This document provides a comprehensive overview of the communication between the VDS and VEO Admin. applications, via the Aareas Interactive API and the embedded iFrame used for visualizing home design options. It describes the API calls made to Aareas for retrieving visualization data and rendering information as well as detailing the messages between VDS and its embedded visualizer’s iFrame, powered by Aareas, to control various aspects of the visualized view. The following documentation aims to facilitate understanding and implementation for developers and stakeholders involved in maintaining and enhancing the VDS platform and describe the interface between VDS and Aareas that make visualization in VDS possible.

# API CALL: FETCH stream URL

VDS calls the Aareas API to fetch a URL that is then used as the source attribute for the <iFrame> element used in the embedded visualizer for DMH and eventually Virtual Model Home.

## REQUEST

**Endpoint:** GET https://apirc.aareas.com/api/Image/GetImage/buildon

**Example URL:**

https://apirc.aareas.com:443/api/Image/GetImage/buildon?sceneId=56794&room=Kitchen&size=2560&userId=6026fbd9-a945-414c-9075-78e9e3723f76|24cf3387-963a-4505-b21b-8b5c2e179505&doNotRenderImage=true&keepCamPosition=False&client=BuildOn&builder=STAGGERED-QA&project=demo&unit=123 Anywhere Way, Houston, TX 77002&applications=[{"Application":"Bedroom 2\_Floor","Product":"f581e34a-35f1-4ce4-a3bd-52d29b46504b"},{"Application":"Kitchen\_Cabinet Door","Product":"2873d388-bc7b-4ad2-8a29-01723036c73c"},...]\*

\**applications list truncated for brevity*

**Query Parameters:**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| sceneId | string | Scene identifier |
| room | string | Room identifier (optional, included if UseDefaultCamera is true) |
| size | string | Width of the frame to be returned |
| userId | string | Consists of <user’s security token> | <session user id> |
| doNotRenderImage | string | Set to "true" |
| keepCamPosition | string | Indicates if camera position should be kept. |
| client | string | Set to "BuildOn" |
| builder | string | Organization name |
| project | string | Community name if billable, otherwise "demo" |
| unit | string | Full address |
| applications | string | JSON string of application and product pairs |

## RESPONSE

**Response Object:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| frameUrl | String | Source URL to be used in the iFrame |

**Example API Response:**

{

"frameUrl":

"https://damrc.aareas.com/RenderImage/index.html?model=&room=KT&doNotRenderImage=True&env=&clientUserId=6026fbd9-a945-414c-9075-78e9e3723f76|24cf3387-963a-4505-b21b-8b5c2e179505&pakname=&size=2560&sceneId=56794&user=unknow@aareas.com&pakVersion=&applications=[{"SurfaceName":"B2\_FLR","DamId":"M18866","StyleId":null},{"SurfaceName":"KT\_CDU","DamId":"M18539","StyleId":"P17927"},...]&keepCamPosition=False&client=BuildOn&project=demo&unit=123 Anywhere Way, Houston, TX 77002&builder=STAGGERED-QA"

}

# api cALL: FETCH AVAILABLE SURFACES

The VEO Admin. application calls the Aareas API to fetch a list of surfaces available to be rendered in a scene in the visualizer.

## Request

**Endpoint:** GET

https:// apirc.aareas.com/api/SceneSurface/GetClientSurfaceList

**Example URL:**

https://apirc.aareas.com/api/SceneSurface/GetClientSurfaceList/b73ce491-bc27-42a7-ad85-6463eca43bfd/53661/Kitchen

**Route Parameters:**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| clientId | string | Client identifier |
| sceneId | string | Scene identifier |
| roomId | string | Room identifier |

## Response

**Response Object:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ResponseObject | Array of SurfaceListResponse | List of surfaces |
| ResponseMessage | string | Message describing the response status |
| ResponseType | Int | Type of the response |
| ResponseStatus | Int | HTTP status of the response |

**SurfaceListResponse Object:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SceneId | string | Identifier for scene |
| RoomName | string | Name of room |
| RenderableSurfaceList | Array of strings | List of identifiers for renderable surfaces |

**Example API Response:**

{

"ResponseObject": [

{

"SceneId": "b73ce491-bc27-42a7-ad85-6463eca43bfd",

"RoomName": "Kitchen",

"RenderableSurfaceList": [

"Surface1",

"Surface2"

]

}

],

"ResponseMessage": "Success",

"ResponseType": 1,

"ResponseStatus": 200

}

# API CALL: FETCH VISUALIZABLE OPTIONS

VDS and VEO Admin. both require knowledge of the options that can be visualized, using Aareas as the visualization provider. VDS calls this endpoint to provide the user, within the visualizer, with information about which options they can visualize. VEO Admin calls the endpoint to determine which options are visualizable for each program in the Visualization Programs view.

## Request

**Endpoint:** POST

https:// apirc.aareas.com/api/ClientProduct/GetClientProductlist/BuildOn

**Example URL:**

https://apirc.aareas.com/api/ClientProduct/GetClientProductlist/BuildOn

**Request Body:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GUIDS | String | Comma-separated list of product GUIDs |

**Example Request Body:**

{

"GUIDS": "481c34b1-065e-4a67-a345-2841363a3f33,cb3ae381-de5a-482a-903a-

2a726d7e3fc4,d5a9f72b-8c29-496c-902e-724e52847bce,…"

}

## Response

**Response Object:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ResponseObject | Array of GUID | List of renderable product GUIDs |
| ResponseMessage | string | Message describing the response status |
| ResponseType | Int | Type of the response |
| ResponseStatus | Int | HTTP status of the response |

**Example API Response:**

{

"ResponseObject": [

"481c34b1-065e-4a67-a345-2841363a3f33",

"cb3ae381-de5a-482a-903a-2a726d7e3fc4",

...

