

# vRealize Orchestrator

## Integration Guide

### Overview

This document will discuss integration between vRealize Orchestrator and UpGuard. UpGuard provides a robust API for integration. This solution will highlight two ways to achieve one common integration point; “UpGuard Node Registration”. The first solution utilizes vRO’s persistent REST host and REST operation objects and pre-built workflows. The second solution is “Script-only” and utilizes Transient REST host and REST operation objects.

UpGuard Node Registration is a common requirement in many use-cases such as Server Provisioning and triggering configuration snapshots before and after changes. Node Registration involves three actions; node lookup, node create; and node scan. Each action has a corresponding UpGuard API endpoint.

### At-A-Glance

VMware vRealize™ Orchestrator™ 5.5 (formerly vCenter™ Orchestrator™) is a drag-and-drop workflow software that simplifies the automation of complex IT tasks. It integrates with VMware vCloud Suite™ components to adapt and extend service delivery and operational management, thereby effectively working with existing infrastructure, tools and processes. It enables administrators to develop complex automation tasks, then quickly access and launch workflows from the VMware vSphere® client, various components of VMware vCloud Suite, or vRealize Orchestrator.

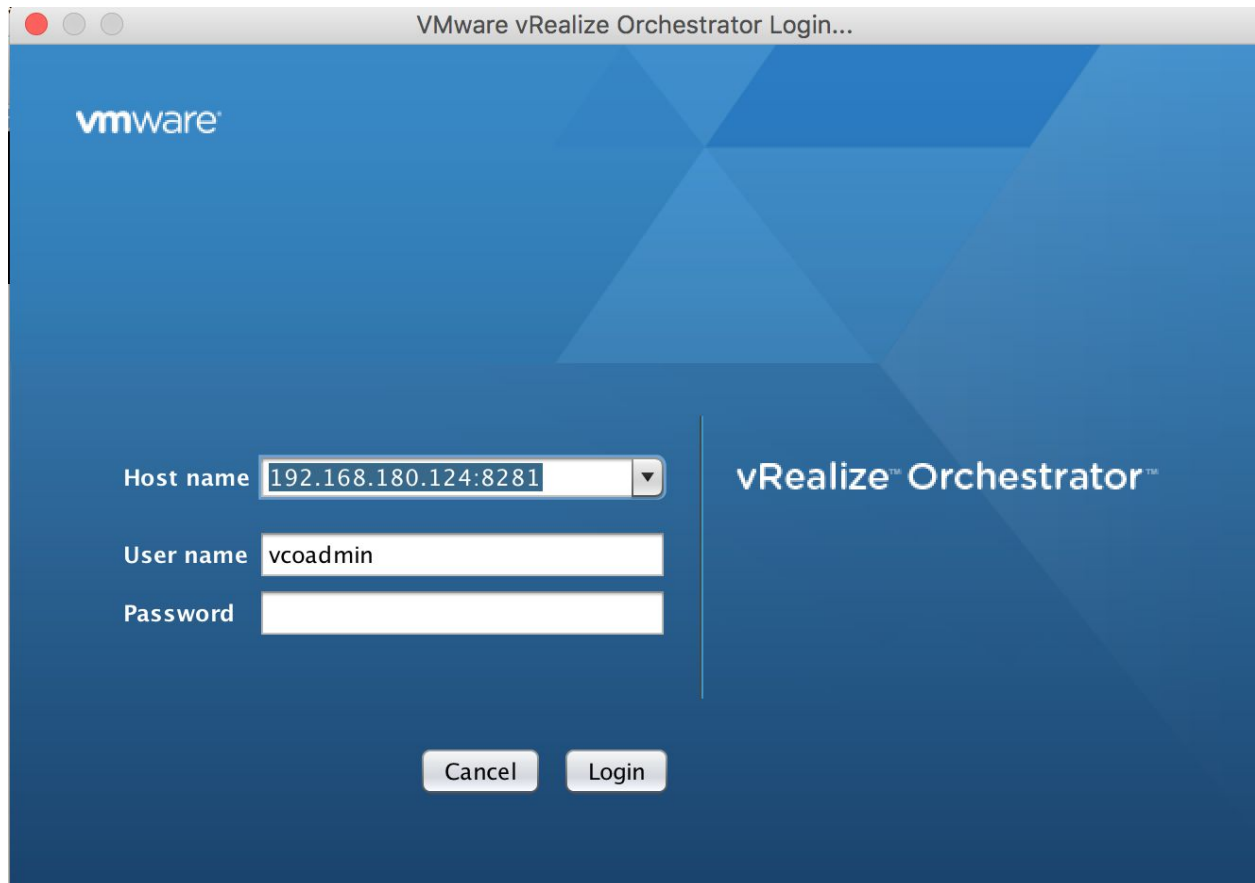
### Requirements

- vRealize Orchestrator 7.0+
- Javascript knowledge
- UpGuard API v2+



# Setup

1. Download and Install Java <https://java.com>
2. Download and Install Orchestrator Client <https://192.168.180.124:8281/vco/>
3. Launch “vRO Workflow Designer”
4. Connect to:
  - a. Hostname: 192.168.180.124.8281
  - b. User name: vcoadmin
  - c. Password: \*\*\*\*\*



5. Switch from Run to Design mode

## 6. Complete HTTP-REST Configuration

- a. Create a REST host <https://demo.upguard.com>

The screenshot shows the 'Start Workflow : Add a REST host' window. On the left, a sidebar lists the steps: 1 Add a REST host (selected), 2 Host Authentication, 3 Proxy Settings, and 4 Advanced. Under step 1, sub-step 1a Host Properties is selected. The main area contains the following fields and options:

- Name:** UpGuard DEMO
- URL:** https://demo.upguard.com
- Connection timeout (seconds):** 30.0
- Operation timeout (seconds):** 60.0
- If set to true, the certificate is accepted silently and the certificate is added to the trusted store:** ☒ Yes ☐ No
- Automatically URL Redirection:** ☐ Yes ☒ No

At the bottom right, there are four buttons: Cancel, Back, Next, and Submit.

- b. Create a REST operation /api/v2/nodes.json GET

The screenshot shows the 'Start Workflow : Add a REST operation' window. On the left, a sidebar lists the steps: 1 Add a REST operation (selected), and 1a Operation Properties (selected). The main area contains the following fields and options:

- Parent host:** UpGuard: https://demo.upguard.com
- Name:** UpGuard Nodes DEMO
- Template URL:** /api/v2/nodes.json
- HTTP method:** GET (selected from a dropdown menu)

At the bottom right, there are two buttons: Cancel and Submit.

c. Create a REST operation `/api/v2/nodes.json` POST

Start Workflow : Add a REST operation

**1 Add a REST operation**  
1a Operation Properties

Properties to create a new operation.

The URL must include only the specific operation part (without the host's URL) and can contain placeholders for parameters that are provided at request run time. Examples:  
/customer/{id}  
/customer/{id}/orders?date={date}  
/customer/orderBy={orderBy}

\* Parent host  
UpGuard: https://demo.upguard.com

\* Name  
UpGuard Node POST DEMO

\* Template URL  
/api/v2/nodes.json

\* HTTP method  
POST

Content type

Cancel Submit

d. Create a REST operation `/api/v2/nodes/{node_id}/start_scan.json?label={label}` POST

Start Workflow : Add a REST operation

**1 Add a REST operation**  
1a Operation Properties

Properties to create a new operation.

The URL must include only the specific operation part (without the host's URL) and can contain placeholders for parameters that are provided at request run time. Examples:  
/customer/{id}  
/customer/{id}/orders?date={date}  
/customer/orderBy={orderBy}

