



VueAction: This component can add graphical user interface (GUI) elements to different contexts (such as menu bar, tool bar, status bar, and so on). For example, when a menu option is selected in the GUI, a VueAction is triggered.

VueBean: This component manages the representation of a file including the resources upon which the file depends.

MarkupBean: This component handles markup functionality.

Server Control: This component handles the communication with the AutoVue server and the session book keeping.

The AutoVue client that ships with AutoVue Client/Server Deployment is an example of web-based AutoVue client. As seen in Figure 2–6, "AutoVue API Packages and Classes", there are a number of packages and classes included in the AutoVue API.

Note: This section provides an overview of the AutoVue API. For more technical information, refer to the *Oracle AutoVue API and ABV Developer's Guide*.

2.4.1 AutoVue API Design Options

With the AutoVue API, you can implement functions from AutoVue in a WEB client, build a customized application, or customize your AutoVue client. These design options are introduced in the following sections.

Note: For detailed information on how to implement these design options, refer to the *Oracle AutoVue API and ABV Developer's Guide*.

2.4.1.1 Implementing Functions from AutoVue in a WEB client

You can modify the functionality of the client that is shipped with Oracle AutoVue. This option is used to build additional menu and toolbars outside of the AutoVue client's interface. You can design a standalone application or a WEB client in a Web page.

When creating your own customized Java clients based on AutoVue API components, it is sometimes easier to implement pre-existing methods from AutoVue. Using AutoVue JavaScript API, many AutoVue methods can be called in your HTML page.

Note: For more information on how to implement functions from AutoVue in WEB client, refer to the *Oracle AutoVue API and ABV Developer's Guide*.

2.4.1.2 Building an AutoVue API Application

You can build your own customized application. This option allows you to build an application that makes calls to the VueBean package. You can leverage AutoVue's viewing and markup technology while maintaining complete control of the behavior of the application.

Note: For detailed information on how to create an application that opens and displays a file using the AutoVue API, refer to the "Building an AutoVue API Application" section of the *Oracle AutoVue API and ABV Developer's Guide.*

2.4.1.3 Customizing AutoVue

You can implement pre-existing methods from Oracle's AutoVue client to build your own customized client. This option is used to customize the existing AutoVue client's menus and toolbars.

Note: For detailed information, refer to the "AutoVue API Packages" section of the *Oracle AutoVue API and ABV Developer's Guide*.

2.5 AutoVue JavaScript Application Programming Interface

AutoVue Client is a Java application that can be started through Java Web Start framework. This framework requires a JNLP file to start the application. An integration solution requires that the server generate a JNLP file to be used by Java Web Start framework to launch AutoVue client. The servlet VueJNLPServlet provided with AutoVue is designed to generate the required JNLP file.

Note: For more information about VueJNLPServlet, JNLP file specifications and its generation, refer to the "Deploying JNLP Components" section of the *Oracle AutoVue Installation and Configuration Guide*.

AutoVue Client proposes the following method to the HTML client to invoke it:

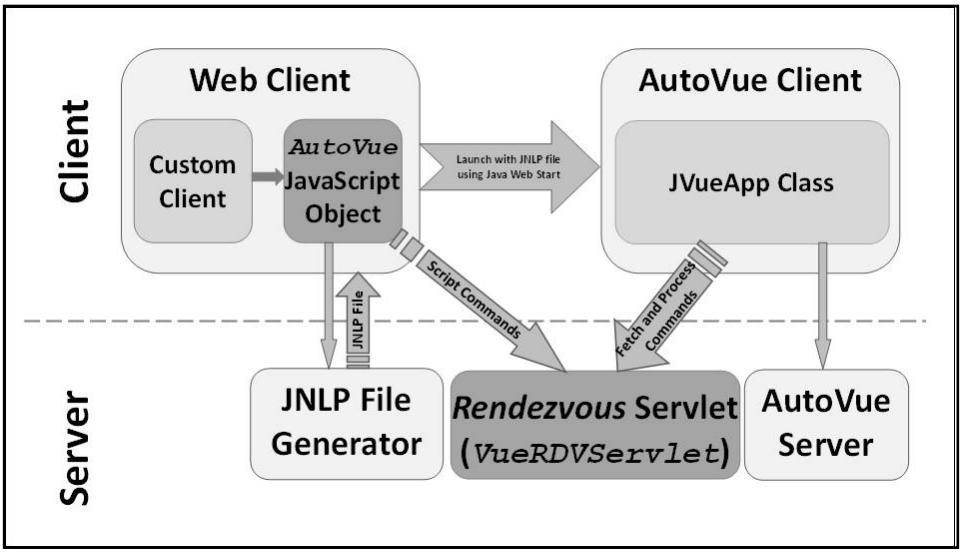
- Rendezvous communication: Through a Rendezvous servlet.

