Oracle® Augmented Business Visualization

Developer's Guide

Release 20.2.1

September 2012

ORACLE®

Oracle ABV Developer's Guide, Release 20.2.1

Contents

5 Hotspot Samples

6 VueAction Sample

7 ABV Design and Security Recommendations

8 Feedback

Preface

An Augmented Business Visualization (ABV) solution connects portion of documents to business data found in enterprise applications.

Audience

This document is intended for system integrators and developers looking to develop ABV solutions which enable users to access ERP data via hotspot links from a document.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program Web site 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:

- *AutoVue API Programmer's Guide*
- *VueBean Javadocs*
- *Oracle AutoVue Installation and Configuration Guide*
- *AutoVue Planning Guide*
- *AutoVue Integration SDK Overview*
- *Oracle AutoVue Web Services Overview*

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's Augmented Business Visualization (ABV) solution creates bi-directional links between parts of a document and Enterprise Resource Planning (ERP) systems. This is done by integrating AutoVue with an enterprise application to query the ERP system through actionable hotspots or by highlighting assets. These hotspots can be used to trigger actions or business processes in the ERP from the documents. For example, a work order can be triggered by clicking on an asset in the drawing. You can also create a visual dashboard by highlighting hotspots in your document based on its ERP data. For example, all assets on a drawing that have had failures in the past 12 months can be highlighted. Note that hotspots can be implemented as text-based, regional (box or polygon), or 3D hotspots and can be page specific.

This document provides an overview of the ABV architecture, ABV use cases, and guidelines on how to create visual dashboards and visual actions using ABV's hotspot capability.

2

Architecture of ABV Solution

The ABV framework can be integrated with a number of enterprise systems. ABV interprets visual, non-visual, structured, and unstructured data. The following diagram details the workflow of an ABV solution:

Figure 2–1   ABV integration workflow
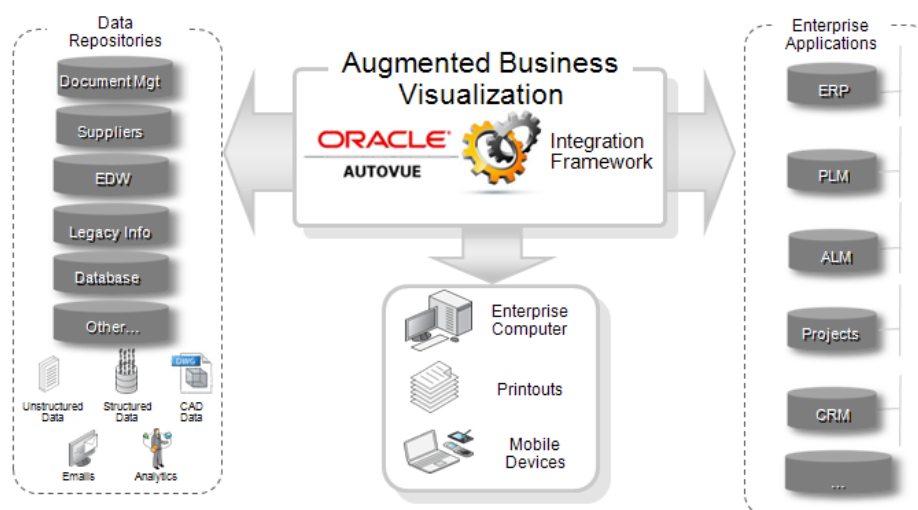
Through the use of AutoVue hotspot capabilities, the ABV integration framework ties together all the individual components that make up the solution (data repository, AutoVue, and enterprise applications). The ABV integration coordinates with the backend systems and can be written in either Java or JavaScript.

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 2–2   Visual action architecture
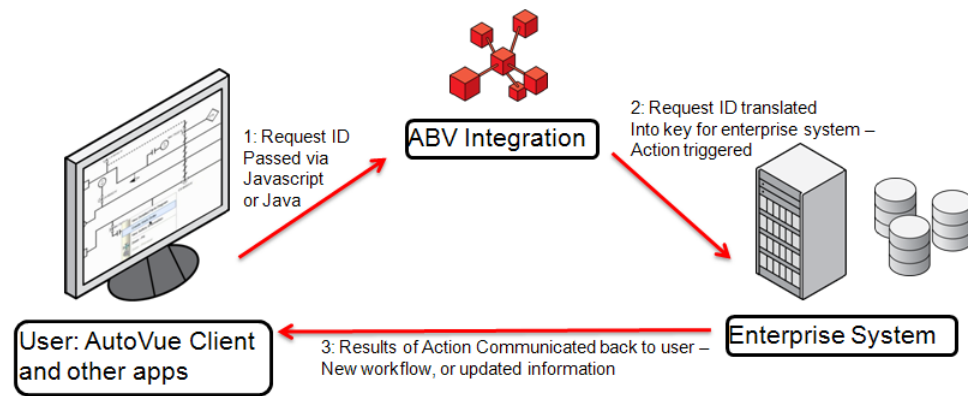
For visual actions, information is retrieved from the data repository when a user clicks on a text hotspot, regional hotspot, or 3D hotspot. A request identifier is passed to the ABV integration by the user. The ABV integration then translates the request ID into a key for the enterprise system and an action is triggered. The results of the action is communicated back to the user from the enterprise system.

Figure 2–3   Visual dashboard architecture
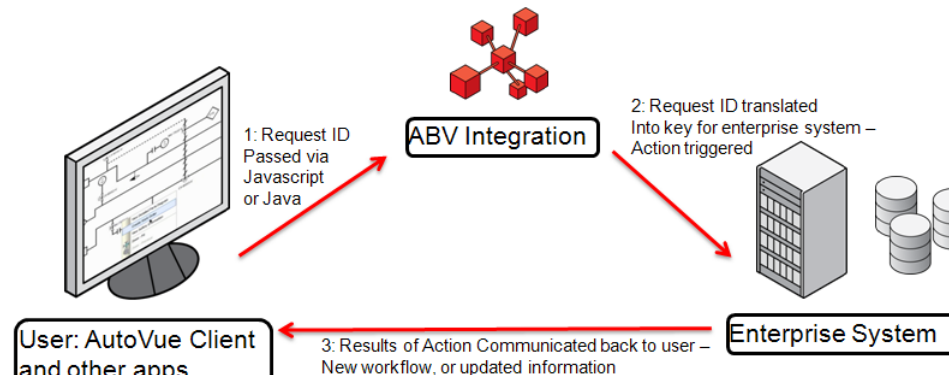
For visual dashboards, the ABV integration retrieves information for the dashboard from the enterprise system. This information is then passed to the hotspot identifiers and colors via json or xml.

