Tutorial 5: How to Use the Web Interface to Control VRED Remotely with Python Script

Download the example scripts here: https://github.com/raemorris/VRED-Python-Tutorial-Examples-Scripts/blob/main/tutorial5.zip?raw=true

Sample Python Code

Here are the accompanying example Python scripts for the *Tutorial 5: How to Use the Web Interface to Control VRED Remotely with Python Script* video.

- To view the video, visit: https://www.youtube.com/watch?v=ESkXH9bqo8Y
- To review the video caption, see *Video Captions*.

custom-api.js

```
import { api } from 'http://localhost:8888/api.js';


// Connect event listener (signal)

api.vrAnnotationService.annotationsAdded.connect(() =>
console.log('Annotations added'));


// Add an annotation with name and text to the scene

function addAnnotation(name, text) {

    api.vrAnnotationService.createAnnotation(name)

    .then(() => {

        api.vrAnnotationService.findAnnotation(name)

        .then(a => a.setText(text));

    })

    .catch(() => console.error('Add annotation failed'));

}


// Get a list of all annotations
```

```
function getAnnotations() {
    api.vrAnnotationService.getAnnotations()
    .then(annotations => annotations.forEach(a => console.log(a)))
    .catch(() => console.error('Get annotations failed'));
}

// Add both functions to the global scope
// So that we can call them from the html document
window.addAnnotation = addAnnotation;
window.getAnnotations = getAnnotations;
```

custom_web_interface_endpoints.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"/>
        <script>
            // Send a generic python command without return value
            function sendPythonCommand(command) {
                var http = new XMLHttpRequest();
                var url = 'http://localhost:8888/python?value=' +
encodeURI(command);
                http.open('GET', url, true);
                http.send();
            }

            // Send a python command with expecting a return value
            function sendPythonCommandWithReturnValue(command) {
                var http = new XMLHttpRequest();
                var url = 'http://localhost:8888/pythoneval2?value=' +
encodeURI(command);
                http.open('GET', url, true);
```

```
                http.onreadystatechange = function() {
                    if (this.readyState == 4 && this.status == 200) {
                        var response = eval(this.responseText);
                        console.log(response);
                    }
                };
                http.send();
            }

            // Write a message to the terminal
            function logInfo(message) {
                sendPythonCommand("logInfo('"+message+"')");
            }

            // Select a variant set
            function selectVariantSet(variantSet) {
                sendPythonCommand("selectVariantSet('"+variantSet+"')");
            }

            // get all viewpoints
            function getViewpoints() {
                sendPythonCommandWithReturnValue('getViewpoints()');
            }
        </script>
    </head>
    <body>
        <!-- Defining your custom python actions -->
        <button onclick="logInfo('This is a message')">Print Info
Message</button>

        <button onclick="selectVariantSet('Black Metallic')">Variant: Select
Black Metallic</button>
```

```
        <button onclick="getViewpoints()">Request Viewpoints</button>
    </body>
</html>
```

custom_web_interface_web_api.html

