knowledge15®
April 19–24, 2015 • Mandalay Bay • Las Vegas, NV

Lab Guide

Discovery 301:

Optimizing Discovery

For Your Infrastructure

Doug Schulze and Aleck Lin

Default Login / Password:

admin / Knowledge15

This

Page

Intentionally

Left

Blank

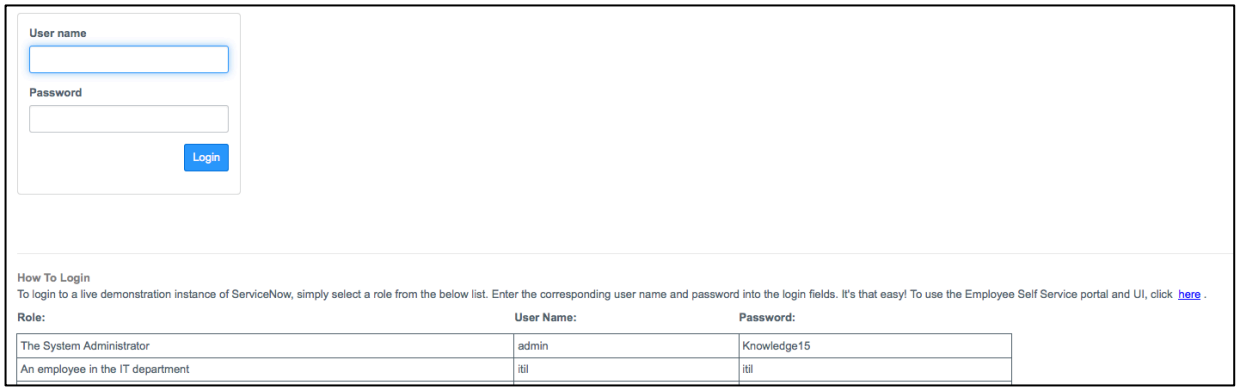
Lab Goal

This lab explains how to manage the incoming of a new CI. You add default values of anything newly created to the hardware **Status** and **Support group**.

Lab 1.1
Manage
New CIs

Log on to Your Training Instance

1. Navigate to the unique instance URL provided to you and log on as **Admin**.

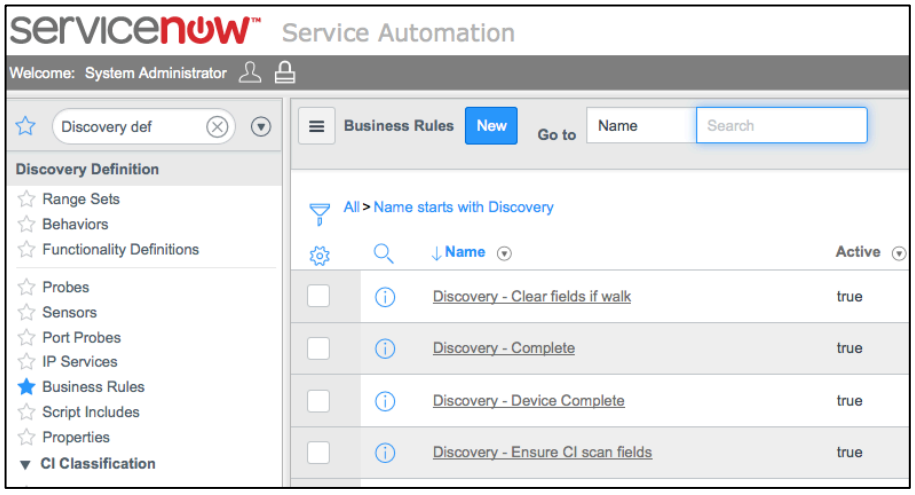


®2015 ServiceNow, Inc. All rights reserved.

Create a Business Rule to Set Default Values for Newly Created CIs

Since Discovery creates base class CIs that are extended from Hardware, set the rule there.

1. Navigate to **Discovery Definition > Business Rules**.

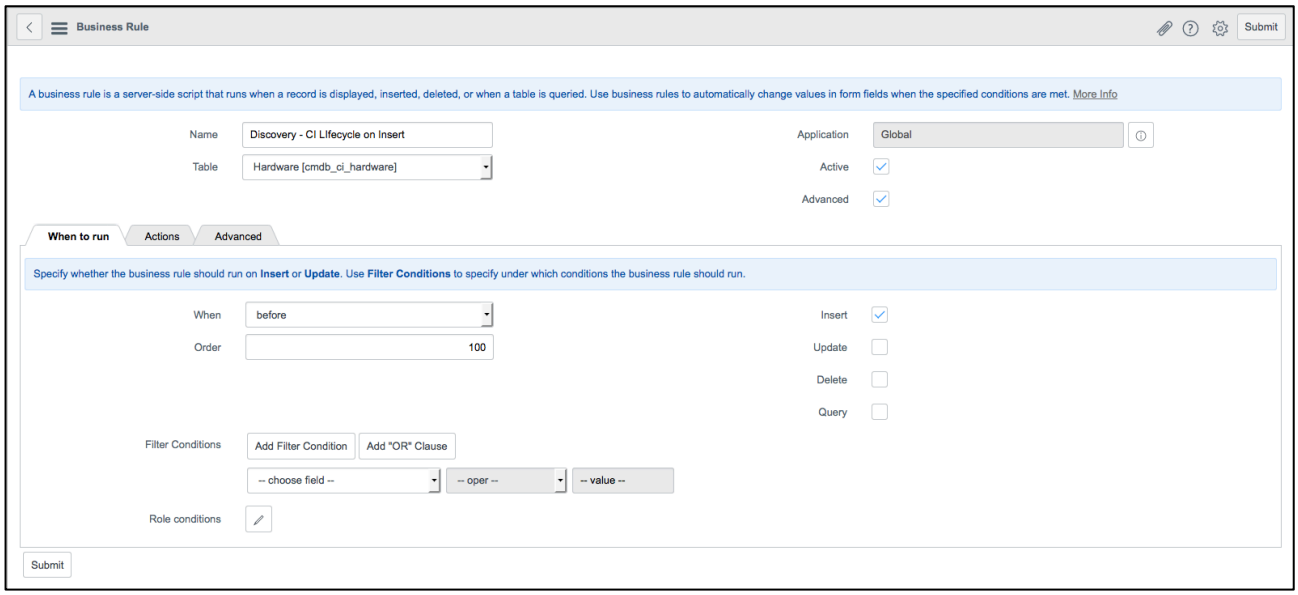


2. Select **Business Rules** and click **New**.

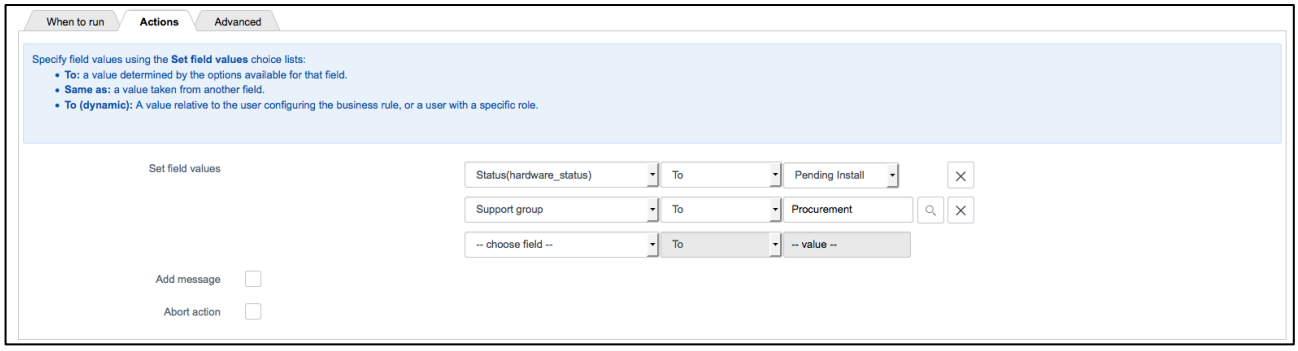
©2015 ServiceNow, Inc. All rights reserved.