For more information on hotspot configurations, refer to Chapter 3, "Hotspots". The following section provides a ABV example.

ABV Example

The following example discusses how an ABV solution can be used when constructing large facilities.

For more information on how to create a visual action and/or visual dashboard, refer to Chapter 3, "Hotspots".

Constructability Analysis

A construction company builds large on-shore plants. They work with sub-contractors who plan and perform the actual constructions 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, etc.) 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 available for the work planned for that time. 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 work that the subcontractor can perform until the 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 ISOs—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 line is then broken down into segments based on the ISO. For construction purposes, each ISO 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 ISOs and pipelines have all the materials available. For pipeline that are not completely available, the managements system can provide a list of ISOs or spools that are missing materials.

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 ISO to retrieve a list of which spools are available/unavailable.

Steps

1. The construction manager performs the constructability analysis for the next two weeks. He brings up a view of the area of plant he is responsible for in AutoVue.

2. The construction manager requests a view of overall availability, and he sees lines 1, 4 and 5 show up fully in green, lines 2 and 3 show up partially in yellow, and lines 6 and 7 show up mostly in red - indicating that 1, 4 and 5 are fully available, lines 2 and 3 are partially available, and lines 6 and 7 are not available (he could also have requested material availability, or documentation availability with similar results).

3. The construction manager starts Primavera, and consults the Primavera schedule. He determines that in the next two weeks the subcontractor should be working on lines 1, 2, 3 and 4. Since lines 2 and 3 are not fully available, he needs to identify an alternative plan.

4. In AutoVue, the manager investigates further. Line 2 has relatively few yellow ISOs, so he looks at Line 2 first. By clicking on the yellow ISOs in line 2 a side panel is updated to show the list of available spools in the ISO. With this information, he is able to determine that each of the yellow ISOs in line 2 have enough material available to start construction, and should not delay the work.

5. The construction manager looks at line 3. It has more yellow ISOs, and by clicking on each ISO he determines that it is not safe to start construction on line 3 as it would likely involve a delay to the project.

6. Since line 5 is displayed fully in green, he looks at the schedule and sees that line 5 is not scheduled to be started for another 3 weeks, even though all of the materials are available today. As a result, he adjusts the schedule to reflect that line 5 will be started this week, and line 3 will be started in 3 weeks since he expects the materials for line 3 will be available by then.

7. The construction manager creates the work packages for the subcontractor to start work on lines 1, 2, 4 and 5.

3

Hotspots

AutoVue'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:

- Hotspot definitions.

- Visual Dashboard: A visual representation of data connection information defining AutoVue Hotspots linked to external objects. Refer to "Creating a Visual Dashboard".

- Tooltip to display on the hotspots defined in AutoVue. Refer to "Tooltips".

- Customization for hotspot selection notification.

- Customization of hotspot color.

- Visual Actions: Customization for available actions for selected hotspots. Refer to "Creating a Visual Action".

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

- Text-based hotspots in 2D, EDA, and PDF documents (based on patterns of text in the document). Refer to "Text Hotspots in 2D and EDA Documents".

- Regional (box and polygon) hotspots in 2D, EDA, PDF and graphic documents are defined by world-coordinates on the page. Refer to "Regional Hotspots".

- Hotspots based on 3D attribute names and/or values. Refer to "3D Hotspots".

- Hotspots in Web CGM files which can be defined by the Name, ID, or URI parameters. Refer to "Web CGM Hotspots".

> **Note:** It is possible to extend the AutoVue applet using the `VueAction()` method to implement a hotspot action. Refer to the "Custom VueAction" chapter in the *AutoVue API Programmer's Guide* for a `VueAction()` hotspot example.

Creating a Visual Dashboard

Figure 3–1   Visual dashboard
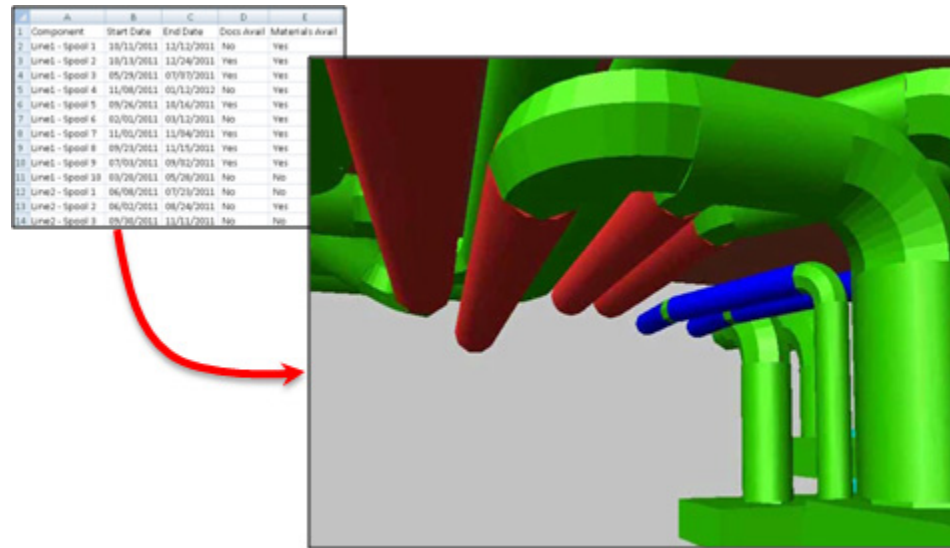
Your enterprise application can highlight hotspots in your document based on its ERP data, creating a visual dashboard. The visual dashboard displays structured data (enterprise application data) on top of a drawing by using color-coded hotspots based on the business data of a document.

Hotspots of all types can be highlighted by providing the hotspot definition key, the appropriate hotspot key, and the desired hotspot color.

To create a visual dashboard, you can highlight various hotspot entities in specific colors:

- Hotspots of all types can be highlighted by providing the hotspot definition key, the appropriate hotspot key, and the desired color.

- Each hotspot entity must be mapped to the appropriate color by the ABV integration.

Creating a Visual Action

A visual action is a hotspot that triggers actions in your enterprise application. These actions can include highlighting an area of the document, zooming into a component, opening a browse dialog, and so on.

Actions can be defined for the following:

- Single-Click

- Double-Click

- Selecting a named action from the RMB menu.

Action handlers can be defined to retrieve the appropriate information as follows:

- Definition key for the hotspot definition used.

- Hotspot key to identify the hotspot element being acted upon.