```
<!DOCTYPE html>
<html>
    <head>
        <script type="module" src="custom-api.js"></script>
    </head>
    <body>
        <!-- Defining your custom python actions -->
        <button onclick="addAnnotation('New Annotation', 'This is a new annotation')">Add annotation</button>
        <button onclick="getAnnotations()">Get all annotations</button>
    </body>
</html>
```

Video Captions

Hello and welcome to our next Python tutorial for VRED Pro. I'm your host Christopher and today I will show you how you use VRED's web interface to control VRED remotely with Python scripts.

You already might be familiar with VRED's web interface and how can use it. When activated, VRED starts a web server that hosts a web app to remotely control VRED with a browser. It's perfect for design reviews or customer presentations because you can move the camera, have a live video stream, and can trigger any variant sets you have defined.

To enable the web interface you have to go to the preferences and scroll down to "webinterface". Here you can enable the webinterface. When you apply the changes, you are basically ready to use the webinterface on your local host. But when you want to have access with an iPad, for example, you will have to set the Host Access to either "no restriction" or add your iPad to your host list with its IP address.

For testing, it's totally fine to set the host access to "no restriction", but for security you should always work with a host list. When you are in the same network, you can just enter the IP address of the device you are using, and you can connect to the web interface. There are some other settings and I will not

cover them all in this tutorial, but all the settings are explained in the online documentation for the webinterface preferences.

Right now, I'm connected to the VRED scene with my iPad and you see I can control the scene with my fingers and switch between the variant sets. I can take snapshot, change the settings, and so on. So, when you want to do a simple presentation, these features would probably be enough. But the VRED streaming app also makes it possible to directly enter a Python script that is then executed in the Terminal. There is already an example here that creates new scene and adds a box to the Scene Graph.

So, with the Terminal, you can execute any Python code from your scene. Either you directly use the Python API to create geometry, like in the terminal example or you call functions that are accessible from your scene, like functions defined in your Script Editor or functions from imported Python modules. The Terminal, here, acts quite like the Script Editor when in VRED. Everything you enter is directly executed and interacts with your scene.

Okay, now you know how you can use the Python Terminal to control VRED using the default streaming app. Another cool method to call your custom Python scripts is to use variant sets. Just create a variant set group called "Scripts" or whatever you like and add variant sets that contains the Python code you want to execute. This way you don't have to program a custom interface just to have buttons for a certain action. The default streaming app already handles this for you and creates a menu entry in the variant sets.

Of course, it's also possible to build your own web interface that can interact with VRED. When doing this, you can fully customize your Remote App and style it exactly as you want.

First, I show you how you can interact with VRED using endpoints. An endpoint is just a GET request to the web interface that contains the Python code you want to execute. Here, I will show you an example how to select a variant set and how to trigger other Python functions within a custom web app. We can send other query Python functions to an endpoint that is provided by the webinterface.

In our first example, we are using the "Python" endpoint. This will send a Python command to VRED and expects no return value. The Python command is delivered with the "value" parameter and has to be URL encoded. With a click of the button, we send our message to the endpoint and can now print a message in the terminal. We use the same method to select variants sets. Here I select the black metallic material, using the variant set that is defined in our scene.

There is another endpoint that can also return data from the webinterface. This endpoint is called "Python eval 2". Like the "Python" endpoint, we can send a command to VRED, but now have to manage the return values, as well. In this case, we request all viewpoints and receive an array of our viewpoints that we print to the console. You have to watch out what data format is returned by the GET requests. In this case, it's a string that can be evaluated to a javascript array.

The modern approach on building a custom webinterface, however, is by using the "web API" Autodesk introduced with version 2021. The web API is a javascript application that integrates in you web app and enables full access to VRED's Python API, without the need to use the Python endpoints. The web API supports all the modules and functions from the Python API version 2.

So, for calls to the API version 1, you would still use the provided endpoints as in our previous example. We can add the web API to our app by importing the script from the webinterface. We create a new javascript module where we can import the API. Then, we add a function to add an annotation and a function to get all annotations back. Also, we connect to a signal event whenever an annotation is added to VRED. This here is a rather simple setup and normally you would include this in a full-fledged web app. Here, I just want to show you the general idea.

To make our functions visible in the HTML document, we have to add them to the global scope. This is done by adding the functions to the window object. In our HTML document, we just link to the new javascript file as type module and can now call the functions when one of our buttons is clicked.

We see from the example that we can call the javascript function, much like the Python version, by referring to the services and calling its functions. We also can use nice javascript features like promises and error handling to deal with the data we receive in an asynchronous way.

To run this example, I have to serve the files with a local webserver. This is because cross-origin resource policies don't allow access to our javascript files when running locally. For this, I use the tool HTTP Server that I installed with node. But of course, you can use any server you like.

I can visit a page, under the address my local webserver provided, and when I click the buttons, we can see that an annotation was added to the scene and the console also shows the new annotation when we request a list of all annotations. Autodesk also included examples of their web app in the examples directory of VRED. So, if you are interested in building you own custom web app, this would also be a good starting point. You probably want to start with the stream js example, as it is the simplest one. Stream app, however, is based on a react js app and a little bit more complex.

The web interface and its web API offer whole new possibilities in interacting with VRED. We can now easily build a fully customized streaming app and build whole new tools on top of the engine.

Okay, that's it for today. I hope you enjoyed our video and see you next time!