3. Complete the Business Rule form to match the example.

Name: **Discovery - CI Lifecycle on insert**
Use Discovery as the prefix to align with other discovery rules.
Table: **Hardware [cmdb_ci_hardware]**
Advanced: **checked**
On **When to run** tab,
set **When** to **before** and check **Insert**



On **Actions** tab,
set field value: **Status(hardware_status) To Pending Install**
set field value: **Support group To Procurement.**

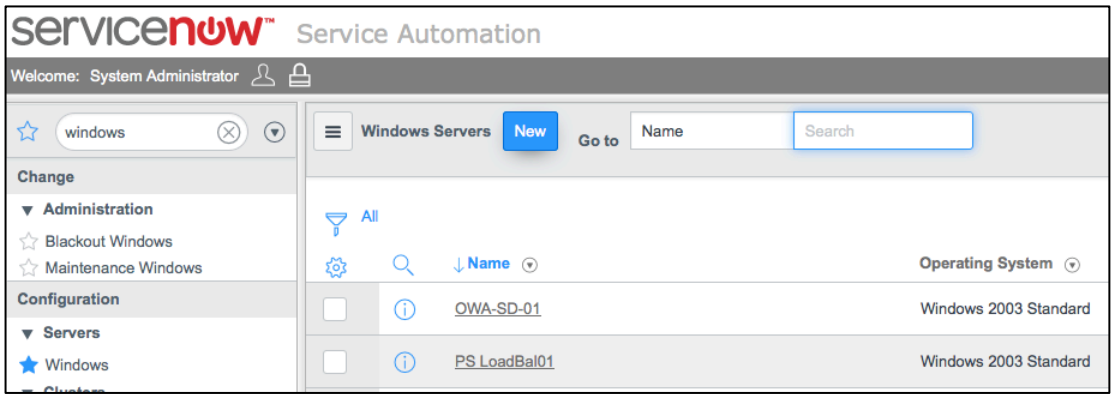


4. Select **Submit**.

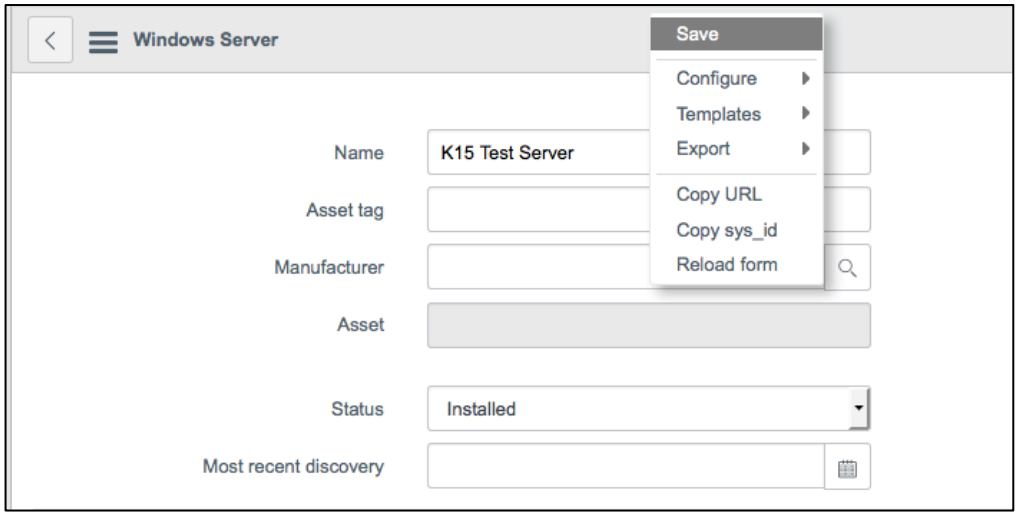
®2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Navigate to **Configuration > Servers > Windows** and select **New**.

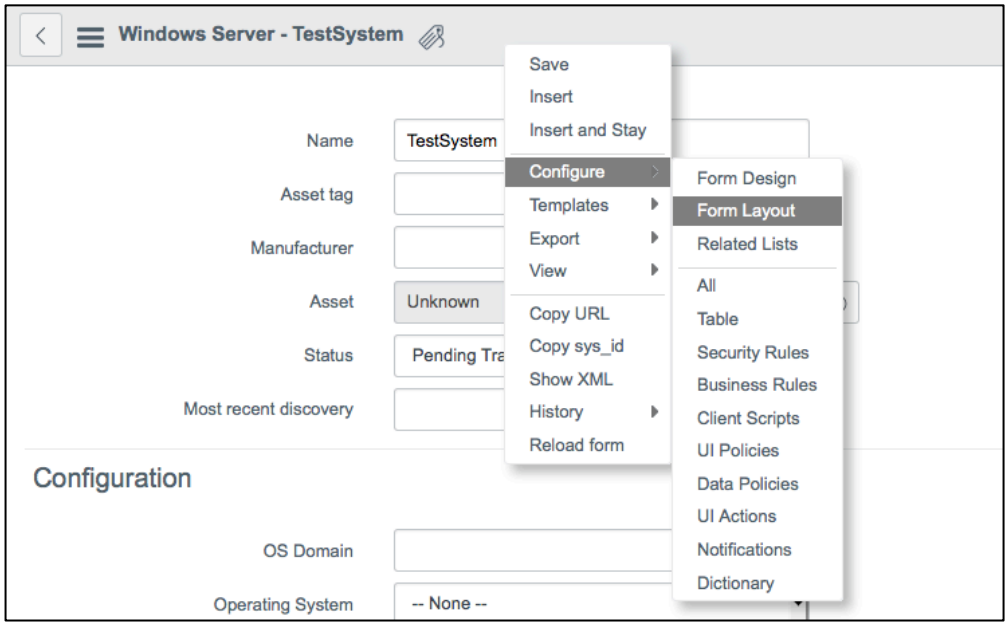


2. Enter in any name. Right-click on the header and select **Save**.

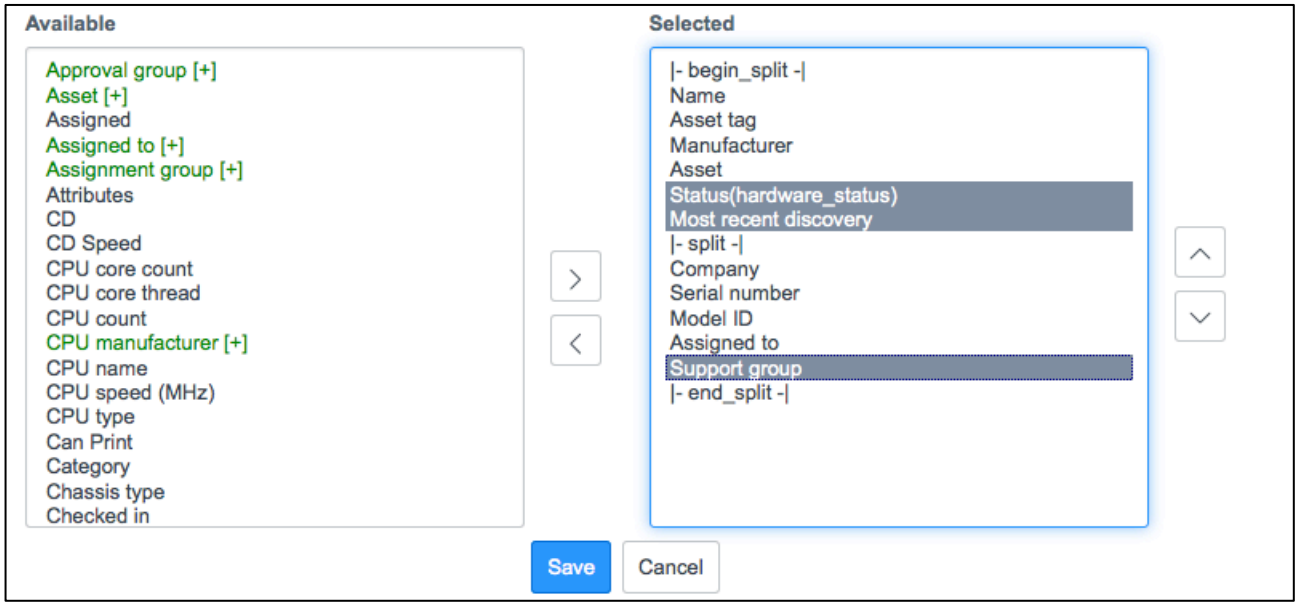


®2015 ServiceNow, Inc. All rights reserved.

3. Right-click on the header and select **Configure > Form Layout**.



4. Add the fields **Status(hardware_status)**, **Most recent discovery**, and **Support group**.



©2015 ServiceNow, Inc. All rights reserved.