- The action to perform.

- Any modifiers keys (Shift, Alt, and so on) that are active when the action is started.

Hotspot Features

AutoVue supports the following user interactions with hotspots.

Tooltips

An active hotspot highlights to indicate that it has an action when a mouse curser hovers over it. Additionally, a tooltip appears describing the hotspot's functionality.

In the event there are multiple layers of tooltips (markup, measurement, hotspot, and so on) that are associated to an object in the drawing, only one tooltip appears. Which tooltip appears depends on the tooltip's priority ranking in the stack of tooltips.

> **Note:** The markup tooltip has top priority.

- Markup tooltip

- Measurement tooltip

- Hotspot tooltip

- EDA entity information tooltip

- Hyperlink tooltip

Triggering Actions

When a user clicks on a hotspot, a notification is fired to the ABV integration with the information identifying the clicked hotspot and the mouse action--single-click, double-click, and right mouse button (RMB) action--as well as keyboard modifiers (Ctrl, Shift, Alt).

As with tooltips, when triggering an action the following precedence rules are used:

> **Note:** The markup entity has top priority.

- Markup: Consumes the mouse action.

- Measurement: Consumes the mouse action.

- Hotspot: Notifies the external application but does not consume the mouse action and allows the subsequent layers to the mouse clicks as well.

- Hyperlink: Does not consume the mouse action.

- EDA Entity selection, 3D Entity selection, Entity properties on double-click, and so on.

3D Hotspots

In 3D files, hotspots are defined by the attribute name. Optionally, an attribute value can be defined. If no attribute value is provided, then AutoVue identifies all parts with the attribute name as a hotspot. That is, the attribute value is used by AutoVue as a key to identify the hotspot attached to the owner part.

Figure 3–2   3D Hotspot
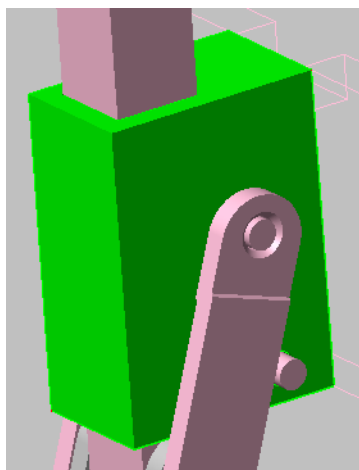
3D hotspots can be used to connect a 3D model to unstructured data such as order status, delivery dates, and so on. By setting up a visual dashboard in the 3D model, all this information can be pulled from an ERP and displayed in real-time.

The following sections describe how to initialize a 3D hotspot and design recommendations.

Defining a 3D Hotspot
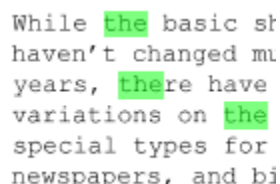
Take the following into consideration when defining 3D hotspots.

- Hotspots are not supported on 3D PMI entities.

- Attribute name or value used in 3D hotspot definitions cannot contain regular expressions. The attribute name/value should not contain any leading or trailing spaces and should exactly match the attribute name/value in the model (case-sensitive).

- AutoVue supports attribute names/values that contain a semi-colon (;). You must precede the semi-colon with a backslash (\).

- Internal attributes that AutoVue displays in 3D models (for example, Mesh Resolution, Transparency, and Layers) should not be used when defining hotspots.

- To prevent conflicts in highlight color, it is recommended to use the Bounding Box Highlight for a 3D selection (default AutoVue setting) instead of the Entity Highlight.

- If a hotspot is defined with density as an attribute, then the specified density value must be the same value saved in the native file without measurement units.

- It is not recommended to define hotspots with attributes that the user can modify after the model loads (for example, Color, Transparency, Display/Render Mode, Visibility, Highlight Color, and Bounding Box Color). If these attributes are used and changed by the user during the file session, then the hotspots may behave inconsistently.

Text Hotspots in 2D and EDA Documents

Text hotspots are supported in 2D and EDA documents. They are based on regular expressions filtering graphical text strings based on AutoVue's text search. You can use regular expressions in the hotspot definition.

Figure 3–3   Text Hotspot

Text hotspot can be used to trigger actions such as Create WorK Order or Open Detailed Parts Diagram from assets in P&ID diagrams. These hotspots can be clicked to retrieve and display asset information such as failure and repair history, working status, and so on.

The following sections describe how to define a text-based hotspot, what types of text and file formats are supported, and design recommendations.

Defining a Text Hotspot

You must use regular expressions in the hotspot definition in order to search for text in the document. Since AutoVue uses the Java library, it relies on Java's regular expression guidelines. For more information, refer to the Java regular expression guidelines at
http://java.sun.com/developer/technicalArticles/releases/1.4regex

Take the following into consideration when defining text hotspots.

- Text hotspot support is not available for raster formats, archive formats, Microsoft Office, Excel, RTF, and Outlook formats.

- Since text hotspots can only be detected on searchable text, text stored in Windows Metafiles (WMF/EMF) cannot be used for hotspotting.

- If there are multiple occurrences of a text, then they are all handled as valid hotspots.

- To be recognized as a text hotspot, characters in a string must share the same baseline. For example, a string with a normal text and a superscript text cannot be recognized as a single hotspot as they have different baselines. Alternately, regional hotspots can be applied.

- For PDF text with large spacing between characters, it is recommended to use the ADVANCEGAP INI option. For more information, refer to "PDF Text Hotspot".

- Strings that include curved text (curved baseline) cannot be used as a text hotspot. Alternately, regional hotspots can be applied to include the curved text.

Regional Hotspots

Regional hotspots can be defined in 2D, EDA, and Raster files. The hotspots can either be drawn as a box or a polygon. The dimensions/extents for the hotspots are based on the coordinates displayed in the AutoVue status bar. Optionally, a user key can be used

by AutoVue as an identifying key for the hotspot. If the user key is not provided, then an empty (string) key is used.