\* Parent host  
UpGuard: https://demo.upguard.com

\* Name  
UpGuard Node Scan DEMO

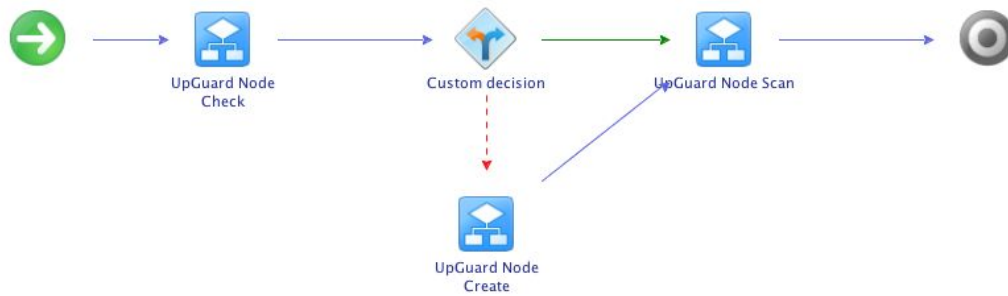
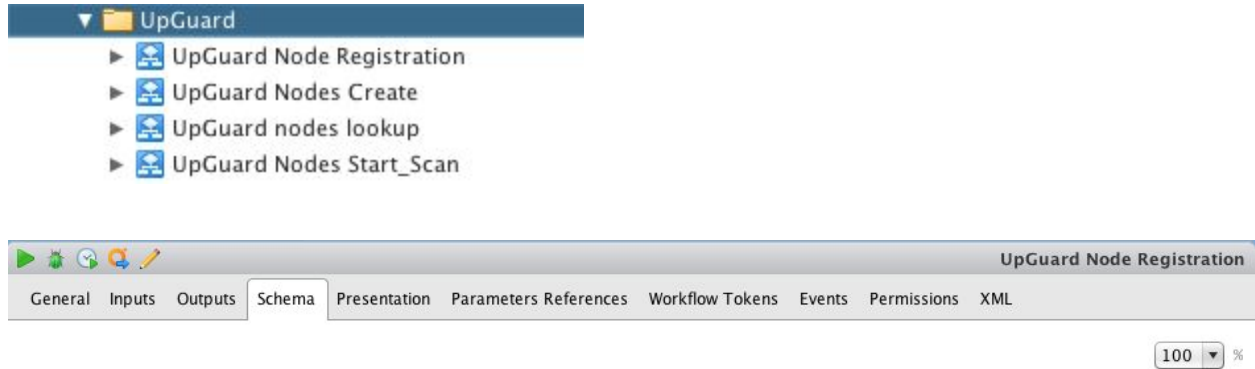
\* Template URL  
/api/v2/nodes/{nodeid}/start\_scan.json?label=vRO