AutoVue provides a JavaScript Object named **AutoVue** and implemented in the file autovue.js. This object supports both communications approaches and simplifies the integration of AutoVue into a WEB context. It provides a JavaScript API that browser could use to interact with AutoVue.

Rendezvous Communication: The HTML client and the AutoVue Client communicate together through a "Rendez-Vous" servlet named ***VueRDVServlet***, deployed on server side. They share a common "Rendez-Vous" ID used by the Rendezvous servlet to link them together.

The Figure 2–7 shows how the communication between the browser and AutoVue is done through a Rendezvous servlet:

Figure 2–7 Architecture diagram for Rendezvous Communication



Note: For more information about AutoVue JavaScript API, refer to the "Java Script API" section of the *Oracle AutoVue API and ABV Developer's Guide*.

2.6 Augmented Business Visualization

Augmented Business Visualization (ABV) is a visualization framework which provides rich and actionable visual decision making environments by connecting portions of documents to business data found in enterprise applications. ABV's hotspot capabilities allow you to create links between objects in AutoVue's data model and objects in an external system. With this hotspot feature, an ABV solution can be built that integrates AutoVue tightly into other applications. By clicking on an area of a document in AutoVue, a visual action is triggered and/or information displays in other applications. With visual dashboards, you can expose data from enterprise systems visually by changing the hotspot color.

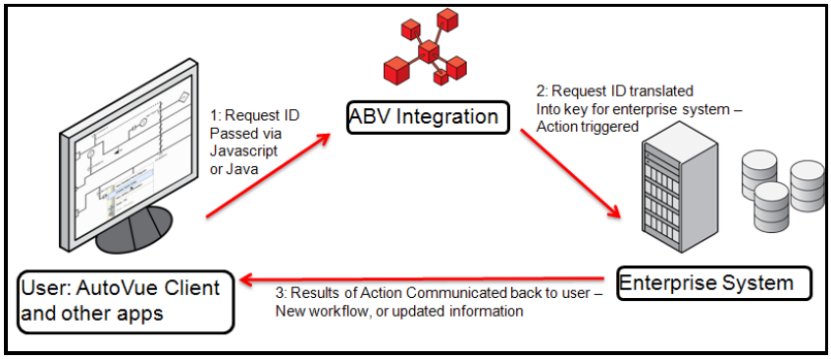
Note: This section provides an overview of the ABV architecture and hotspots. For more detailed information, refer to the *Oracle AutoVue API and ABV Developer's Guide*.

2.6.1 Architecture

Through the use of AutoVue hotspot capabilities, the ABV integration framework ties together all the individual components that make up the solution (data repository, AutoVue, and enterprise applications). The ABV integration coordinates with the DMS and can be written in either Java or JavaScript.

Depending on how the ABV framework is configured, the hotspots can be set up as a visual action solution or as a visual dashboard solution.

Figure 2–8



For visual actions, information is retrieved from the data repository when a user clicks on a text hotspot, regional hotspot, or 3D hotspot. A request identifier is passed to the ABV integration by the AutoVue client application. The ABV integration then translates the request ID into a key for the enterprise system and an action is triggered. The results of the action is communicated back to the user from the enterprise system.

For visual dashboards, the ABV integration retrieves information for the dashboard from the enterprise system. This information is then passed to the hotspot identifiers and colors via JSON or XML.

2.6.2 Hotspots

ABV's hotspot capabilities allow system integrators to create links between objects in AutoVue's data model and objects in an external system. With this hotspot feature, an ABV solution can be built that integrates AutoVue tightly into other applications. By clicking on an area of a document in AutoVue, a visual action is triggered and/or information displays in other applications. With visual dashboards, you can expose data from enterprise systems visually by changing the hotspot color.

AutoVue provides the following capabilities around hotspots:

- Defining hotspots.
- Defining Visual Dashboard.
- Tooltip to display on the hotspots defined in AutoVue.
- Customizing for hotspot selection notification.
- Customizing of hotspot color.
- Defining visual actions.

AutoVue provides the ability to define the following types of hotspots:

- Text-based hotspots in 2D and EDA (based on patterns of text in the document/PDF).
- Regional (box and polygon) hotspots in 2D, EDA, PDF and graphic documents are defined by world-coordinates on the page.
- Hotspots based on 3D attribute names and/or values.
- Hotspots in Web CGM files which can be defined by the Name, ID, or URI parameters.

Note: For more information on creating visual dashboards and visual actions, refer to the "Hotspots" section of the *Oracle AutoVue API and ABV Developer's Guide*.

Note: Use of hotspots with optical character reader (OCR) is not recommended.

3

Integration Scenarios

This section discusses possible integration scenarios for AutoVue. For technical information and code snippets on how to implement these integrations refer to their respective technical documents mentioned in the Preface.