Figure 3–4   Box Hotspot
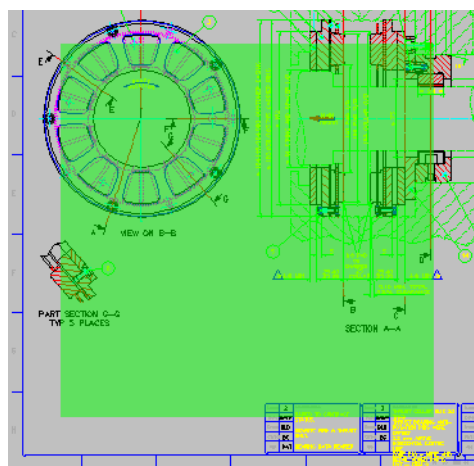
Figure 3–5   Polygon Hotspot
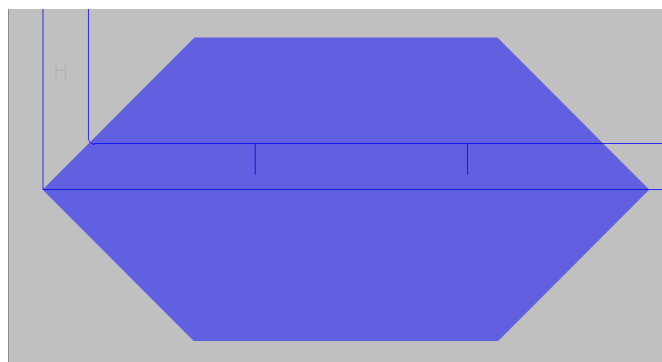
Take note of the following when defining regional hotspots:

- Regional hotspots are not supported for archive formats, Microsoft Word, Microsoft Excel, Microsoft Outlook, RTF formats.

- Vector files and raster files do not use the same World Coordinate System in AutoVue. Vector files use the bottom-left corner of the client area as the origin and the Y-axis oriented down-top, while the raster files use the Top-Left corner as the origin and the Y-axis oriented top-down. This mismatch is already exposed in AutoVue with the current user interface (UI) because the mouse position is reported in World Coordinates System on the Status Bar of the UI. Since regional hotspots are provided relative to World Coordinate System, the regional definitions need to consider this difference between raster and vector files.

Defining Page-Specific Regional Hotspots

When working with multi-page documents, it may be required to define page-specific regional hotspots. For example, a floor plan of interest may be on the second or third page of PDF. As a result, a new parameter allows the administrator to specify the

pages where to apply the define regional hotspot. Refer to Chapter 4, "AutoVue Hotspot API" for information on the DEFINITION_PAGE parameter.

Defining Coordinates of a Box/Polygon

To define the coordinates of a box/polygon in a drawing, you can outline the box/polygon with a markup entity and then dump the coordinates to the regional hotspot definition. The status bar displays the world coordinates of the mouse position. The box/polygon hotspot can be manually defined to use these coordinates. For more information, refer to "Polygon Hotspot" for an example of using a markup entity to create a polygon hotspot.

Defining a Box Hotspot

A box hotspot is defined by minimum and maximum points. Where {X1, Y1} and {X2, Y2} are the coordinates of the box minimum and maximum points, respectively. Refer to Chapter 4, "AutoVue Hotspot API" for information on the DEFINITION_BOX parameter.

Defining a Polygon Hotspot

A polygon hotspot can include an arbitrary number of sides. You can define as many sides as required for a polygon hotspot: (x1, y1), (x2, y2),..., (xn, yn). Where n is the number of sides of the polygon. Refer to Chapter 4, "AutoVue Hotspot API" for information on the DEFINITION_POLYGON parameter.

Invoking performHotspot()

To perform an action on a regional hotspot, the definition key and hotspot key parameters must be defined for performHotspot(). The hotspot key for regional hotspots is the user key. If the user key is not provided then an empty (string) key is used. For more information, refer to "Perform an Action on a Hotspot".

Web CGM Hotspots

In Web CGM files, hotspots are defined in the native file. The hotspot information contains three attributes:

- Name

- ID

- URI

External systems can interact with these hotspots using the VueBean API using a given name. AutoVue matches the name to the ID property of the hotspot. If this fails, AutoVue matches the name to the Name property in order to highlight a specific hotspot. The definition key is always provided by the user (as with all hotspot definitions). The Web CGM hotspots include a hotspot key and definition key and are handled in the same manner as all other hotspots.

4

AutoVue Hotspot API

The AutoVue Application Programming Interface (API) is a Java-based toolset that provides tools to modify the functionality of Oracle's AutoVue client, and allows you to create your own customized Java applets/applications based on AutoVue API components. For more information on the AutoVue API, refer to the *AutoVue API Programmer's Guide*.

The AutoVue API's jVue class includes two methods that handle hotspots:

- `setHotSpotHandler()`: Defines hotspot
- `performHotSpot()`: Performs action on a hotspot

Hotspot INI Options

When working with 2D, EDA, PDF and graphic documents, through the use of the Augmented Business Visualization (ABV) integration framework, you can add AutoVue's hotspot capabilities to create links between objects in AutoVue's data model and objects in an external system.

AutoVue provides the following configuration options for hotspots.

PDF Text Hotspot

Syntax and additional information for the option described here is in section "PDF Text Hotspot INI Options".

AutoVue provides you the ADVANCEGAP INI option to recognize PDF text with large spacing between characters as a single hotspot The value should be less than the maximum number of spaces between consecutive strings. Note that one space is equal to a 1/4 of the height of the font text.

Figure 4–1   Advance gap

For more information on available INI options, refer to the *Oracle AutoVue Viewing Configuration Guide*.

PDF Text Hotspot INI Options

All of the following options should be placed in the [HOTSPOTS] header of the INI file.

Parameter	Description	Default
ADVANCEGAP =[integer]	Specify the maximum number of spaces between consecutive text strings. When there is large spacing between characters, you can increase the value assigned to the INI option.	3

Define Hotspots

```
setHotSpotHandler (final String definitionType, final String
definitionKey, final String Definition)
```

This method sets the hotspot handler for a given hotspot definition. This should typically be called before opening the file. It initializes hotspots in the files opened in AutoVue based on external application data.

Parameter	Description
definitionType	The hotspot definition type. Specify if the hotspot is a WebCGM hotspot, text search hotspot, box/polygon hotspot, or a 3D hotspot.
definitionKey	The hotspot definition key. This is the identifier for the hotspot.
definition	A string separated by semicolons specifying hotspot definition parameters. For example: name1 = value1; name2 = value2.

Hotspot Definition Types

Hotspot definition types supported in *setHotSpotHandler()*:

Parameter	Description
DEFINITION_TYPE_NATIVE	Native Web CGM hotspot.
DEFINITION_TYPE_TEXT	Text search hotspot.
DEFINITION_TYPE_BOX	Box hotspot.
DEFINITION_TYPE_POLYGON	Polygon hotspot.
DEFINITION_TYPE_3D_ATTRIBUTE	3D entity hotspot.

Hotspot Definition Parameters

The following are hotspot definition parameters supported in the key-value string parameter (definition) of the method *setHotSpotHandler()*.

