

TaskList VMWare

Single-instance all instances of the following Script's code (and similar) using a single vRO Workflow:

The screenshot displays a vRO workflow editor. On the left, a tree view shows the 'PSS' library with various tasks. The main workspace shows a workflow sequence: a start node, 'PSS Get Subnet CIDR', 'Get virtual machine', 'Prepare VM & VPort', and a decision node 'Running?'. Below the workflow, a script editor is open, showing a JavaScript function that iterates through an array of VMs and checks for specific configurations.

```
IN: (Array/VC:VirtualMachine) vms
OUT: (string) vmUUID, (string) vmMAC, (string) errorCode, (string) vmName, (VC:VirtualMachine) vm, (string) vmInterfaceName

try {
    if (vms.length == 0) {
        throw "No VMs found with specified Hostname";
    }

    for (var i = 0; i < vms.length; i++) {
        vm = vms[i];

        if (i == 1) {
            throw "Multiple VMs with specified Hostname";
        }

        var vmName = vm.config.name;
        var vmInterfaceName = vmName + "-nic0";
        var vmUUID = vm.config.uuid;
        var vmMAC = "";

        var deviceArray = vm.config.hardware.device;
        if (!deviceArray) {
            throw "Error getting hardware config array of VM";
        }

        var j = 0;

        //loop through array, find NICs and retrieve there MACs
        //assumes only a single NIC configured on a VM
        for (var i in deviceArray) {
            var currentDevice = deviceArray.pop();
            if (currentDevice) {
```

Remove the hardcoded JSON in the following Script's code (and similar) by leveraging vRA Configuration functionality, in the exact same manner as has been done with "PSS Ansible Base Create".

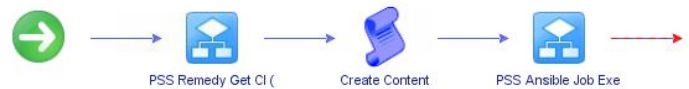
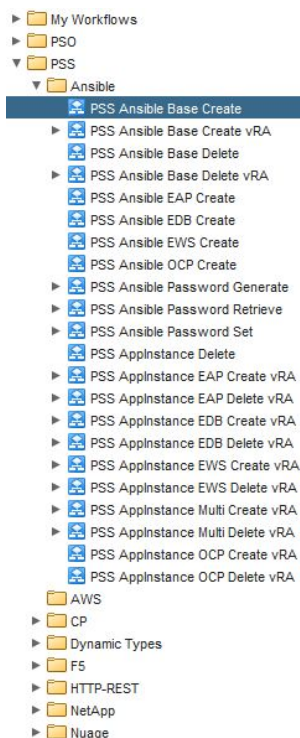


```

Info  IN  OUT  Exception  Visual Binding  Scripting
IN  (string) adJson
OUT (string) contentLaunch
try {
    var adObj = JSON.parse(adJson);

    contentLaunch = '{ ' +
        '"credential": 5, ' +
        '"extra_vars": { ' +
            '"adDomain": ' + adObj.domain + ', ' +
            '"bmcServer": "10.0.0.1", ' +
            '"central_creds": false, ' +
            '"environment": "prod" ' +
        '}' +
    '}' ;
}

```



```

Info  IN  OUT  Exception  Visual Binding  Scripting
IN  (string) adJson , (string) ntpJson , (Array/string) vmNames , (string) appInstanceName , (string) osCreateJson
OUT (string) contentLaunch
var adObj = JSON.parse(adJson);
var ntpObj = JSON.parse(ntpJson);

if (!appInstanceName) {
    appInstanceName = "";
}

var contentLaunchObj = JSON.parse(osCreateJson);

contentLaunchObj.extra_vars.central_creds = false;
contentLaunchObj.extra_vars.timezone = ntpObj.timezone;
contentLaunchObj.extra_vars.adDomain = adObj.domain;
contentLaunchObj.extra_vars.platform_code = vmNames[0].substring(9, 12);
contentLaunchObj.extra_vars.application_instance_name = appInstanceName;
contentLaunchObj.extra_vars.ngdc_environment = vmNames[0].substring(8, 9);

contentLaunch = JSON.stringify(contentLaunchObj);

System.debug("PSS Ansible Base Create - Content Launch: " + contentLaunch);

```

Duplicate the following Avnet RemedyForce vRO Actions similar to the existing two duplicated vRO Actions into the “com.prudential.pss” folder & modify them in the same manner (i.e. simple 3-retries logic):

- RemedyForce_GetRecord

- RemedyForce_GetAllRecords
- RemedyForce_CreateRecord
- RemedyForce_DeleteRecord

The screenshot shows the vRO console with the 'pss_RemedifyForce_UpdateRecord' action selected in the left pane. The main pane displays the script for this action, which is a PowerShell script. The script defines a function 'RemedyForce_UpdateRecord' that takes 'RemedyForceHost', 'resourceName', and 'resourceData' as inputs. It then iterates over a range of 0 to 3, attempting to patch the resource. If successful, it returns the action result; otherwise, it catches the error and debugs it.

```

RemedyForce.UpdateRecord(RemedyForceHost remedyForceHost, string resourceName, string resourceData)
for (var i = 0; i < 3; i++) {
    try {
        var actionResult = remedyForceHost.patchRemedyForceResource(resourceName, resourceData);
        return actionResult;
    } catch (errorCode) {
        System.debug("pss_RemedifyForce_UpdateRecord - Try: " + (i + 1) + " out of: 3 - Error Code: " + errorCode);
    }
}