\* HTTP method  
POST

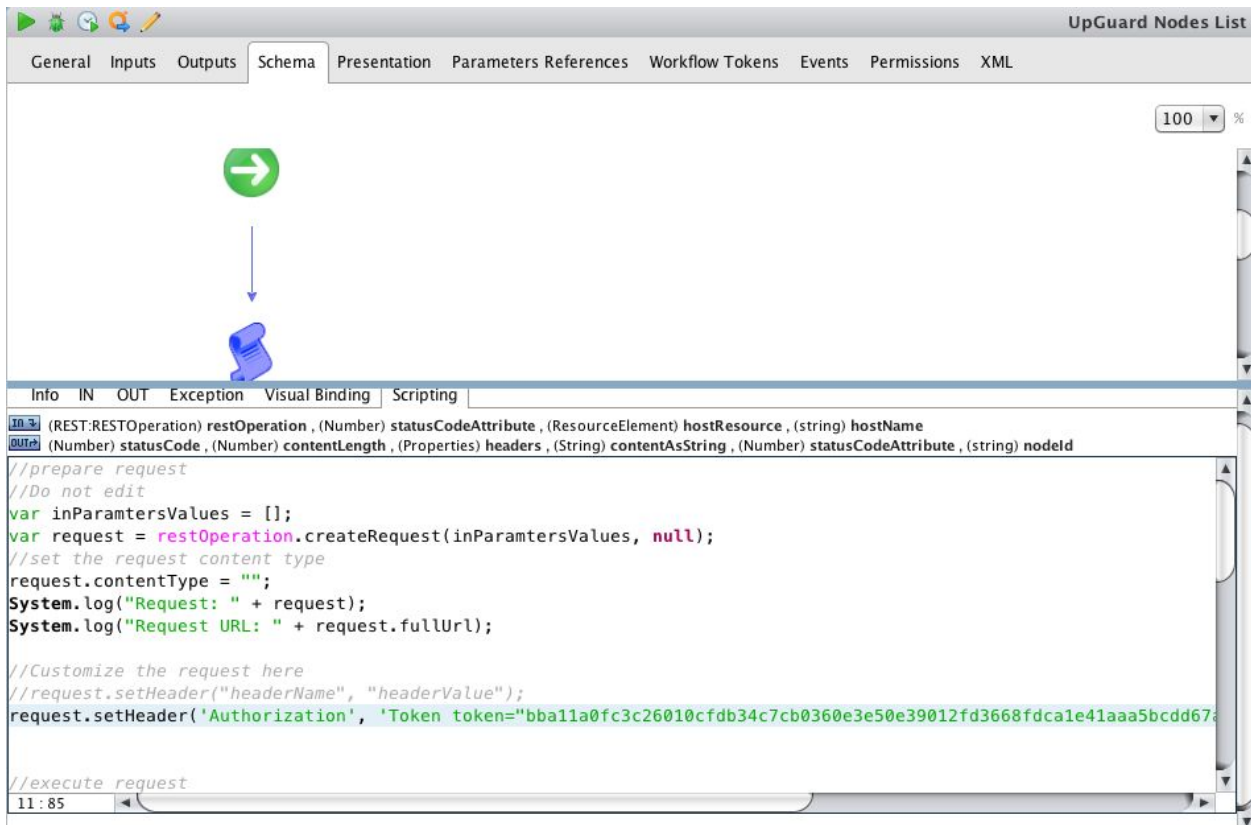
Content type  
application/json

Cancel Submit

7. Import UpGuard Nodes Lookup Workflow
8. Import UpGuard Nodes Create Workflow
9. Import UpGuard Nodes Scan workflow
10. Import UpGuard Node Registration Workflow



## 11. Update “Authorization” token in each workflow Script, or create a parameter to pass through workflows.



The screenshot shows the 'UpGuard Nodes List' application. The 'Schema' tab is selected, displaying a diagram of a workflow node with a green arrow pointing to a blue document icon. Below the diagram, the 'Scripting' tab is active, showing a JavaScript script for a REST client. The script includes comments and code for preparing and executing a request. The 'Authorization' header is set to a specific token.

```
//prepare request
//Do not edit
var inParametersValues = [];
var request = restOperation.createRequest(inParametersValues, null);
//set the request content type
request.contentType = "";
System.log("Request: " + request);
System.log("Request URL: " + request.fullUrl);

//Customize the request here
//request.setHeader("headerName", "headerValue");
request.setHeader('Authorization', 'Token token="bba11a0fc3c26010cfdb34c7cb0360e3e50e39012fd3668fdca1e41aaa5bcd67a');

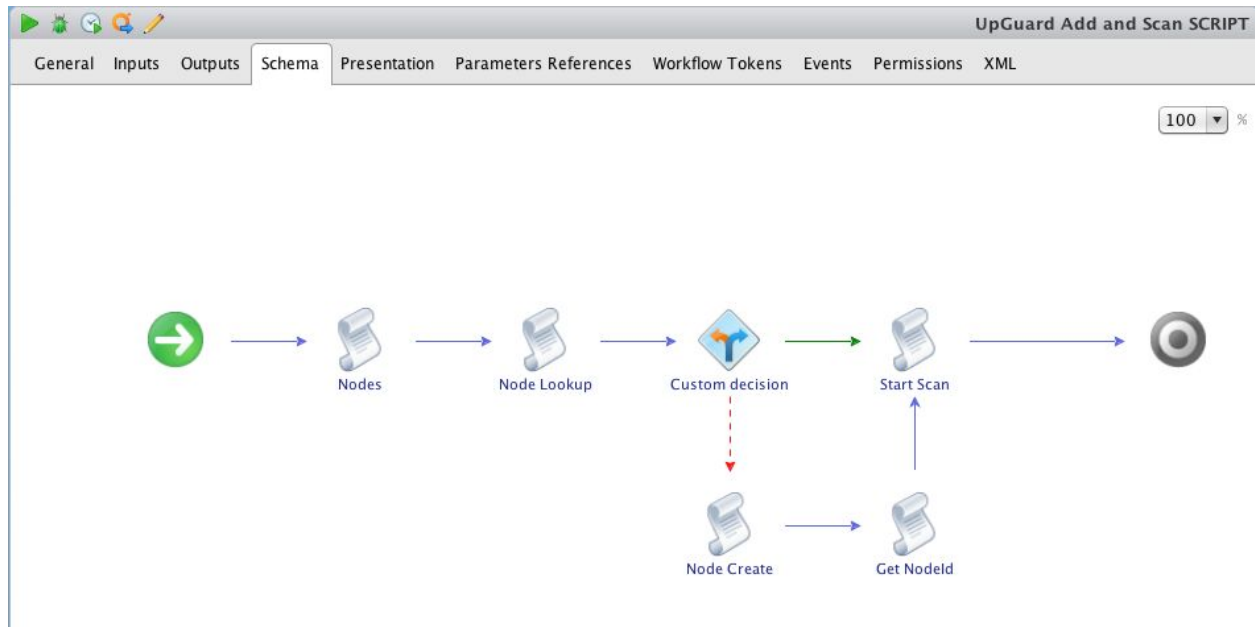
//execute request
```