Common Definition Parameters

The following are definition parameters that are common for all hotspots.

Parameters	Description
DEFINITION_TOOLTIP	The tooltip that displays when a mouse point hovers over a hotspot defined by the handler.

Parameters	Description
DEFINITION_ONINIT	The JavaScript method to call when page is loaded and ready to interact. `Syntax: DEFINITION_ONINIT="action1 name, action2 name,... actionN name"`
DEFINITION_FUNCTION	The JavaScript function to call when user performs an action on the hotspot.
DEFINITION_ACTIONS	Popup actions to show when user right-clicks on a hotspot.
DEFINITION_COLOR	The highlight color to use when user hovers the mouse cursor over a hotspot. Note that AutoVue parses the RGBA value as a string. Example: (R, G, B, [A]) Refer to the 3D and Box hotspots examples in Chapter 5, "Hotspot Samples" for more information.

Text Definition Parameters

The following are definition parameters for text hotspots.

Parameters	Description
DEFINITION_REGEX	Regular expression to use only in Text Search Hotspot handlers. For more information, refer to the Java regular expression guidelines at http://java.sun.com/developer/technicalArticles/releases/1.4regex.
DEFINITION_MATCHCASE	Whether to handle case sensitivity in Text Search Hotspot handlers only. `Syntax: DEFINITION_MATCHCASE=[TRUE│FALSE]`

3D Definition Parameters

The following are definition parameters for 3D hotspots

Parameters	Description
DEFINITION_ATTRIB_NAME	The attribute name assigned to a 3D hotspot on the model.
DEFINITION_ATTRIB_VALUE	The attribute value assigned to a 3D entity on the model. (Optional) If this parameter is not specified, then all parts with an attribute of the specified name will be made into a hotspot.
DEFINITION_MATCHCASE	Whether to handle case sensitivity when searching name and value attributes assigned to 3D entities. `Syntax: DEFINITION_MATCHCASE=[TRUE│FALSE]`

Regional Definition Parameters

The following are definition parameters for box and polygon hotspots.

Parameters	Description
DEFINITION_BOX	Define the bounds of the rectangular box given the minimum and maximum points. Where {X1, Y1} and {X2, Y2} are the coordinates of the box minimum and maximum points.
	Note that the points are based on the world-coordinates of the page.

	`Syntax:`
	`DEFINITION_BOX=#X1#Y2#X2#Y2`
	`Example:`
	`_boxDef = "DEFINITION_BOX=#0 #0 #100 #100;`
	`DEFINITION_USER_KEY=box1; DEFINITION_PAGE=1"`

DEFINITION_PAGE	Restricts box and polygon hotspot definitions to the page specified by this parameter. If no page is specified, then the hotspots apply to all pages. The following example defines the hotspot on page 2.
	`Example:` `DEFINITION_PAGE = 2`
DEFINITION_POLYGON	Define the bounds of the polygon as a set of points in the world coordinates in the following format:
	`#(x`$_1$`, y`$_1$`) #(x`$_2$`, y`$_2$`)...#(x`$_n$`, y`$_n$`) where n is the` `number of points.`
	Note that the minimum number of points for a polygon is 3 and that it is treated as a closed polygon (do not have to repeat the final point).

	`Example:`
	`_defPoly = "DEFINITION_POLYGON=#(0,0) #(50, -50)`
	`#(150, -50) #(200, 0) #(150, 50) #(50, 50)" +`
	`"DEFINITION_USER_KEY=box; DEFINITION_PAGE=1"`

DEFINITION_USER_KEY	Define a user key for the box/polygon. This user key allows you to link multiple boxes with various definitions to the same external object. This is the hotspot key used for the hotspot. (Optional)
	If the user key is not defined, then the hotspot key is an empty string.
	`Syntax:` `DEFINITION_USER_KEY=box1`

Perform an Action on a Hotspot

```
performHotSpot (final String definitionKey, final String hotspotKey, final
String action, final String params)
```

Perform a hotspot action on the given hotspot. This method should be called during the file session when the hotspots have been already initialized (only after the external application is notified that hotspots have been initialized in the file).

Parameters	Description
definitionKey	The hotspot definition key (the hotspot identifier) provided at creation.
hotspotKey	The hotspot property key string found based on the definition key.

Parameters	Description
action	The action to perform on the hotspot. Refer to Chapter , "Hotspot Actions".
params	A string separated by semicolons specifying hotspot action parameters. For example: `name1 = value1; name2 = value2.`

Hotspot Actions

The hotspot actions supported in `performHotSpot()` and their arguments are as follows:

Action Name	Description	Arguments
HIGHLIGHT	Perform a highlight action.	HOTSPOT_COLOR: The color for a highlight to add (RGBA Format). If this argument is not provided, the action is interpreted as a Highlight Removal.
ZOOMTO	Zoom to all hotspot instances.	
ZOOMNEXT	Zoom to the next hotspot instance.	
ZOOMPREV	Zoom to the previous hotspot instance.	

AutoVue API for ABV Integration

The ABV integration can call the AutoVue API for manipulating hotspots from the following user actions:

- Highlight (Multiple Selection, Add/Remove)
  - Text Highlight as used in text search.
  - 2D Entity Highlight for Web CGM format.
  - 3D Entity Highlight for 3D formats.
  - Regional Highlight for regional hotspots.
- Zoom to a hotspot, or the hotspots associated with a specific external object.
- Browse the hotspots associated with a specific external object using **Zoom Previous/Zoom Next**.

> **Note:** When a user selects a hotspot, all hotspots associated with the same ABV integration may be selected by using the highlight mechanism provided above.

Interactions with Hotspots from JavaScript

The following is a code prototype for a custom JavaScript function call to initialize hotspots when the file/page loads:

```
initialization_script(String definitionKey)
```

The following is a code prototype for a custom JavaScript function call when a user interacts with hotspots:

```
notification_script(String definitionKey, String hotspotKey, String action, int
keyModifiers, String properties)
```

keyModifiers describes the status of the Shift, Alt and Ctrl keys.

action may be a custom action sent during the definition of the hotspot handler (RMB actions) or one of these two predefined actions:

OnHotSpotClicked	To send when user clicks on the hotspot.
OnHotSpotDoubleClicked	To send when user double-clicks on the hotspot.

properties that could be sent to the external application notification script are as follows:

PROPERTY_ID	Property ID.
PROPERTY_NAME	Name of native WebCGM hotspots.
PROPERTY_URI	URI of native WebCGM hotspots.