```

Then replace all references to those vRO Actions in vRO Workflows in the “pss” folder to the improved vRO Actions. Any existing (redundant-resulting) retry logic contained in vRO Workflows is to be removed. For example:

The screenshot shows a vRO workflow diagram and its script. The diagram illustrates a process starting with a 'Waiting timer' action, followed by a 'pss_RemedifyForce_Get' action, then an 'Obtain CI ID' action, and finally a 'Set Timer' action. A red exclamation mark indicates an error or warning in the workflow. The script pane shows the code for the 'pss_RemedifyForce_Get' action, which is a PowerShell script that uses the 'System.getModule' function to call the 'pss_RemedifyForce_GetRecordIdByName' action.

```

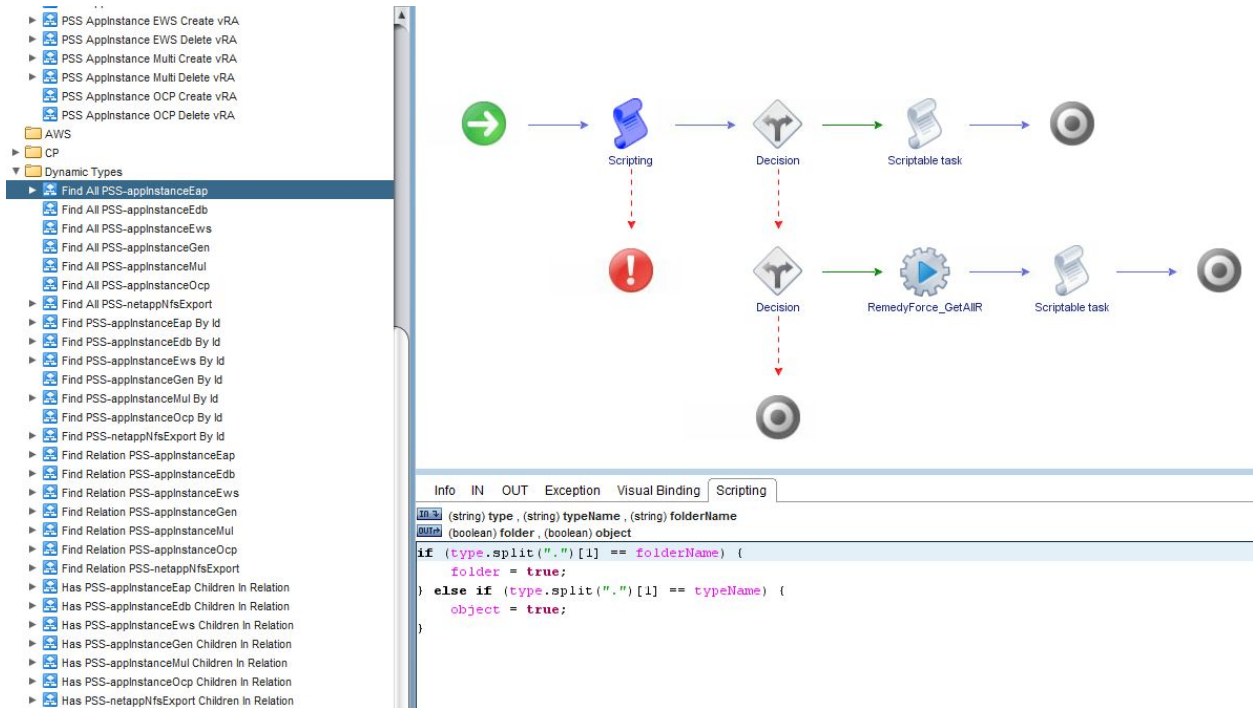
//Auto generated script, cannot be modified !
(RemedyForce.RemedyForceHost) remedyForceHost, (string) name, (string) type
(RemedyForce.RemedyForceResults) actionResult

actionResult = System.getModule("com.prudential.pss").pss_RemedifyForce_GetRecordIdByName(remedyForceHost, type, name) ;

```

Advanced - Replace each of the 4 types of vRO Workflow with a single vRO Action maintaining the same Input, Output & Functionality (this will result in much faster execution of Dynamic Types vRO Inventory population):

- Find All
- Find ... By Id
- Find Relation
- Has ... Children in Relation



In general:

- Prefix any System.debug or System.log output with the name of the vRO Workflow (aids with tracking where the code is, which has generated the log entry). Please modify where not the case.
- Remain strictly aligned with the existing prevailing naming convention & coding format (e.g. Use of upper- & lower-case, code spacing & indenting, bracketing, new-lines)
- Only vRO Workflows in the following sub-folders (within "PSS") are to be modified (others are deprecated or system-generated, etc.):

- -
 - Ansible
 -
 - Dynamic Types (only for RemedyForce vRO Action replacement)
 -
 - Nuage
 -
 - Other
 -
 - Remedy
 -
 - Support
 -

-
- Please note that Arrays are in use in various places where best practices would dictate one should use JSON or Properties. This is due to vRO's requirement in using an Array to perform Workflow execution repetition.
-
- Please make a note of any other code, which would benefit from refactoring, while keeping in mind the target audience for future Code maintenance/understanding/troubleshooting does have an Operations background (not Development background).