3.1 AutoVue API Use Case

The following example shows how an AutoVue API solution can be used to print documents from a DMS.

Note: For sample code on how to implement the AutoVue API, refer to the "Sample Cases" section of the *Oracle AutoVue API and ABV Developer's Guide*.

3.1.1 Printing Documents from a DMS

A customer in the oil and gas industry needs to print designs in large plants where printers are dispersed, and different departments have different printing needs. Documents are stored in a DMS and a custom application provides access to files and handles integration with AutoVue.

3.1.1.1 Requirements

The customer is in need of a batch print solution. However, a single server-side printing solution is not possible since the different business units have different business needs and are located in different physical locations. Instead, given that the printers are dispersed around the plants, the customer needs to have a batch print capability on the client side where users can choose the appropriate printer in their local environment.

3.1.1.2 Integrator steps

1. Create a custom ISDK integration to fetch documents from the DMS. Note that each document has a unique ID which represents it in the repository and the custom integration has the list of these Doc ID's to be passed to the AutoVue integration.
2. Build a custom VueBean batch printing solution which calls a non-GUI session to carry out the AutoVue VueBean batch print method.
3. Display a dialog with all print options so the end user can choose the appropriate options to print the documents.

3.1.1.3 End-user steps

1. An engineer prints multiple DWG, DGN and TIFF files which are all related to a certain design in their enterprise application. To do so, the engineer navigates to the custom application interface, selects the files, and then clicks **Bulk Print**.
2. For the next design, the custom application's bulk print feature is used to print the contents of the entire folder stored in the DMS.
3. The engineer goes to the nearest printer to pickup the printed designs. Meta data from the DMS (such as file name, last modified date and status) are included as part of the header of the printout.

3.2 AutoVue ISDK Use Cases

The following examples show how an AutoVue ISDK solution can be used for reviewing, annotating and approving designs.

Note: For sample code on how to implement the AutoVue ISDK to implement the following use cases, refer to the *AutoVue ISDK Technical Guide*.

3.2.1 Reviewing and Annotating Documents

A wireless technology company uses PTC WindChill and Agile PLM to create and store their phone and wireless-related designs. They also use AutoVue to collaborate on these designs, which are mainly PTC Creo files.

3.2.1.1 Requirements

The company requires a solution to streamline their reviewing process.

3.2.1.2 Integrator steps

1. Create an integration with PTC WindChill using AutoVue ISDK.
2. Create a VueAction called **Create Report** and modify the GUI file to add the new **Create Report** VueAction to the menu. This VueAction launches a report of all the annotations. The report is a simple Web application with one page. In this report, you can use the AutoVue API to get a list of markup files.
3. From the Java application, use the extract feature of the VueBean API to retrieve information about each markup entity and the bounding box of where the markup entity is located on the page.
4. Feed the markup entity location information to the export feature of the VueBean API to create a thumbnail of the page.
5. List all of these thumbnails in a frame on the left hand side and embed AutoVue in the right frame of this page to show the full design.

Note: Using the ABV framework you can add JavaScript code to enable the user to click on a thumbnail in order to highlight the relevant portion of the document.

3.2.1.3 End-user steps

1. The engineer, who has just completed a design for a new technology for a smart phone, sends an email to all his peers that includes a link to the AutoVue server where the file can be opened. Each reviewer clicks on a link in their email and goes directly to AutoVue where they can view the file and add their annotations.
2. Once the reviews are complete, the engineer's manager opens the design from the AutoVue server and clicks on **Create Report** to view the annotations.
3. The manager uses the report to go through all the annotations and clicks on each thumbnail to view a larger image of the design on the right hand frame within AutoVue.

3.2.2 Reviewing and Approving Documents

A federal government receives approximately 1000 permit requests per day. Typically users go to a federal portal and submit their applications. Through this process the applicants also upload the related documents. Although a number of different formats are accepted, the majority of time the documents are in PDF format. Documents can be single page or multi-page and a document can be up to a 1 GB file size. A coordinator identifies the departments to review the documents (typically there are 8 to 10 reviewers for each application). The files are reviewed using AutoVue.

Once the submissions are reviewed, they are routed to an engineer who makes the final call about approving them. When the review process is finished and the design is approved, the coordinator adds approval stamps on each page and converts all associated documents with the stamps to a PDF/A for archival purposes.

3.2.2.1 Requirements

The submissions coordinator is looking to streamline their reviewing and approval process by integrating a case management application system with AutoVue.