5

Hotspot Samples

The following sections provide sample code on how to implement AutoVue's hotspot capability to your enterprise application.

For a detailed ABV integration example, refer Doc ID 1172899.1 in the Oracle Customer Support knowledge base:
https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1472899.1

Adding a Hotspot

The following hotspot example shows how the setHotSpotHandler() and performHotSpot() methods are implemented to add hotspot capability to AutoVue. Note that this example only adds one definition, but it is possible to add multiple definitions.

1. Initialize the hotspots with the ONINIT applet parameter in the applet tag when creating the AutoVue applet. This parameter is used to call the onAppletInit() method after the AutoVue applet has initialized.

   > **Note:** If a newly added definition key already exists, then the existing definition is replaced by the new one.

   ```
   <PARAM NAME="ONINIT" VALUE="onAppletInit();">

   function onAppletInit() {
       var handlerStr = "DEFINITION_REGEX=AutoVue;
       DEFINITION_TOOLTIP=AutoVue 2D Professional";
       // The following function is called once when AutoVue is ready to
       // interact with a hotspot.
       handlerStr += ";DEFINITION_ONINIT=onHotSpotInit";
       // The following function is called each time a hotspot is fired.
       handlerStr += ";DEFINITION_FUNCTION=onHotSpot;
       DEFINITION_ACTIONS=Menu1, Menu2";
       color = "(0,0,255,128)";
       handlerStr += ";DEFINITION_COLOR=" + color;

       //The following call sets up the hotspot definition.

       window.document.applets["JVue"].setHotSpotHandler("DEFINITION
           _TYPE_TEXT", "AV2D", handlerStr);

   }
   ```

2. Method onHotSpotInit() is called for each definition when the current page is loaded and ready for hotspot interactions. Note that the method name should be exactly the same as the one specified in the hotspot definition DEFINITION_ONINIT in step 1.

```
function onHotSpotInit(hotspotDefinitionKey) {
    alert("HotSpot definition initialized: " + hotspotDefinitionKey);
}
```

3. The following onHotSpot() method is invoked when a hotspot is fired when the user either clicks on the hotspot or by selecting one of the Hotspot menu items defined in variable DEFINITION_ACTION in step 1.

```
function onHotSpot(hotspotDefinitionKey, hotspotKey, action, modifiers,
properties) {
 if (equalsIgnoreCase(action, "onHotSpotClicked")) {
        alert("User clicked on hotspot: " + hotspotKey);
 } else if (equalsIgnoreCase(action, "onHotSpotDoubleClicked")) {
        alert("User double clicked on hotspot: " + hotspotKey);
 } else if (equalsIgnoreCase(action, "Menu1")) {
        alert("User Peformed Menu1 action: " + hotspotKey);
 } else if (equalsIgnoreCase(action, "Menu2")) {
        alert("User Peformed Menu2 action: " + hotspotKey);
    }
}
```

> **Note:** The method name should be exactly the same as the one specified in the hotspot definition DEFINITION_FUNCTION in step 2. The onHotSpotClicked and onHotSpotDoubleClicked methods are predefined keys when the user clicks on the hotspot.

4. The following code performs specific actions on the clicked hotspot such as Highlight, Zoom, and so on.

```
// Highlight the "AutoVue" hotspot, "AV2D" is the definition key.
// Color : (R,G,B,A)
params = "HOTSPOT_COLOR=" + (((128 & 0xFF) << 24) | ((255 & 0xFF) << 16) |
((255 & 0xFF) << 8) | ((0 & 0xFF) << 0));
window.document.applets["JVue"].performHotSpot("AV2D", "AutoVue", "Highlight",
params);

// To clear the hotspot highlight simply set the params (color) to null.
window.document.applets["JVue"].performHotSpot("AV2D", "AutoVue", "Highlight",
null);

// To clear the definition highlights, set the hotspot key to null.
window.document.applets["JVue"].performHotSpot("Highlight", "AV2D", null,
null);

// To clear all hotspot highlights, set the definition key to null.
window.document.applets["JVue"].performHotSpot(null, null, "Highlight", null);

// Zoom to the next "AutoVue" hotspot.
window.document.applets["JVue"].performHotSpot("AV2D", "AutoVue", "ZoomNext",
null);

// Zoom to the previous "AutoVue" hotspot.
window.document.applets["JVue"].performHotSpot("AV2D", "AutoVue", "ZoomPrev",
null);
```

3D Hotspot

The following example defines a 3D hotspot.

1. Get the JVue applet

    ```
    jApplet = window.document.applets["JVue"];
    ```

2. Define a 3D hotspot. The following code snippet defines a hotspot matching a part number in a 3D unigraphics assembly file. The sample file is included with the AutoVue Client/Server Deployment installation: <AutoVue Installation Folder>/samples/3D/Unigraphics/3DUnigraphics_iLearn-Assy.prt.

    ```
    //Turn the part with PART_NUMBER = ITEM-UG-00003 into a hotspot. You can leave
    //out the ATTRIB_VALUE if you want to highlight everything with the PART_NUMBER
    //attribute
    item00003Def = "DEFINITION_ATTRIB_NAME=PART_NUMBER; DEFINITION_ATTRIB_
    VALUE=ITEM-UG-00003;"
            + "DEFINITION_TOOLTIP=Board;"
            + "DEFINITION_ONINIT=onHotSpotInit;"
            + "DEFINITION_FUNCTION=onHotSpot;"
            + "DEFINITION_ACTIONS=Add Part, Remove Part;"
            + "DEFINITION_COLOR=(255, 0, 0)";
    ```

3. Set the 3D hotspot handler.

    ```
    jApplet.setHotSpotHandler("DEFINITION_TYPE_3D_ATTRIBUTE", "item00003",
    item00003Def);
    ```

Box Hotspot

The following example details how to define a box hotspot.

1. Get the JVue applet.

    ```
    jApplet = window.document.applets["JVue"];
    ```

2. Define a box hotspot. The following code snippet defines a box hotspot that encloses the Oracle logo from the PDF sample file included with the AutoVue Client/Server Deployment installation: <AutoVue Installation Folder>/samples/Desktop-Office/Basell_Autovue_Case_Study.pdf.

    > **Note:** The box coordinates are defined by #minX #minY #maxX #maxY. Each coordinate must be preceded by a hash (#).