5. Your result should look like the example



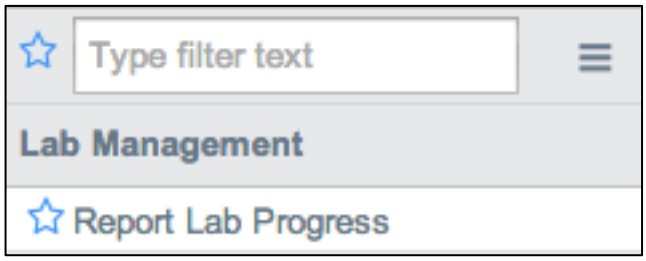
Decision Point/Think About

Remember that when a CI is created, an Asset is automatically created. Because some organizations want to have control over when Assets are created and tracked as part of their IT Asset Management (ITAM) programs, we provide a simple check box (**u_verified**) and include (**current.u_verified = 1**) in the condition script of the three out of box business rules that manage the process.

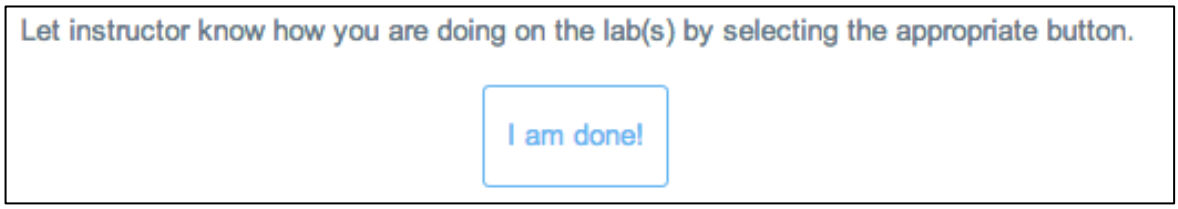
Create Asset on insert	true	Configuration Item [cmdb_ci]
Create asset on model change	true	Configuration Item [cmdb_ci]
GenerateAssets	true	Model Category [cmdb_model_category]

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.



2. Click **I am done!**



®2015 ServiceNow, Inc. All rights reserved.

Lab Goal

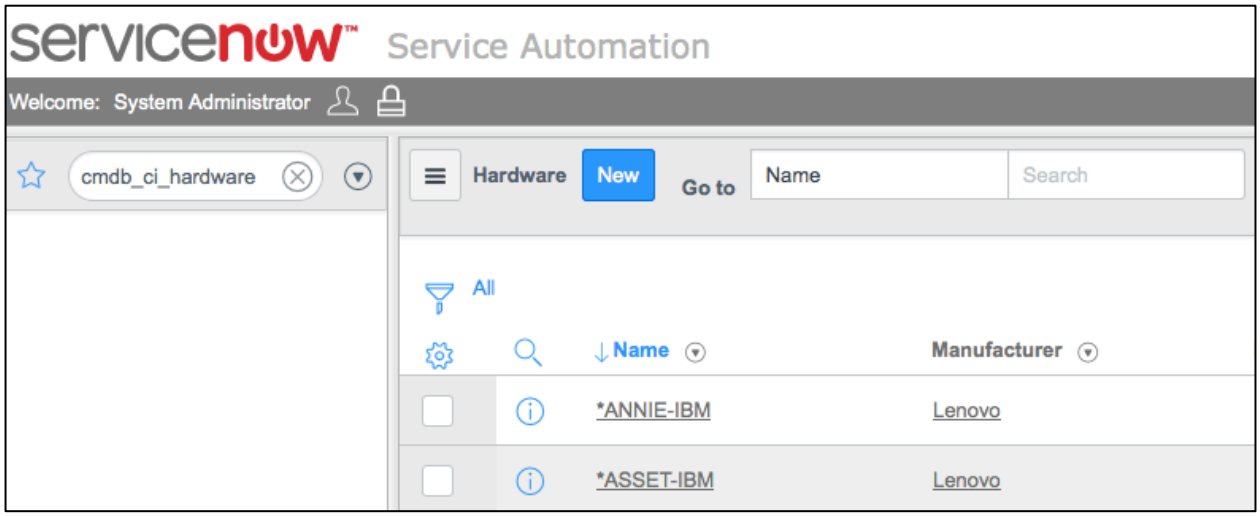
The Business Rule above created a new Hardware CI and set the state to Pending Install. The Procurement team can assign the CI to the appropriate support group through a mix of UI Policies and Client Scripts.

Lab 1.2
Ownership

Make a Support Group Mandatory

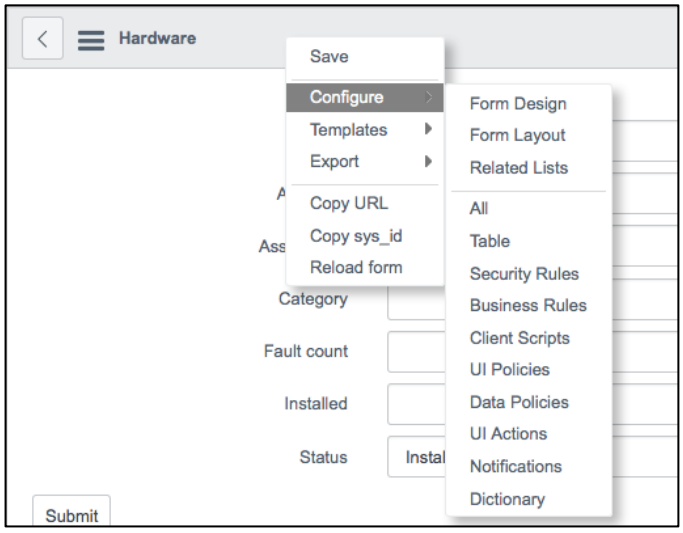
1. The Procurement group locks in a production support group when they move the hardware state to **Installed**. Do this on the base hardware table in order to apply it across all extended classes found by discovery.

2. In the application search bar, type **cmdb_ci_hardware.list**. Select **New**.

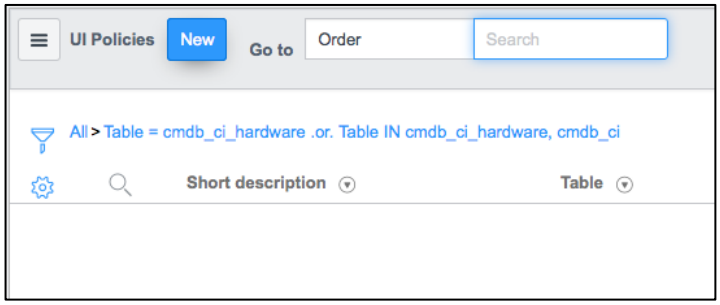


©2015 ServiceNow, Inc. All rights reserved.

3. Right-click the gray header bar next to the label **Hardware** and select **Configure > UI Policies**.



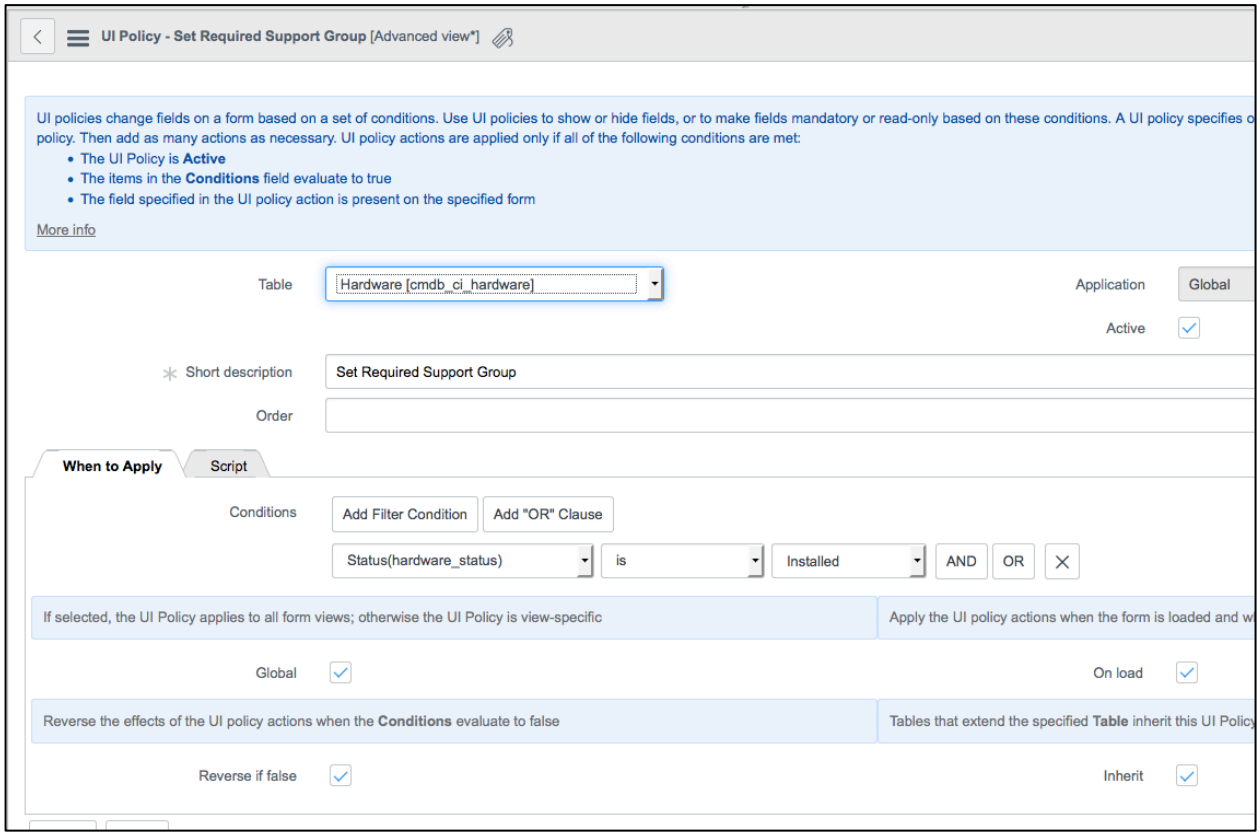
4. Select **New**.