3.2.2.2 Integrator steps

1. Create an integration to the repository using the AutoVue ISDK, and integrate AutoVue with a case management application system to create a complete visual application. You can also included a button in the online portal to launch AutoVue and pass the document ID to it so that AutoVue can open the file.
2. Create a VueAction called **Approve** which prompts the user to pick a stamp. The GUI is also modified to add Approve to the menu and toolbar.
3. Use the Oracle Enterprise Visual Applications (OEVA) framework to tie all the annotations to a workflow ID. Doing so helps end users view annotations in a specific sequence.
4. Use the stamping feature of the VueBean API to add a stamp on each page of the document and to save it to the file.
5. Use the conversion feature of the VueBean API to convert the document to PDF with the file name in the footer of the document, and a *Final* watermark on the output file.

> **Note:** AutoVue does not support conversion of 3D pages or documents.

3.2.2.3 End-user steps

1. A permit application for a new construction is submitted to the local government through the ePermitting portal.

2. A coordinator from the government receives notification that a new application has been submitted.
3. The coordinator identifies ten people to review the submitted documents.
4. Reviewers use AutoVue to review the documents and add annotations using the AutoVue UI. All annotations are tied to the given case. The approver reviews all the annotations, creates one consolidated markup file for all the annotations, and then uses the **Approve** menu item to approve the documents.
5. Once the user selects **Approve**, a dialog appears prompting the user to select a stamp and then click **OK**.

3.3 AutoVue Web Services Use Cases

The following examples show how AutoVue WS can be used to manage work orders and to identify faulty/recalled products.

Note: For sample code on how to use AutoVue WS to implement the following use cases, refer to "Appendix A: Sample Client Code in Java" in the *Oracle AutoVue Web Services Installation and Developer's Guide*.

3.3.1 Managing Work Orders

Firms in the asset intensive industries store work order information in a DMS. When there is a repair that needs to take place for a facility, a field worker needs to be sent onsite to fix the issue. The field worker needs to have a print-out of a work order and all of its related documents when performing maintenance onsite.

A maintenance planner reviews all work orders that need to be completed in a day and prepares work order packets for each assigned field worker. Additionally, the maintenance planner reviews all work orders in the system and selects a list of work orders that need to be completed for a given day, and then prints them with their selected supporting documents.

3.3.1.1 Requirements

To design a system where work orders can be viewed, updated and printed anywhere.

3.3.1.2 Integrator steps

1. Customize an Application Lifecycle Management (ALM) system to display a page for the user to choose the attachments and print options. All the printers have been pre-registered on the server. Upon launching of this page, make a call to AutoVue WS to get a list of all printers and display this list to the user. Once the user picks a printer, another call to AutoVue WS is made to get a list of all available print options and display then to the user. These options include print orientation, page range, number of copies and paper size.
2. Make a call to the ALM system to receive a list of all attachments linked to the work order and display the list to the user. The user selects one or more attachments from the list.
3. Include meta data (such as file name) in the header and work order ID in the footer of the documents to identify the files. The *Confidential* watermark is added to all documents that are tagged as such in the ALM system.
4. When the **Print** action is selected, a call is made to the packet printing feature of the VueBean API in order to print all selected documents as a packet. The cover page, summary page, and page number features of the packet printing APIs are included to make the packet contents easily identifiable to the user.

3.3.1.3 End-user steps

1. When performing a search on all Work Orders in the ALM system, a maintenance planner reviews all work orders in the ALM system and chooses a list of work orders that need to be completed for that day.
2. The maintenance planner then selects all these work orders and clicks **Print**.
3. The maintenance planner then goes to the printer to pick up the work orders and their supporting documents and collates them so that each field worker can have a set of work order packets for that day.
4. The maintenance planner gives the work order packets to the field workers who perform the necessary operations.

3.3.2 Identifying Recalled Components

A company relies heavily on information contained in their repository of millions of documents which have critical information about the design of their product. Employees from different departments require access to these documents in order to search for information on parts and designs.

The engineering department needs to have access to this information to locate parts and determine whether a part can be re-used. Customer support and marketing departments need to have access to product details in order to better market and support their customers. If there are recalls of certain parts of a product, it is necessary to search the product catalog to identify products containing recalled parts.

The company uses a search application along with AutoVue Text Extraction API's to retrieve text from all the different files types in their multiple repositories. The search application provides indexing and searching capability, but cannot read technical documents and relies on AutoVue to extract text from technical documents.

3.3.2.1 Requirements

Use AutoVue's text search capabilities to build an indexing and searching capability across all the technical documents in the organization.

3.3.2.2 Integrator steps

1. Use text extraction capability from AutoVue WS to retrieve text from documents and store the text in a DMS for search purposes. Text needs to be extracted from all CAD documents.