    ```
    oracleDef = "DEFINITION_BOX=#6.4 #0.7 #8.1 #0.4; DEFINITION_USER_KEY=oracle;"
            + "DEFINITION_TOOLTIP=www.oracle.com;"
            + "DEFINITION_ONINIT=onHotSpotInit;"
            + "DEFINITION_FUNCTION=onHotSpot;"
            + "DEFINITION_ACTIONS=Open Link;"
            + "DEFINITION_COLOR=(0, 0, 255, 64)";
    ```

3. Set the box hotspot handler.

    ```
    jApplet.setHotSpotHandler("DEFINITION_TYPE_BOX", "oracleBox", oracleDef);
    ```

Polygon Hotspot

The following code snippet shows how to get a markup entity bounds, convert the bounds to a polygon, and then format the polygon to a polygon hotspot.

```
//Get the markup entity that is to be turned into a hotspot
MarkupEntity ent = getMarkupEntity();
//Get the entity specification
MarkupEntitySpec spec = getMarkupBean().getMarkupEntityFullSpec(ent);
PAN_CtlPolygon poly = null;
//If the markup entity is a polyline, the get the polygon directly from the
//specification
if (spec instanceof PolyLineSpec) {
   poly = ((PolyLineSpec) spec).getPolygon();
} else {
   // Otherwise, get the world bounding rectangle and covert it to a polygon
   PAN_CtlRange rg = ent.getBoundingRect();
   poly = rg.to2DPolygon(null);
}
//Then convert the polygon points into a hotspot polygon
StringBuffer sb = new StringBuffer() ;
for (PAN_CtlPoint pt : poly.points) {
  sb.append("#(" + pt.x + "," + pt.y + ") ");
}
//Create the polygon definition
String defPoly = "DEFINITION_POLYGON=" + sb.toString();
```

Text Hotspot

The example details how to define a text hotspot.

1. Get the JVue Applet.

```
jApplet = window.document.applets["JVue"];
```

2. Define a text hotspot. The following example defines a text hotspot (regular expression) matching the AutoVue string. The PDF sample from "Polygon Hotspot" includes the AutoVue string in multiple locations.

```
autovueDef = "DEFINITION_REGEX=AutoVue; DEFINITION_MATCHCASE=false;"
          + "DEFINITION_TOOLTIP=AutoVue Professional;"
          + "DEFINITION_ONINIT=onHotSpotInit;"
          + "DEFINITION_FUNCTION=onHotSpot;"
          + "DEFINITION_ACTIONS=AutoVue 2D, AutoVue 3D, AutoVue EDA,
             AutoVue Electro-Mechanical;"
          + "DEFINITION_COLOR=(0, 255, 0, 128)";
```

3. Set the text hotspot handler.

```
jApplet.setHotSpotHandler("DEFINITION_TYPE_TEXT", "AutoVue", autovueDef);
```

Text Hotspot with Visual Actions and Visual Dashboard

This example details how to define a text hotspot that utilizes the visual action and visual dashboard features.

1. Define the text hotspot.

```
//Turn strings starting with CV into a Control Valve hotspot
handlerStr  = "DEFINITION_REGEX=CV.*;";
handlerStr += "DEFINITION_MATCHCASE=false;";
handlerStr += "DEFINITION_TOOLTIP=Control Valve;";
handlerStr += "DEFINITION_ONINIT=onHotSpotInit;";
//Actions are handled by JavaScript function OnHotSpot
handlerStr += "DEFINITION_FUNCTION=onHotSpot;";
//When a hotspot is right-clicked, a menu appears with the following options:
```

```
//View Detailed Parts Diagram, Create Work Order and View Safety Information
handlerStr += "DEFINITION_ACTIONS=View Detailed Parts Diagram, Create Work
Order, View Safety Information;";
//Color : (R,G,B,A). When a mouse hovers over a hotspots, //they are
highlighted in 50% transparent blue
color = "(0,0,255,128)";
handlerStr += "DEFINITION_COLOR=" + color;
getJVueApplet().setHotSpotHandler("DEFINITION_TYPE_TEXT", definitionKey,
handlerStr);
```

2. Create the hotspot actions.

```
//Variable hotspotKey contains the identifier for the hotspot entity that
//triggers the action when click.
function onHotSpot(defKey, hotspotKey, action, modifiers, properties) {
//If the hotspot entity is clicked, the side panel updates with information on
//the entity.
if (equals (action, "onHotSpotClicked")) {
    updateSidePanel(defKey, hotspotKey, modifiers);
//Otherwise, if the RMB is clicked, an action can be selected from the menu.
 } else if (equals (action, "Create Work Order")) {
    createWorkOrder(defKey, hotspotKey);
} else if (equals (action, "View Detailed Parts Diagram")) {
    showDetailsPartsPage(defKey, hotspotKey);
} else if (equals (action, "View Safety Information")) {
    showSafetyInfo(defKey, hotspotKey);
}
```

Note that each function must know how to retrieve the appropriate information
and/or to trigger the appropriate actions in the backend systems.

3. Define the highlighted hotspots for the visual dashboard. The ABV integration
identifies which entities need to be highlighted and their specified color. Each
entity is then passed to performHotspot to highlight the entity appropriately.

```
function showHighlights{
  data = getData(); // returns array of JSON objects
  for (i=0 ; i < data.length() ; i++) {
    entity= data[i];
    performHotSpot("Highlight", entity.defKey, entity.hotspotKey,
entity.color);
  }
}
```

3D Hotspot with Visual Actions and Visual Dashboard

This example details how to define a 3D hotspot that utilizes the visual action and
visual dashboard feature.

1. Define the 3D hotspots.

```
//Turn 3D parts with ASSET_ID attribute into hotspots.
handlerStr  = "DEFINITION_ATTRIB_NAME=ASSET_ID;";
handlerStr += "DEFINITION_TOOLTIP=ASSET;";
handlerStr += "DEFINITION_ONINIT=onHotSpotInit;";
//Actions are handled by JavaScript funtion onHotSpot.
handlerStr += "DEFINITION_FUNCTION=onHotSpot;";
//When a hotspot is right-clicked, a menu appears with the following options:
//View Detailed Parts Diagram, Create Work Order and View Safety Information
handlerStr += "DEFINITION_ACTIONS=View Detailed Parts Diagram, Create Work
Order, View Safety Information;";
```

```
//Color : (R,G,B,A). When a mouse hovers over a hotspots, they are highlighted
//in 50% transparent blue
color = "(0,0,255,128)";
handlerStr += ";DEFINITION_COLOR=" + color;
getJVueApplet().setHotSpotHandler(
        "DEFINITION_TYPE_3D_ATTRIBUTE", definitionKey, handlerStr);
```

2. Create the 3D hotspot actions.

```
//Variable hotspotKey contains the identifier for the hotspot entity that
//triggers the action when click.
function onHotSpot(defKey, hotspotKey, action, modifiers, properties) {
//If the hotspot entity is clicked, the side panel updates with information on
//the entity.
if (equals (action, "onHotSpotClicked")) {
    updateSidePanel(defKey, hotspotKey, modifiers);
//Otherwise, if the RMB is clicked, an action can be selected from the menu.
 } else if (equals (action, "Create Work Order")) {
    createWorkOrder(defKey, hotspotKey);
} else if (equals (action, "View Detailed Parts Diagram")) {
    showDetailsPartsPage(defKey, hotspotKey);
} else if (equals (action, "View Safety Information")) {
    showSafetyInfo(defKey, hotspotKey);
}
```

Note that each function must know how to retrieve the appropriate information
and/or to trigger the appropriate actions in the backend systems.