## 12. Sample Logging Info

```
[2017-07-28 19:59:32.843] [I] Request: DynamicWrapper (Instance) : [RESTRequest]-[class
com.vmware.o11n.plugin.rest.Request] -- VALUE : com.vmware.o11n.plugin.rest.Request@6ed3944e
[2017-07-28 19:59:32.851] [I] Request URL:
https://demo.upguard.com/api/v2/nodes/lookup.json?name=3D-Cart
[2017-07-28 19:59:32.987] [I] Response: DynamicWrapper (Instance) : [RESTResponse]-[class
com.vmware.o11n.plugin.rest.Response] -- VALUE :
com.vmware.o11n.plugin.rest.Response@11eeca1f
[2017-07-28 19:59:32.988] [I] Status code: 200
[2017-07-28 19:59:32.989] [I] Content as string: {"node_id":352}
[2017-07-28 19:59:32.990] [I] NodeId: 352
[2017-07-28 19:59:33.084] [I] Request: DynamicWrapper (Instance) : [RESTRequest]-[class
com.vmware.o11n.plugin.rest.Request] -- VALUE : com.vmware.o11n.plugin.rest.Request@1d73b08f
[2017-07-28 19:59:33.085] [I] Request URL:
https://demo.upguard.com/api/v2/nodes/352/start_scan.json?label=vRO
[2017-07-28 19:59:33.528] [I] Response: DynamicWrapper (Instance) : [RESTResponse]-[class
com.vmware.o11n.plugin.rest.Response] -- VALUE :
com.vmware.o11n.plugin.rest.Response@2b3ce197
[2017-07-28 19:59:33.529] [I] Status code: 201
[2017-07-28 19:59:33.530] [I] Content as string: {"job_id":31796}
```

# Script-Only Solution

This solution utilizes javascript and available script classes within vRealize Orchestrator workflow engine. While this script may be more portable, it also tends to be more redundant as objects are not reusable across multiple workflows and steps.



## Sample Output

```
[2017-07-28 20:10:45.117] [I] Creating transient REST host with base URL:
https://demo.upguard.com
[2017-07-28 20:10:45.125] [I] Request URL:
https://demo.upguard.com/api/v2/nodes/lookup.json?name=vrotestnode
[2017-07-28 20:10:45.200] [I] Status code: 200
[2017-07-28 20:10:45.233] [I] Parsing nodeId
[2017-07-28 20:10:45.263] [I] Starting scan for nodeId: 4570
[2017-07-28 20:10:45.265] [I] Creating transient REST host with base URL:
https://demo.upguard.com
[2017-07-28 20:10:45.271] [I] Request URL:
https://demo.upguard.com/api/v2/nodes/4570/start_scan.json?label=vRO
[2017-07-28 20:11:46.967] [D] Response content as string: {"job_id":31798}
```