3.3.2.3 End-user steps

1. The administrator in charge of recalls needs to ensure that any product which has been recalled for safety reasons is accounted for. Additionally, any other products that may use the recalled part should also be recalled. Having access to all the information in a document is critical as it may have an impact on the public’s safety.
2. The administrator logs into the search application to look for a specific part number to see how many designs use that part number.
3. The search application has already searched all the files in the company’s DMS and indexed all text in those documents using AutoVue text extraction APIs.
4. The search application browses through its data store to find the searched keywords. The user is given a list of files that match the searched keywords.
5. The products that contain the problematic parts are identified and the appropriate action is taken to ensure those parts are also recalled.

3.4 ABV Use Cases

The following example discusses how an ABV solution can be used for highlighting parts based on availability as well as regional, 3D, and text hotspots.

Note: For sample code on how to implement ABV and hotspots, refer to the "Hotspot Samples" section of the *Oracle AutoVue API and ABV Developer's Guide*.

3.4.1 Material Availability Using Hotspots

A construction company builds large off-shore plants. They work with sub-contractors who plan and perform the actual construction tasks. However, the construction company is responsible for assigning high-level tasks on a weekly basis and to ensure that all the materials and information (documentation, and so on) are available for the tasks. The overall project is divided into multiple areas, and each area has a construction manager.

The onsite construction manager performs a constructability analysis on a week-by-week basis to ensure that all the materials and documentation are available for the work planned for that period. If the manager notices that some of the materials/documentation for the work planned during the new few weeks are not available, then they must identify other tasks that the subcontractor can perform until the missing materials/documentation are provided.

A main area of interest of the construction manager is the construction of pipelines in the plant. Each pipeline makes a connection between two pieces of equipment and is broken down into a series of smaller pipeline segments—the split can be done on various criteria such as length or when the pipeline passes through different areas of the plant. The 3D model of the pipeline is then broken down into the pipeline segments. For construction purposes, each pipeline segment is further broken down into spools, which are components that are individually assembled and welded into place.

The construction company uses a material management system to provide current availability information of components in a spool and to provide summary information indicating which of the pipeline segments and pipelines have all the materials available. For pipelines that are not completely available, the management system can provide a list of pipeline segments or spools that are missing materials.

3.4.1.1 Requirements

The construction manager would like to see each pipeline highlighted based on availability down the spool—where each spool will be colored in green (fully available), yellow (partly available), or red (not available). To view textual information for each spool, the construction manager can then click on a highlighted pipeline segment to retrieve a list of which spools are available/unavailable.

3.4.1.2 End-user steps

1. The construction manager (CM) brings up a view of the area of interest of the plant in AutoVue.
2. The CM requests a view of overall availability, and observes that pipelines 1, 4 and 5 show up fully in green, pipelines 2 and 3 show up partially in yellow, and pipelines 6 and 7 show up mostly in red. This indicates that 1, 4 and 5 are fully available, pipelines 2 and 3 are partially available, and pipelines 6 and 7 are not available Note that a request could have also been made for material availability, or documentation availability with similar results.
3. The CM consults the Primavera schedule and determines that in the next two weeks the subcontractor should be working on pipelines 1, 2, 3 and 4. Since pipelines 2 and 3 are not

fully available, the construction manager needs to identify an alternative plan.

4. In AutoVue, the CM investigates further. Since pipeline 2 has relatively few yellow ISOs, it is handled first. By clicking on the yellow pipeline segments in pipeline 2, a side panel is updated to show the list of available spools in the pipeline segment. With this information, the CM is able to determine that each of the yellow pipeline segments in pipeline 2 have enough material available to start construction and should not delay the work.
5. The CM looks at pipeline 3. It has more yellow pipeline segments, and by clicking on each pipeline segment it is determined that it is not safe to start construction on pipeline 3 as it would likely involve a delay to the project.
6. Since pipeline 5 is displayed fully in green, the CM looks at the schedule and sees that pipeline 5 is not scheduled to be started for another 3 weeks, even though all of the materials are available today. As a result, the schedule is adjusted to reflect that pipeline 5 will be started this week, and pipeline 3 will be started in 3 weeks since the materials for pipeline 3 are excepted to arrive by then.
7. The CM creates the work packages for the subcontractor to start work on pipelines 1, 2, 4 and 5.

3.4.2 Property Management Using Hotspots

A property management firm manages shopping malls and charges store owners a flat rent plus a percentage of revenue. As a result, it is in the management firms best interest to keep only customers that are running successful businesses.

The facility management software they use categorizes tenants into four categories:

- RT/RL (right tenant/right location)
- RT/WL (right tenant/wrong location)
- WT/RU (wrong tenant/right usage)
- WT/WL (wrong tenant/wrong location)

It also identifies shops where some action is expected in the next twelve months. In their sample, they indicate these rooms by highlighting them.

All of these operations happen on PDF floor plans of the mall.