©2015 ServiceNow, Inc. All rights reserved.

5. Define the UI policy to match the example.

   Table: **Hardware [cmdb_ci_hardware]**
   Short description: **Set Required Support Group**
   On **When to Apply** tab, set condition: **Status(hardware_status) is Installed**
   Inherit: **checked** so the UI Policy applies to extended tables



6. Right-click in the header bar and select **Save**.

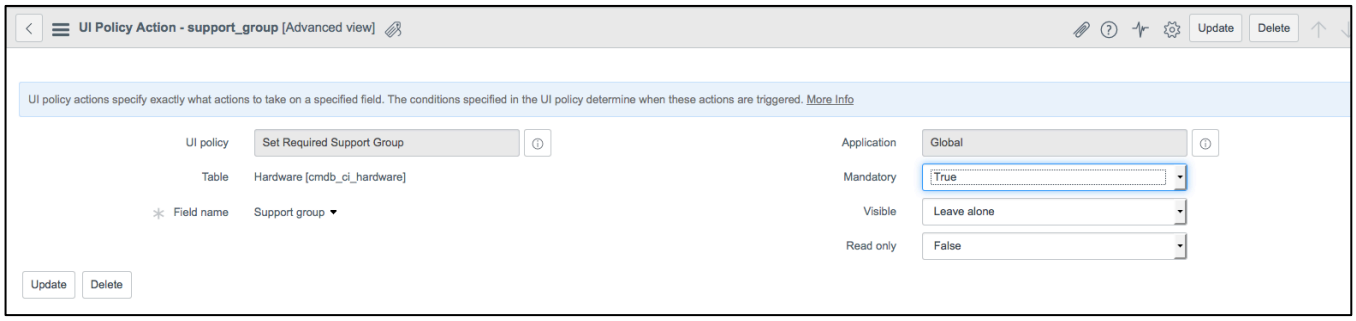


©2015 ServiceNow, Inc. All rights reserved.

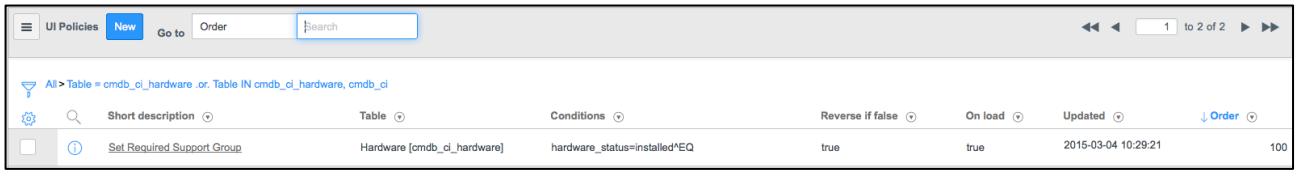
7. This generates a related list on the bottom called **UI Policy Actions.** Select **New**.



8. Set Field name to **Support group**, set Mandatory to **True**, and click **Update**.



9. Click **Update** again to return to the **UI Policies** screen.



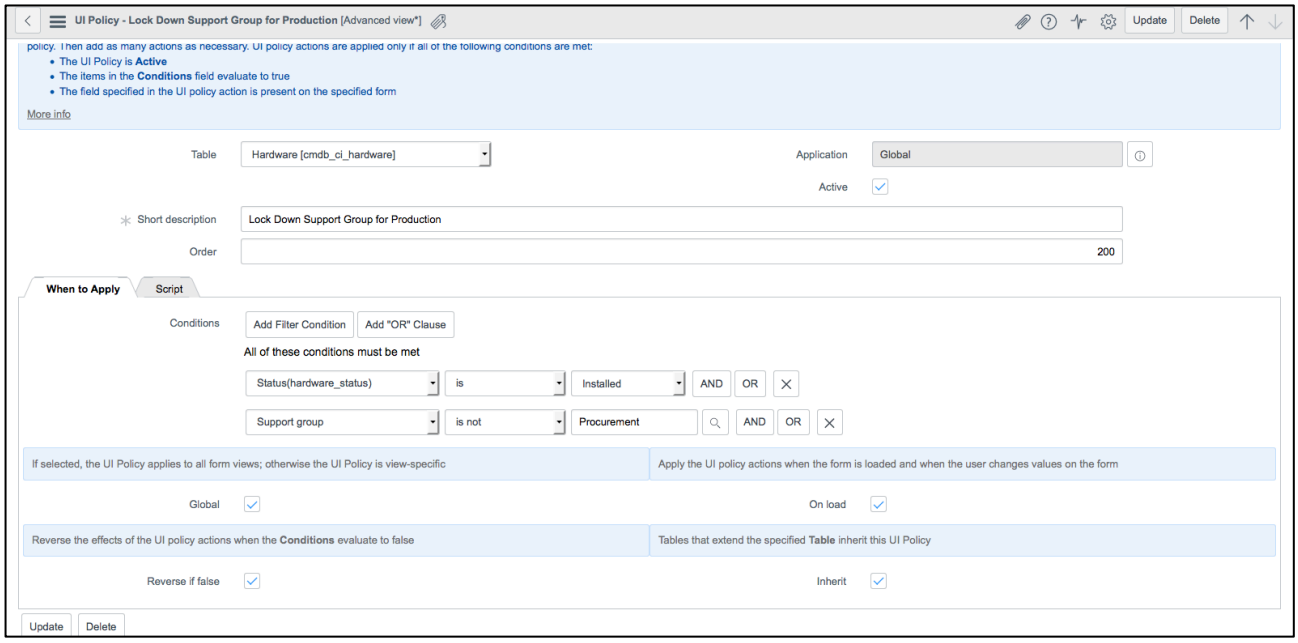
©2015 ServiceNow, Inc. All rights reserved.

Set the Support Group to Read Only when a CI is in Production

To prevent tampering of support group assignments, lock the support group field when a CI is **Installed**.

1. Create a **New** UI Policy to match the example.

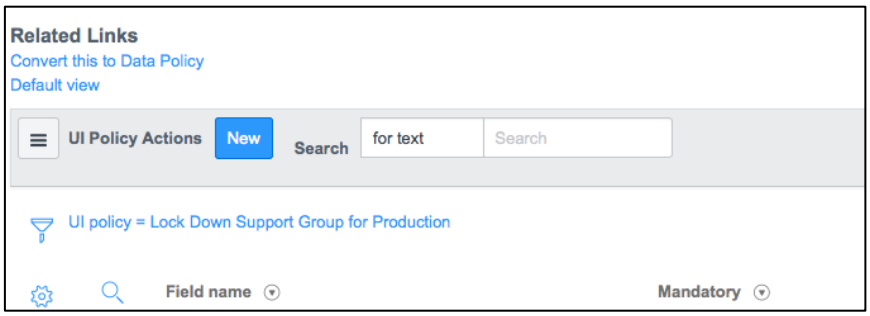
   Table: **Hardware [cmdb_ci_hardware]**
   Short description: **Lock Down Support Group for Production**
   Order: **200**
   On **When to Apply** tab,
   set condition: **Status(hardware _status) is Installed**
   set condition: **Support group is not Procurement**
   Inherit: **checked** so the UI Policy applies to extended tables



2. Right-click in the header bar and select **Save**.

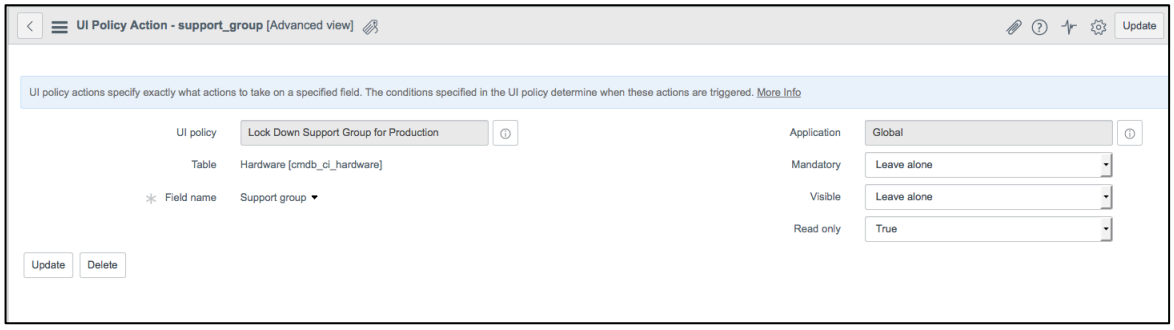
®2015 ServiceNow, Inc. All rights reserved.