## Script

```
// VMware vRealize Orchestrator action sample
//
// Executes a transient RESTOperation for a transient RESTHost.
// Performs the operation without having the operation nor the host persist in the RESTHost
// Inventory.
// Returns the response string and status code as named key-value pairs "responseString" and
// "statusCode", respectively.
//
// For vRO 7.0+
//
// Action Inputs:
// baseUrl - string - Base URL for connecting to the RESTful application
// username - string - Optional username for Basic authentication to the REST host
// password - SecureString - Optional password for Basic authentication to the REST host
// opMethod - string - Method of the REST operation
// opUrl - string - URL template of the REST operation including variablized in-line
// arguments; e.g., /items/{arg1}
// urlParamValues - Array/string - In-line parameter values, if any
// headers - Array/CompsiteType(key:string,value:string) - Optional headers of the request
// contentType - string - Optional content type of the request body (null for GET and DELETE
// operations)
// content - string - Content of the request body (null for GET and DELETE operations)
//
// Return type: CompsiteType(statusCode:string,responseString:string) - The REST response
// string and status code as named key-value pairs

var baseUrl = "https://{URL}";
var username = null
var password = null
var opMethod = "GET";
var opUrl = "/api/v2/nodes.json?per_page=10000";
var urlParamValues = [];
var header = {"key":"Authorization","value":"Token token=\"{APIKEY}\""}
var headers = [header]
var contentType = "Application/json";
var content = "";

// Create transient host and Op
var host = createHost(baseUrl, username, password);
var op = createOp(host, opMethod, opUrl);

// Execute request
var request = setRequest(op, setUrlParamValues(urlParamValues), headers, contentType,
content);

var response = request.execute();

//Process response
var responseString = parseResponse(response);
var statusCode = response.statusCode;

// Create a transient RESTHost
// If given user/password, uses Basic auth in Shared Session mode
function createHost(url, user, pw) {
    System.log("Creating transient REST host with base URL: " + url);
```

```

        var host = new RESTHost(url);
        host.name = generateNameFromUrl(url);
        host.url = url;
        host.hostVerification = false;
        host.proxyHost = null;
        host.proxyPort = 0;
        host.authentication = createSharedBasicAuth(user, pw);

        host = RESTHostManager.createTransientHostFrom(host);

        RESTHostManager.reloadConfiguration();

        return host;
    }

    // Generate a friendly name for a RESTHost or RESTOperation from a given URL,
    // removing "HTTP" and "HTTPS", and replacing non-words with '_'
    function generateNameFromUrl(url) {
        var name = url;
        name = name.replace(/https:\/\/\/i, '');
        name = name.replace(/http:\/\/\/i, '');
        name = name.replace(/\W/g, '_');
        return name;
    }

    // Instantiate REST Basic authentication in Shared Session mode
    function createSharedBasicAuth(user, pw) {
        if (!isSet(user) || !isSet(pw)) {
            return null;
        }

        var authParams = ["Shared Session", user, pw];
        var authObject = RESTAuthenticationManager.createAuthentication("Basic", authParams);

        System.log("REST host authentication: " + authObject);

        return authObject;
    }

    // Is a given string non-null and not empty?
    function isSet(s) {
        return s != null && s != "";
    }

    // Create a transient RESTOperation
    // For POST and PUT, the default content type is application/json
    function createOp(host, method, url) {
        var name = generateNameFromUrl(url);

        var op = new RESTOperation(name);
        op.method = method;
        op.urlTemplate = url;
        op.host = host;

        if (method.toUpperCase() === "POST" || method.toUpperCase() === "PUT") {
            op.defaultContentType = "application/json";
        }

        System.debug("Creating operation '" + name + " with URL '" + url + "'");
        System.debug("New operation: " + op);

        op = RESTHostManager.createTransientOperationFrom(op);

        return op;
    }

```

```

}

// If no in-line parameter values are given, return empty array by default
function setUrlParamValues(urlParamValues) {
    return (!urlParamValues) ? [] : urlParamValues;
}

// Prepare the RESTRequest object for executing the RESTOperation
function setRequest(op, urlParamValues, headers, contentType, content) {
    var request = op.createRequest(urlParamValues, content);
    request.contentType = contentType;
    for each (var header in headers) {
        System.debug(header.key);
        request.setHeader(header.key, header.value);
    }

    System.debug("Request to execute: " + request);
    System.log("Request URL: " + request.fullUrl);

    return request;
}

// Parse the RESTResponse object returned from executing a RESTOperation
function parseResponse(response) {
    const HTTP_ClientError = 404;
    var statusCode = response.statusCode;
    System.log("Status code: " + statusCode);

    var headers = response.getAllHeaders();
    for each (var headerKey in headers.keys) {
        System.debug(headerKey + ": " + headers.get(headerKey));
    }

    var contentAsString = response.contentAsString;
    System.debug("Response content as string: " + contentAsString);

    if (statusCode > HTTP_ClientError) {
        throw "HTTPError: status code: " + statusCode;
    } else {
        return contentAsString;
    }
}

```