3.4.2.1 Requirements

The management firm would like to be able to show a floor plan of the mall with each store highlighted in different colors based on the categorization, and with an easy way to identify stores where an action will take place in the next twelve months. Additionally, they would like an easy way to see various information about the store (current tenant, lease info - start/end, size in square meters, etc.), and to trigger various actions in their facilities management system. For example, to start the process to cancel a lease, or to transfer a tenant to another location.

3.4.2.2 Integrator steps

1. Create an integration between the facilities management system and AutoVue's hotspots.
2. Build a custom tool which stores the hotspot information and the Doc ID's in an XML document.
3. Build a hotspot integration using Ajax that reads in this XML document and creates the hotspot definitions for this file when it is opened.
4. Implement hotspots to make calls to the DMS upon triggering actions in a drawing.

3.4.2.3 End-user steps

1. For each floor of each building managed by the property firm, an administrator creates a mapping between the store locations and the store names. To do this, the administrator does the following:
   - Opens the floor plan PDF in a user-defined AutoVue hotspotting administration tool and clicks on the polygon representing the store. The shape is highlighted and the administrator is prompted to enter the name of the hotspot.
   - The administrator enters the store name and clicks **OK**.
   - When all the hotspots are verified, the administrator clicks the **Save** and an XML file is generated containing the file name, the doc ID, and a list of hotspots.

   Note: Each hotspot entry contains the name and polygon points of the hotspot shape. The XML file is saved with the floorplan in the DMS system.

2. A facility manager selects a floor of a building. The floor plan is retrieved, and the facilities management system retrieves the hotspot information for the floor plan and passes it to AutoVue.
3. The facility manager sees the floor plan with the color-coded rooms based on the categorization and where an action will happen in the next twelve months.
4. The facility manager moves the mouse cursor over each store to see a tooltip indicating the tenant name, lease start/end dates, and area of the store.
5. The facility manager initiates various actions in the facility management system by clicking or right-clicking on each store. For example, the actions may initiate a cancellation of a lease or the relocation of a tenant.
6. The facility manager shares the current status of the floor with colleagues by converting the document to PDF, and then emails it to colleagues.

   Note: AutoVue does not support conversion of 3D pages or documents.

3.4.3 Tracking Delivery and Stock Quantities of Product Parts Using 3D Hotspots

A company that constructs energy-oriented facilities wants to connect their 3D models to data (such as order status and delivery date) located in a DMS. Currently, they have a mapping table that maps the items in the CAD model to the data in the DMS. However, the mapping table has been known to fail when the model is updated (Unique IDs in the CAD file change, for instance). As a result, a more reliable solution is needed.

3.4.3.1 Requirements

The company requires a visual representation of stock quantities for specific parts. Some information such as order status, delivery date and so on should be displayed as tooltips or in a side-panel. Other information can be displayed as a visual dashboard by coloring the parts appropriately.

3.4.3.2 Integrator steps

1. Use the AutoVue ISDK to create an integration between the DMS and AutoVue hotspots.

2. Configure a mapping between key attributes in the model and data in DMS. This mapping is stored as an XML file in a location that is accessible.
3. Leverage ABV to read the XML file using Ajax and create the appropriate hotspots.
4. Use ABV to handle the action and to make calls to the enterprise system to retrieve part data, initiate payment, and so on.

> **Note:** It is often required to display visual information on a large number of parts. There may be cases when the company wants to apply hotspots to higher-level assemblies, and then over-ride the hotspots on sub-assemblies or individual parts. In this situation, the hotspot applied at the lowest level of the structure should be used.

3.4.3.3 End-user steps

1. The user selects the plant to view. The integrator sets up the mapping between the parts in that model and the DMS.
2. The user is presented with a dashboard which shows untargeted stock as green, targeted stock as orange, and unordered items as red.
3. The user can see various part information (such as cost, order status, and delivery date) by hovering the mouse cursor over hotspotted parts.
4. Even though the parts are highlighted based on the stocking status, when the hotspot is clicked, the integration knows which part has been clicked and accesses its other attributes.
5. When a part is received on-site, the user can click on the part and initiate payment for the part, or initiate a return if the part is damaged.

3.4.4 Plant Maintenance Using Text Hotspots

A company in the asset-intensive industry uses AutoVue and SAP Plant Maintenance (PM) to plan maintenance activities.

The maintenance planner is an individual who typically manages maintenance activities, handles inventories, orders parts and is responsible for maximizing plant uptime.

The maintenance planner knows that there are some assets with faults in the east wing of the plant. For each asset, work orders are to be created for the work that needs to be done and a decision made on whether the asset should be repaired or replaced.

3.4.4.1 Requirements

Integrate the SAP PM with AutoVue using ABV in order to streamline work order creation, updating and processing.