3. Select **New** UI Policy Action.



4. Complete the UI Policy Action form to match the example.

   Field name: **Support Group**
   Ready only: **True**



5. Select **Update**.

Lab Success Verification

1. Navigate to **Configuration > Servers > Windows** and select **New**.

2. Enter in any name and set Status to **Installed**. Right-click on the header and select **Save**.

3. Verify **Support group** is read only.



®2015 ServiceNow, Inc. All rights reserved.

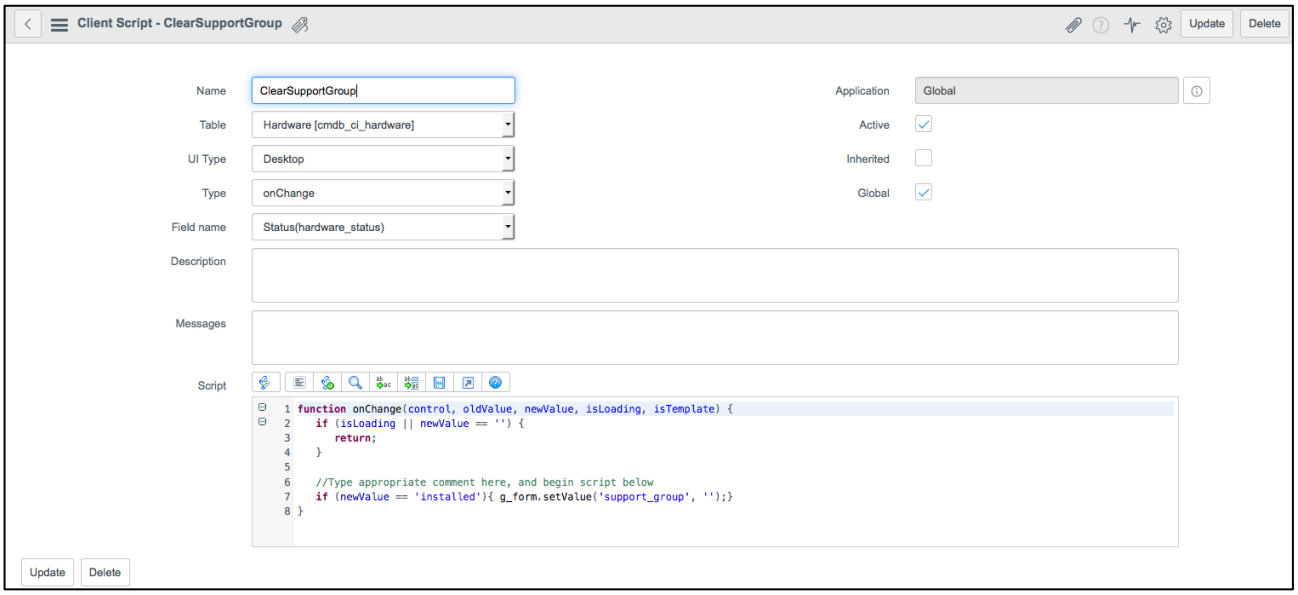
Decision Point/Think About

You build things as fault-tolerant as possible and think of errors before people make them. As you see when we move the state to installed, **support_group** became required. But what if that person forgets to change the value? You certainly do not want Procurement responsible for production devices. So why not clear out that value when in a fresh installed state!

This can be addressed with a client script on the hardware table.

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || newValue == '') {
        return;
    }

    //Type appropriate comment here, and begin script below
    if (newValue == 'installed'){g_form.setValue('support_group', '');}
```



Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.

2. Click **I am done!**

©2015 ServiceNow, Inc. All rights reserved.

Lab Goal

There is now a good process for setting up new CIs that are discovered and a solid method for allowing Procurement to assign responsible parties, while keeping them honest at the same time. This lab focuses on what happens to those CIs as they live in an environment. What happens when they disappear? What happens when they have not been discovered in the past 30 days?

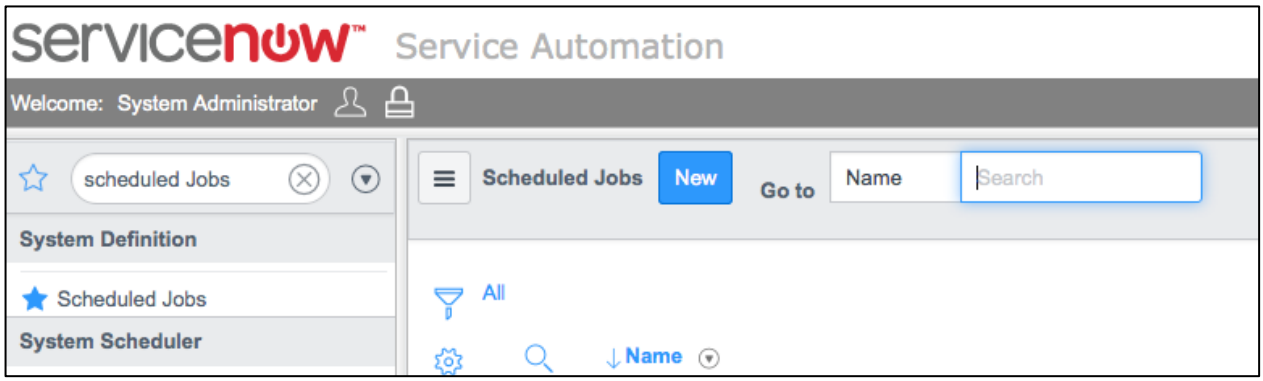
Lab 2.1
Life as a CI

To identify the lost CIs, measure a field called **Most Recent Discovery**. This attribute is set whenever discovery identifies and updates the record. You might be thinking, why not use the **Updated** attribute? Although discovery would set that value as well if anyone did anything to that record such as add notes, that updated field would be refreshed while **Most Recent Discovery** retains the same last Discovered Date. To do this, you need a schedule Job to check that date and set the state accordingly.

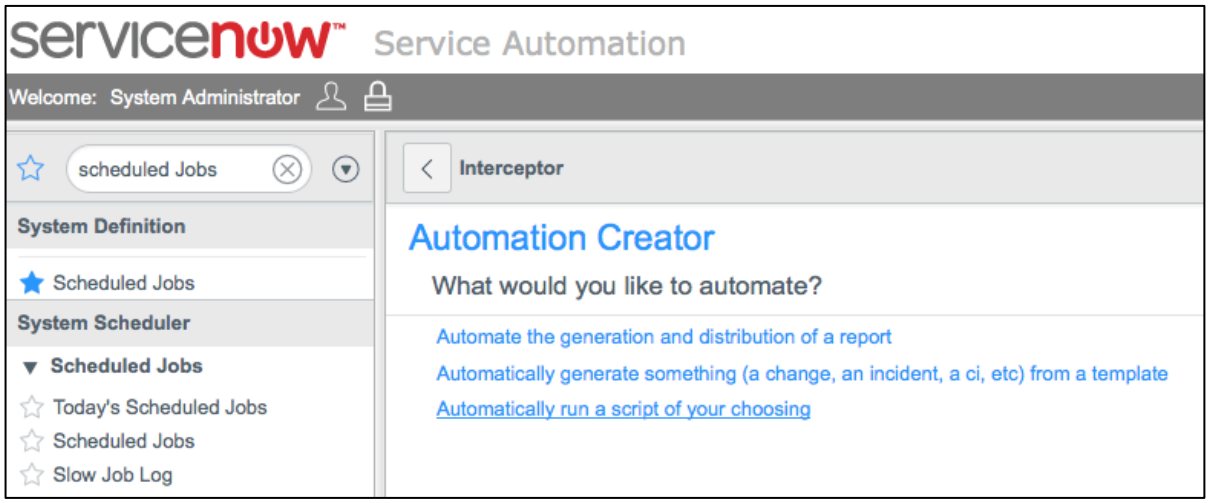
®2015 ServiceNow, Inc. All rights reserved.

Identify Lost CIs

1. Navigate to **System Definition > Scheduled Jobs** and select **New**.



2. Choose **Automatically run a script of your choosing**.



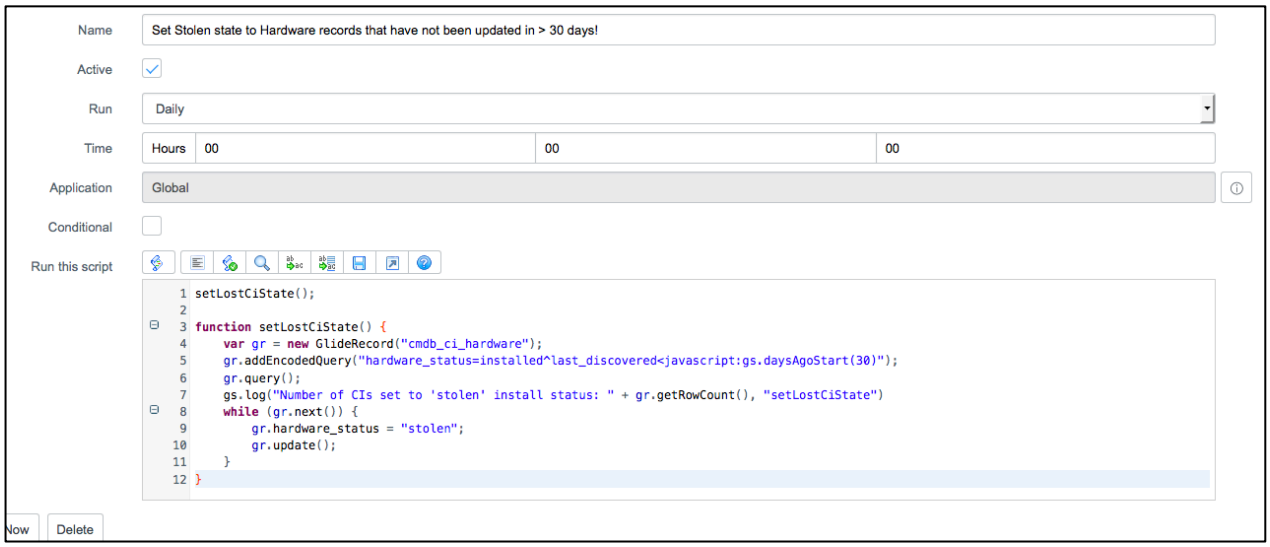
©2015 ServiceNow, Inc. All rights reserved.

3. Complete the form to match the example. This script says – if the hardware state is installed and it has been greater than 30 days since the most recent discovery (**last_discovered**) then set the state to stolen.

   Name: **Set Stolen state to Hardware records that have not been updated in > 30 days!**
   Leave default run time of daily at midnight.
   Script:

```
setLostCiState();

function setLostCiState() {
    var gr = new GlideRecord("cmdb_ci_hardware");
    gr.addEncodedQuery("hardware_status=installed^last_discovered<javascript:gs.daysAgoStart(30)");
    gr.query();
    gs.log("Number of CIs set to 'stolen' install status: " + gr.getRowCount(), "setLostCiState")
    while (gr.next()) {
        gr.hardware_status = "stolen";
        gr.update();
    }
}
```

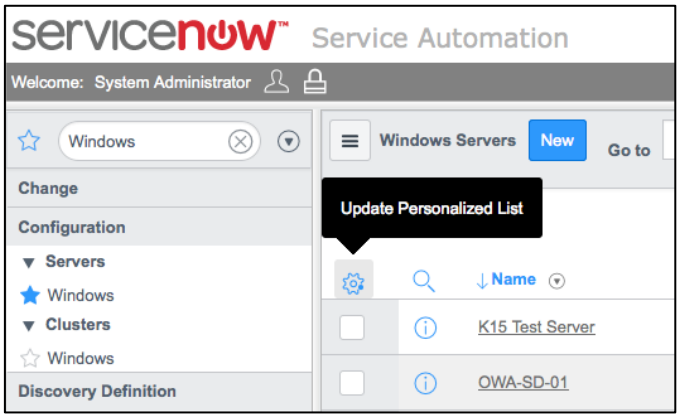


4. Click **Submit**.

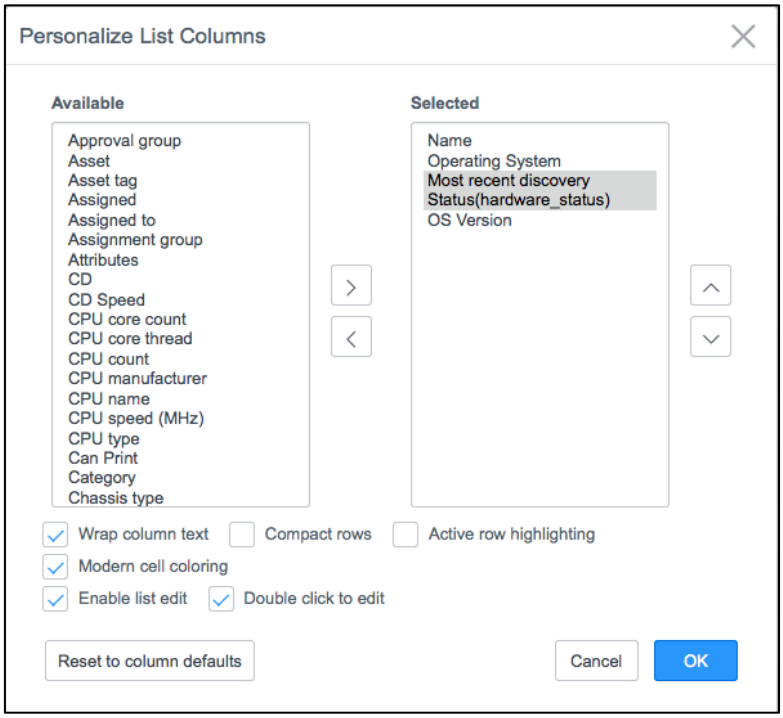
®2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Navigate to **Configuration > Servers > Windows**.



2. Modify the view to include **Most recent discovery** and **Status(hardware_status)**.

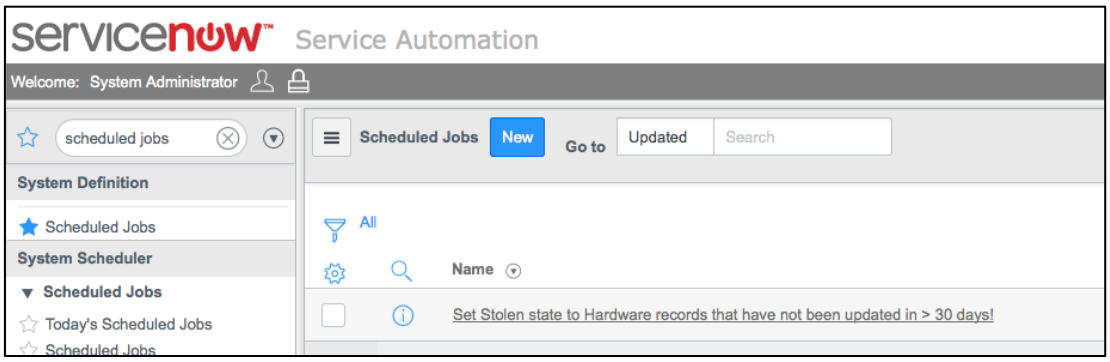


3. There is one out of box candidate for a change.

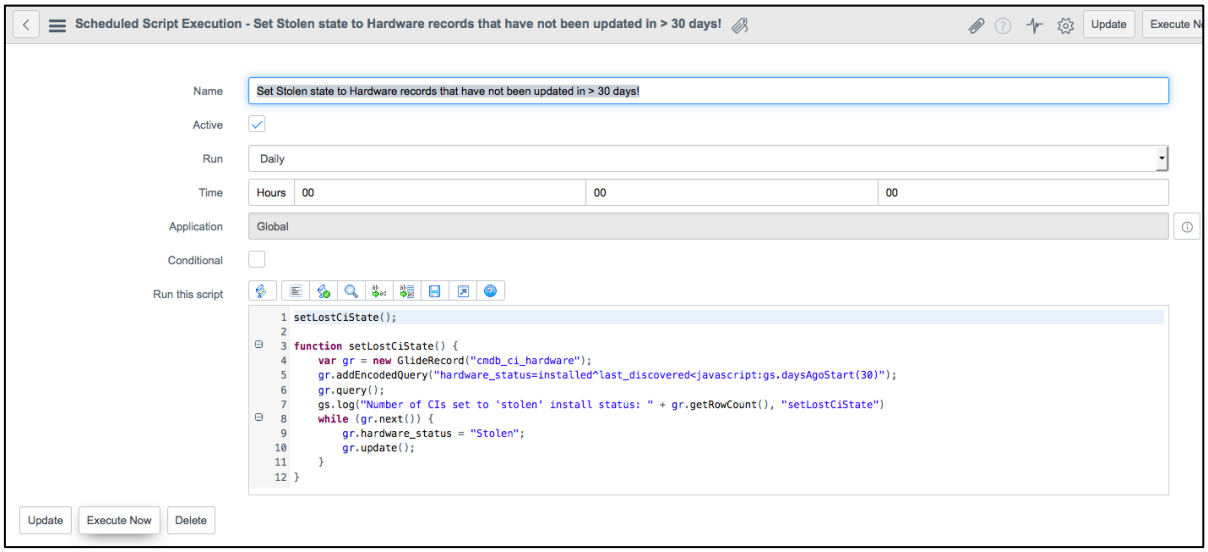


©2015 ServiceNow, Inc. All rights reserved.

4. Return to **System Definition > Scheduled Jobs**.



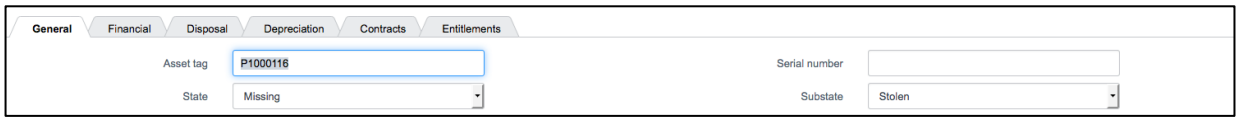
5. Select the job you created. At the bottom, click **Execute Now**.



6. Return to **Windows Servers** and to see the CI that was created:



And the asset:



©2015 ServiceNow, Inc. All rights reserved.

Decision Point/Think About

We used the **stolen** state to make the lab easier to configure. It is an extreme assessment just because something has not been discovered in 30 days. Many situations could have taken place, such as an extended vacation or a forgetful admin that did not fill out the change order to let people know their project had ended and the systems were decommissioned. Consider using custom terms for device status such as **Unknown** until a proper disposition can be determined.

Also what should you do when a device goes missing? Perhaps send an e-mail notification to the support group or create an incident assigned to the support group looking for information about the CI.