],

"ResponseMessage": "Client Products found",

"ResponseType": 1,

"ResponseStatus": 0

}

# VDS COMMUNICATION WITH IFRAME

The postMessage browser API is a way to safely enable cross-origin communication between different browsing contexts, such as between a parent window and an iFrame. It allows the parent window to send, or publish, messages to the iFrame and listen for and respond (a.k.a. “subscribe”) to events sent from the iFrame. Since VEO Design Studio’s embedded visualizer uses an iFrame to implement the visualizer powered by Aareas Interactive, this two-way communication is instrumental in how the visualizer functions.

## MESSAGES PUBLISHED TO IFRAME

### **postMessageToStream**

The postMessageToStream function uses the postMessage API to publish a message to the iFrame’s content window. The main application can control various aspects of the visualized view within the iFrame by sending specific messages. The iFrame subscribes to these messages and updates the visualized content according to the specific messages that are received. The postMessageToStream function is used by three other methods, each with a specific purpose related to the visualization.

### **postApplyMaterialsMessage**

As different options are applied to the visualized view, the postApplyMaterialsMessage function sends a message to the iFrame, by calling postMessageToStream, to apply a new set of materials to the visualizer's rendered view. Imagine you want to see how different cabinets or flooring materials look in a virtual room. This method sends a message to update the materials displayed in the iFrame, so you can visualize the changes in real-time.

In addition to the event, a URL is passed to Aareas, containing surface and product information.

**Example Message:**

{

"event": "Apply Materials",

"url":

"https://damrc.aareas.com/RenderImage/index.html?model=&room=&doNotRenderImage=True&env=&clientUserId=7305bda7-bb77-4108-9ce8-ccc94aa1d216|24cf3387-963a-4505-b21b-8b5c2e179505&pakname=&size=2560&sceneId=56794&user=unknow@aareas.com&pakVersion=&applications=[{\"SurfaceName\":\"B2\_FLR\",\"DamId\":\"M18866\",\"StyleId\":null},{\"SurfaceName\":\"KT\_FLR\",\"DamId\":\"M21019\",\"StyleId\":null},{\"SurfaceName\":\"EN\_CDD\",\"DamId\":\"M18547\",\"StyleId\":\"P24587\"},{\"SurfaceName\":\"EN\_CTP\",\"DamId\":\"M18693\",\"StyleId\":null},{\"SurfaceName\":\"EN\_FLR\",\"DamId\":\"M18901\",\"StyleId\":null},{\"SurfaceName\":\"EN\_FLR\",\"DamId\":\"M7468\",\"StyleId\":null},{\"SurfaceName\":\"KT\_CDU\",\"DamId\":\"M20509\",\"StyleId\":\"P17927\"}]&keepCamPosition=False&client=BuildOn&project=demo&unit=123 Anywhere Way, Houston, TX 77002&builder=STAGGERED-QA"

}

### **postSwitchCameraMessage**

Switching the camera’s point of view to a different area or angle within the visualized space is accomplished by calling postSwitchCameraMessage. It constructs a message object, with event type and surface name, and passes it to postMessageToStream to switch the camera view in the iFrame. This allows the user to explore the virtual space from different perspectives.

Along with the event, the surface name is passed to Aareas.

**Example Message:**

{

"event": "Switch Camera",

"value": "Kitchen\_Backsplash"

}

### **postToggleHotspotsMessage**

The postToggleHotspotsMessage function sends a message to toggle the visibility of hotspots in the visualizer’s rendered view. Hotspots are interactive markers, represented by red dots, that help the user navigate the virtual space. This method sends a message to show or hide these markers, making it easier to focus on the room’s design or explore the layout. VDS also uses this capability to turn the hotspots off in order to capture higher quality screenshots of the visualized view.

A display value of true/false is passed to areas, depending on whether the hotspots are to be toggled on or off respectively.

**Example Message:**

{

"event": "Toggle Hotspots",

"display": false

}

## EVENTS SUBSCRIBED TO FROM IFRAME

VDS subscribes to events on the iFrame listening for messages like renderedimage, camchange, and roomswitched from the iFrame, enabling real-time updates and seamless communication between the main application and the visualizer.

**renderedimage**

This event is fired when the iFrame has finished rendering an image and provides the rendered image data. This allows VDS the capability to save scene snapshots from the visualizer once the rendering process completes.

**Event Structure:**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| event | string | The event type, set to "renderedimage" |
| data | string | The rendered image data encoded in base64 format |

**Example Event:**{

"event": "renderedimage",

"data": "..."

}

### **camchange**

This event is fired when the camera angle or position changes, which occurs as the user “moves” through the virtual space.

**Event Structure:**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| event | string | The event type, set to "camchange" |
| data | string | Set to “Camera position change” |

**Example Event:**{

"event": " camchange ",

"data": "Camera position change"

}

**roomswitched**

This event is fired when the user switches rooms in the visualizer.

**Event Structure:**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| event | string | The event type, set to "roomswitched". |
| data | array | An array containing the new room information. |

**Example Event:**{

"event": "roomswitched",

"data": ["Kitchen\_Cabinet Door"]

}