3.4.4.2 Integrator steps

1. Use AutoVue ISDK to create an integration between SAP PM and AutoVue.
2. Implement ABV and use regular expressions to parse the names of specific model parts and turn them into hotspots. Then make a call to the SAP PM to retrieve the failure history for that part.
3. Implement ABV and implement code to handle left-clicking on the hotspots. By making a call to the SAP PM application to retrieve information when a hotspot is clicked, you can display the failure history in the left frame of the window.

4. Use ABV to create the right-click menu and define the actions on it. The **Add to New Work Order** action is tied to a method which keeps track of all the parts.
5. The ABV implementation stores the part information and displays it on the left frame.
6. Use the same right-click menu and added a new action **Check Inventory**, which is tied to a method which makes a call to the SAP PM to check for inventory given the part number. Using the ABV framework, color code the parts based on the outcome of this method.
7. Add another action to **Create Work Order**. This action is tied to a method which makes a call to the SAP PM API's to create a new work order and populates the basic information, including the part numbers.

3.4.4.3 End-user steps

1. The maintenance planner logs into the SAP PM application and opens a drawing of the east wing of the plant in AutoVue.
2. The maintenance planner clicks a button, and AutoVue (specifically, ABV) colors valves 1, 2, and 3 red, showing that there are faults associated with these assets.
3. The maintenance planner clicks on one of the valve 3 that is colored red, and is presented with some detailed information about the failures (such as the dates the valve failed and the failure codes). Based on this information, the planner decides that the valve should be repaired instead of being replaced. The repair involves replacing some parts in the valve.
4. Next, the planner opens up the detailed drawing of the valve, which shows all the parts that make up the valve. From the drawing, the planner can see that three parts need replacement: 7, 8, and 9. The planner right clicks on part 7. From the menu displayed, selects **Add to New Work Order**. The part is highlighted and in the left hand side of the window, 7 is listed in a list of all the parts to be added to the new work order.
5. The planner repeats this process for parts 8 and 9. The list on the left hand side shows 7, 8, and 9.
6. The planner right-clicks one of the parts and selects **Check Inventory**. The three parts 7, 8, and 9 all turn green, showing they have the parts in inventory. If a part turns red, it indicates it is not in inventory, and the planner can order it through the company's procurement system.
7. The planner right-clicks and selects **Create Work Order**. The work order creation dialog pops up in SAP PM system, pre-populated with some basic information about the pump, and with 7, 8, and 9 already added to the list of Required Materials.
8. Through the SAP PM interface, the planner adds a drawing of the valve to the list of work order attachments.
9. The planner opens up the attached plant drawing in AutoVue, and creates a markup to indicate which parts of the valve should be replaced. Valves 7, 8, and 9 are highlighted and zoomed to on the drawing and text informing the technician the type of problems this valve has had over the last year.

4

FAQ

The following sections provide answers to frequently asked questions regarding AutoVue.

- When to use AutoVue API vs. Web Services?
  - Our Web Services API provides a subset of AutoVue's capabilities. It is intended for applications that do not require too many customizations or are implemented in different languages. For example, when performing a background conversion of document to PDF, it is recommended to use the Web Services API.

    If the application is implemented using Java, then the AutoVue API can be used with a much richer set of capabilities. For example, when printing documents that are stored in a DMS, it is recommended to use the AutoVue APIs.
- When to use VueBean vs. JVueApp?
  - The JVueApp class should be used when using or customizing the AutoVue GUI. It can be done through another Java application. For example, the JVueApp class can be used to create a new custom action. The action can be invoked from the menu, toolbar, or right-click menu as specified in the GUI file.
    The VueBean should be used to leverage AutoVue's functionality without making use of its existing GUI layer. For example, a custom application may be written to use the VueBean and leverage only its rendering or conversion capabilities.
- How to integrate with a DMS if using Desktop Deployment?
  - This is currently not supported.
- Difference between batch print and packet print?
  - Batch printing is when multiple unrelated documents are sent to the printer together at one time from one command. For example, a script is created to print all documents in a file system folder.

    A packet print is a collection of related documents with the following characteristics:

    * Each page of each document has been marked with a packet identifier.
    * A cover page is included that lists the packet identifier and all the documents included in that packet.
    * A summary page outlines what documents are printed (along with page numbers) and their status (Success/Failure).

    For example, a work order that contains CAD drawings, specifications sheets, instructions, and so on is considered a packet print since all the documents are related.
- How to convert a non-3D document to grayscale
  - AutoVue does not support document conversion to grayscale. However, black and white conversion can be performed through the force-to-black (monochrome) flag.