3. Define the highlighted hotspots for the visual dashboard. The ABV integration
    identifies which entities need to be highlighted and their specified color. Each
    entity is then passed to performHotspot to highlight the entity appropriately.

```
function showHighlights{
  data = getData(); // returns array of JSON objects
  for (i=0 ; i < data.length() ; i++) {
    entity= data[i];
    performHotSpot("Highlight", entity.defKey, entity.hotspotKey,
entity.color);
  }
}
```

6

VueAction Sample

The VueAction sample included with the AutoVue installation illustrates how to implement a custom hotspot action in Java. This sample is ready to be tested out of the box, but has limited application as it is not integrated with an enterprise visualization system. It is presented solely as a skeleton framework to show how hostspots can be applied.

The sample includes the following files:

File	Description
PartCatalogueAction.java	This is an example of how to write a custom action for AutoVue. This example illustrates implementation of an action that does more than one thing. It consists of several related sub-actions that access information about parts of a product. This action is added to two components to the AutoVue GUI: AutoVue toolbar buttons and hotspot RMB menu items. AutoVue toolbar buttons: ■ **None**: Disables mouse detection over hotspots. ■ **Description**: Enables mouse detection over hotspots and displays hotspot description as a tooltip. ■ **ID**: Enables mouse detection over hotspots and display hotspot ID as a tooltip. Hotspot RMB menu items: ■ **Order Part**: Displays a dialog that includes part information and a quantity order field. Note that this dialog does not actually retrieve any part information. It is only used to display possible RMB menu actions. ■ **Show Part Information**: Displays part name and ID.
PartListAction.java	This is an example of how to write a custom action for AutoVue. This action performs a single task and is added to the List Product Parts option of the Analysis menu: ■ List Product Parts: Lists the hotspots that user double-clicked.
PartInfo.java	This class provides product part information. It contains the catalog ID, part ID and part description.
VueActionSample.jar	JAR file for the VueAction sample.
JavaDocs	Provides detailed information on the classes included in the sample.

File	Description
custom.gui	Defines the custom user interface of AutoVue. It adds PartCatalogueActions to the AutoVue toolbar and Hotspot RMB menu and the PartListAction to the Analysis menu.
hotspots.txt	Contains the hotspot definitions. For information on how to define hotspots, refer to "Hotspots".
customjvue.bat	Batch file that runs the sample. Note that the file illustrates how to bundle the custom action with the custom user interface.
Basell_AutoVue_Case_Study.pdf	Sample file to be used with the VueAction sample. It is located in <AutoVue Installation>\html\samples\Desktop-Office directory.
PartCatalogueAction_de.properties	German resource files.
PartCatalogueAction_en.properties	English resource files.
PartCatalogueAction_fr.properties	French resource files.

> **Note:** For detailed information on PartCatalogueAction.java, PartInfo.java or PartListAction.java, refer to the Javadocs included with the VueAction sample.

Running the VueAction Sample

The following steps detail how to test the VueAction sample.

1. Double-click customjvue.bat.
   AutoVue launches and populates the toolbar with the **None**, **Description** and **ID** buttons.

2. To test the hotspot implementation, open the hotspot sample file, Basell_AutoVue_Case_Study.pdf.

> **Note:** The hotspot definition file, hotspots.txt, is configured for Basell_AutoVue_Case_Study.pdf. If you want to load another file, you must updated the hotspot definitions in hotspots.txt.

3. Click **Description** to allow hotspot detection and to view tooltips. Alternately, you can click **ID** to allow hotspot detection for the hotspot ID.

   The following regular expressions are defined in hotspots.txt: *AutoVue.** and *Document*. That is, when you hover the mouse cursor over "AutoVue", the string along with any inline text t hat follows it is highlighted and a "AutoVue 2D Professional" tooltip appears. For the "Document" string, the string is highlighted and the "Basell Document" tooltip appears.

Customizing the VueAction Sample

The VueAction sample can be customized to be used with a different file and with data from enterprise visualization systems. Take note that all hotspots are defined in hotspots.txt. In this file, you can specify the definition key, regular expression, whether

the text search should match case, define a tooltip, and so on. For more information on defining hotspots, refer to "Hotspot Definition Parameters".

The following steps describe how to update the VueAction sample with a customized hotspots.txt.

1. Updated hotspot definitions in hotspots.txt.

2. Save hotspots.txt.

3. Extract the files from VueActionSample.jar.

4. From the extracted JAR file, replace hotspots.txt with your customized file.

5. Create a new JAR file, VueActionSample.jar.

6. Run the batch file, customjvue.bat.

7

ABV Design and Security Recommendations

The following are design and security recommendations for your ABV solution.

- Store hotspot definitions in a database and set them dynamically rather than hard-coding in the html.

- Use JSON or XML and XMLHttpRequest to pass visual dashboard information from ABV server component to ABV client components.

- Be aware that if using regional hotspots, the hotspots may need to be updated when the drawing changes

- Ensure JavaScript and Java logging is on - need to set VERBOSE=true in applet parameters.

- If you are using custom GUIFILEs - ensure VueActionHotspots is included in all GUIFILES.

- Be aware that performHotSpot method does not generate errors if given an invalid hotspotKey --- ensure that your hotspotKey is correct if things are not working.

8

Feedback

If you have any questions or require support for ABV please contact your system administrator. If the administrator is unable to resolve the issue, please contact Oracle Corp.

General AutoVue Information

Telephone	+1.514.905.8434 or 1.800.363.5805
Web Site	http://www.oracle.com/us/products/applications/autovue/index.html
Blog	http://blogs.oracle.com/enterprisevisualization/

Oracle Customer Support

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

My Oracle Support AutoVue Community

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

Sales Inquiries

E-mail	autovuesales_ww@oracle.com