The following AutoVue API example shows how to enable force-to-black when converting a file to PDF.

```
public void convertFile() {
  VectorConvertOptions opts = new VectorConvertOptions();

  opts.setStepsPerInch(1);
  PAN_CtlFileInfo fi = m_vueBean.getFileInfo();
  PAN_CtlRange ps = m_vueBean.getPageSizeEx();

  if (fi.getType() == fi.PAN_DocumentFile) {
    ps = fi.getPageSize();
  }

  opts.setInputRange(ps);
  opts.setArea(ConvertOptions2D.AREA_EXTENTS);
  opts.setScaleFactor(100);
  opts.setScaleType(ConvertOptions2D.TYPE_SCALE);
  opts.setUnits(Constants.UNITS_INCH);
  opts.setPages(ConvertOptions2D.PAGES_RANGE);
  opts.setFromPage(1);
  opts.setToPage(1);
  opts.enableMode(PAN.CTLMODE_MONOCHROME); // Apply force-to-black
  opts.setFormat("PCVC_PDF");
  opts.setSubFormatID(0);
  opts.setFileName("C:\\temp\\out.pdf");

  Property[] p = m_vueBean.uploadMarkups();
  opts.setProperties(p);
  m_vueBean.convert(opts);
}
```

- How to customize markup tooltips to display only the description?
  - This is currently not supported.
- How to make the server generate a streaming file programmatically?
  - If the server is configured to generate streaming files, the simplest way is to use the AutoVue API to open (`VueBean.setFile()`), and then close the document (`VueBean.closeFile()`) for which you want to create a streaming file. This schedules the creation of a streaming file.

    The streaming file can be created on demand by calling `VueBeanController.saveMetafile()` while the file is still loaded (between the `setFile()` and `closeFile()`). It can also be retrieved via `VueBeanController.getMetafile()`.
- How to change the color of an existing markup entity using AutoVue APIs?

  Note: The markup must be selected for the following AutoVue API code snippet to work.

```
MarkupBean markupBean = vueBean.getMarkupBean();
MarkupEntity ent = markupBean.selectionGetEntity();
MarkupEntitySpec spec = markupBean.getMarkupEntitySpec(ent);
spec.setColor(lineColor);
markupBean.selectionSetSpec(spec);
```

- How to hide a certain markup type in the UI

- Use a custom GUI file and remove the Markup actions corresponding to the type to hide. For example, if you want to remove Text markups from the UI, copy the default.gui file to create a new GUI file. In the custom GUI file, remove all entries with `MrkActionText`.

- What format types are supported for stamps?
  - AutoVue supports adding Windows Metafile (WMF), Enhanced Metafile (EMF), and Bitmap (BMP) files as background images for your Stamp. It is recommended to use EMF as the background image for Stamps.

- What formats are supported in conversion using AutoVue APIs vs. Web Services
  - For AutoVue API, the following formats are supported:
    * PDF
    * TIFF (Uncompressed, PackBits, Fax III and Fax IV)
    * Windows Bitmap

    The AutoVue API does not explicitly support conversion to JPEG or PNG, but this can be done in conjunction with other Java libraries. An example of the version to JPEG is included in the *AutoVue API and ABV Developer's Guide*.
  - For Web Services, the following formats are supported:
    * JPEG
    * PNG
    * PDF
    * Windows Bitmap
    * TIFF

- How do you close an AutoVue application cleanly programmatically?
  - To cleanly close AutoVue programmatically, using the AutoVue API directly, call the following VueBean API methods:
    * `vueBean.closeFile()`
    * `vueBean.destroy()`
    * `vueBean.getServerControl().sessionClose()`

    Then call the `com.cimmetry.JVueApp` class' `destroy()` method.

- How to give a name to a new markup programmatically?
  - To give a name to a new markup programmatically, call the `com.cimmetry.markupbean.Markup` class' `setName()` method and call `MarkupBean.notifyMarkupChanged()`.

    For example:

    ```
    Markup mrk = getMarkupBean().getActiveMarkup();
    mrk.setName(markupName);
    ```

- How can I get technical assistance with API/customization questions?
  - Consult the product documentation, JavaDocs, and My Oracle Support knowledge base as primary resources answering technical questions and issues.
  - For basic questions regarding the purpose or usage of specific API methods, a Service request can be logged to the Oracle Support team via the My Oracle Support portal.

Note: Support's scope does not include customizations, application development to include creating code/examples, or debugging custom code, demo code, or sample code.

– For more complex technical questions regarding design, development or debugging, it is recommended to either post in the AutoVue Integrations Forum or reach out to the Oracle Consulting Services (OCS) or one of the Oracle AutoVue Development partners.

A

Feedback

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

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

A.1 General AutoVue Information

Web Site	https://www.oracle.com/applications/autovue/

A.2 Oracle Customer Support

Web Site	https://www.oracle.com/support/

A.3 My Oracle Support AutoVue Community

Web Site	https://community.oracle.com/hub/

A.4 Sales Inquiries

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