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.
2. Click **I am done!**

®2015 ServiceNow, Inc. All rights reserved.

Lab Goal

So we did a good job managing those Hardware CIs that seem to have fallen off the network, but what about those applications that are also created. This lab explains how to set states to those if they disappear as well.

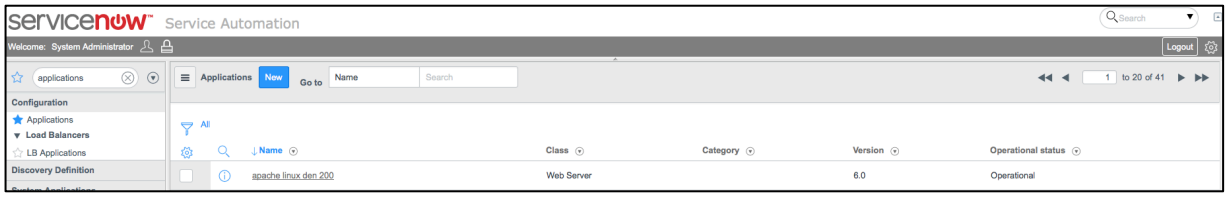
It is important to remember that a running application does not have an associated asset. Unlike installed software, it is not something you "own"; it is just something that provides capabilities to a business service.

Lab 2.2
Life as an App CI

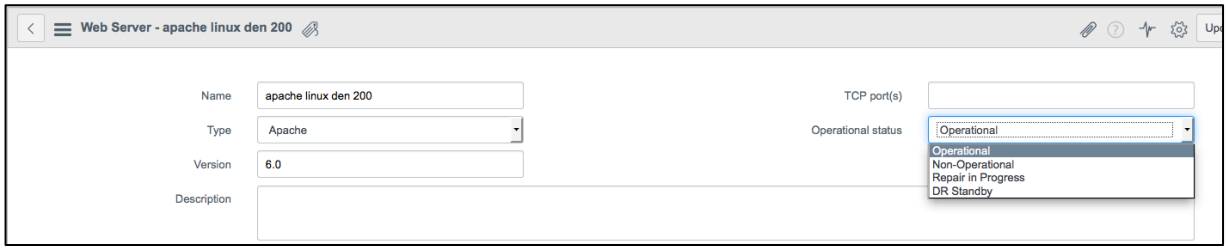
Set CI Status in the Application Table (cmdb_ci_appl)

The process is the same as setting the hardware state after "**X**" number of days, through a scheduled job. Identify the current state of the applications and look for a field that is out-of-box for this lab.

1. Navigate to **Configuration > Applications** and open **apache linux den 200** or other CI.



2. The out-of-box **Operational** status can be used for this exercise.



3. Navigate to **System Definition > Scheduled Jobs** and select **New**.

4. Choose **Automatically run a Script of your choosing**.

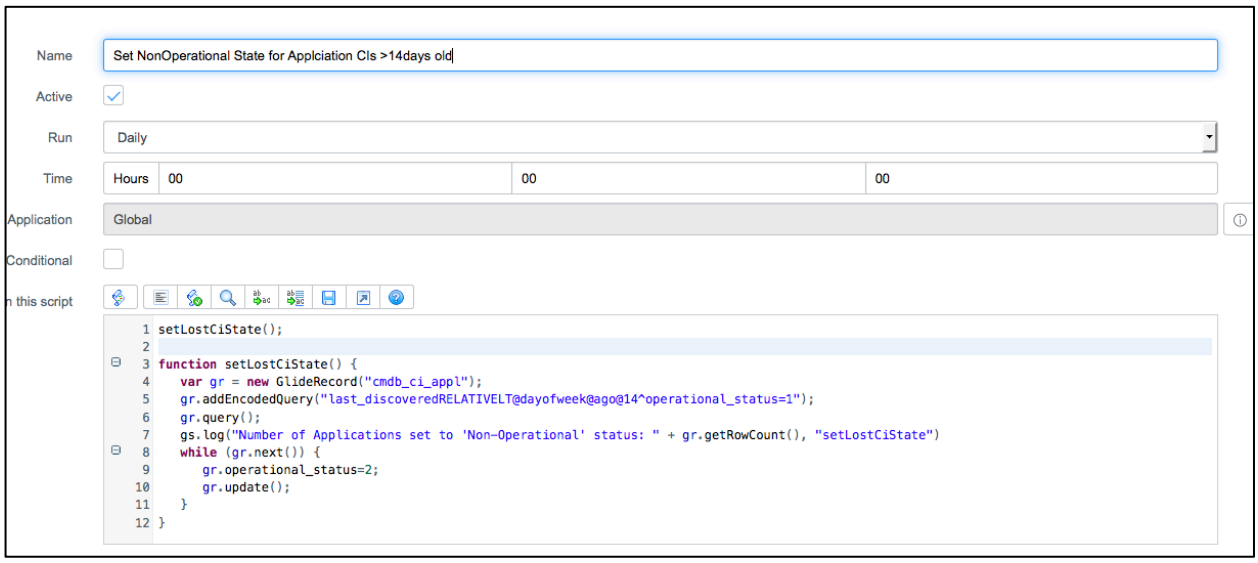
©2015 ServiceNow, Inc. All rights reserved.

5. Complete the form to match the example. This script looks through the application table (**cmdb_ci_appl**) to find records for which the last discovered date is greater than 14 days ago and in an Operational state. The script sets the nonoperational state when found.

   Name: **Set NonOperational State for Application CIs > 14 days old**
   Leave default run time of daily at midnight.
   Script:

```
setLostCiState();

function setLostCiState() {
   var gr = new GlideRecord("cmdb_ci_appl");
   gr.addEncodedQuery("last_discoveredRELATIVELT@dayofweek@ago@14^operational_status=1");
   gr.query();
   gs.log("Number of Applications set to 'Non-Operational' status: " + gr.getRowCount(), "setLostCiState")
   while (gr.next()) {
      gr.operational_status=2;
      gr.update();
   }
}
```



6. Click **Submit**.

7. Click **Execute now**.

8. Go to the **Applications** table to see which applications were set to **Non-Operational**.



©2015 ServiceNow, Inc. All rights reserved.

Decision Point/Think About

What about just removing the Application CIs that have not shown themselves in a while? They are not an asset and they spin up and down all the time! Maybe there is no need to keep the clutter and your process can determine that it is ok to just remove them after, say, 30 days?

So it is just another scheduled job that looks through the Application table, and removing the record

```
var myCMDBApps = new GlideRecord ('cmdb_ci_appl');
myCMDBApps.addEncodedQuery('operational_status=2^last_discoveredRELATIVELE@dayofweek@ago@30');
myCMDBApps.query();
gs.log("The number of rows returned are " + myCMDBApps.getRowCount(), "test");

while (myCMDBApps.next()) {
gs.print('Application not discovered for morethan 30 days ' + myCMDBApps.name + ' deleted');
myCMDBApps.deleteRecord();
}
```

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.
2. Click **I am done!**

®2015 ServiceNow, Inc. All rights reserved.

Lab Goal

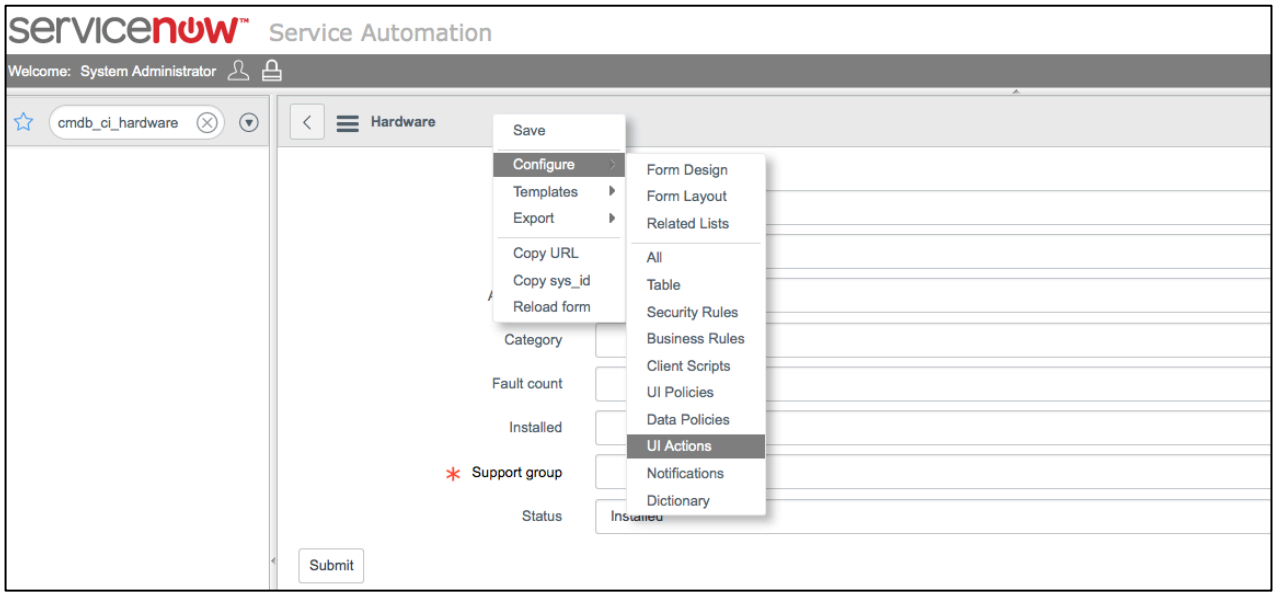
Life has been great having the CIs around, but now it is time to say goodbye (queue Bocelli's "Time to Say Goodbye," or even better, the Will Ferrell version from Step Brothers).

This has to be the biggest issue - what to do with these records. There are needs across the spectrum ranging from just deleting them to leaving them for audit purposes. This lab explains how to let the platform help you with either case, while cleaning up supporting records that are just clutter in the CMDB.

Lab 3
Time to Say Goodbye

Create UI Action to Remove Related Records when a CI is Retired

1. In the Application Search Bar, type **cmdb_ci_hadware** to quickly return to the **Hardware Table.**
2. Select **New**. Right-click the header bar and select **Configure > UI Actions**.

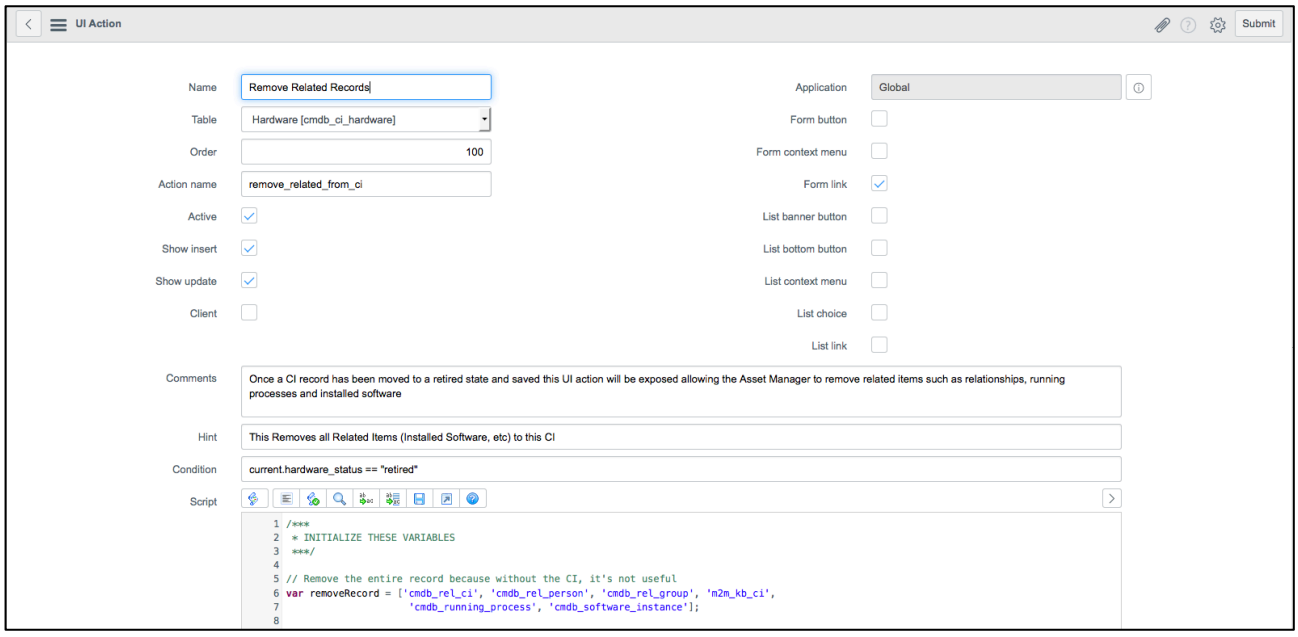


3. Select **New.**

®2015 ServiceNow, Inc. All rights reserved.

4. Complete the form to match the example.

   Name: **Remove Related Records**
   Table: **Hardware (cmdb_ci_hardware)**
   Action Name: **remove_related_from_ci**
   Form link: **Checked**
   Enter **Comments** and a **Hint** as you see fit.
   Condition: **current.hardware_status == "retired"**. Ensures link is only visible if the CI is retired.



Script:

```
/***
 * INITIALIZE THESE VARIABLES
 ***/

// Remove the entire record because without the CI, it's not useful
var removeRecord = ['cmdb_rel_ci', 'cmdb_rel_person', 'cmdb_rel_group', 'm2m_kb_ci',
                                  'cmdb_running_process', 'cmdb_software_instance'];

// By default, we ignore task tables. These too are ignored if necessary.
var knownExclude = [];

/************ DO NOT MODIFY BELOW THIS LINE *************/

removeRelated(current.getValue('sys_id'), current.getTableName());
action.setRedirectURL(current);

/*
 * removeRelated - look through all non-task tables and remove
 *                 any references to this CI
 *                 If the CI appears in a known many-to-many table, the record is removed
 *                 If the CI appears is an known exclusion (like ecc_queue) it is skipped
```

©2015 ServiceNow, Inc. All rights reserved.

```
 *
 * @param ciId - sys_id of the CI we're looking to remove
 * @param tableName - name of the table that could be referenced (or one of its parents)
 * @return none
 *
 */

 // Sample
internal_type=reference^reference=cmdb_ci^ORreference=cmdb_ci_hardware^ORreference=cmdb_ci_netgear^name!
=change_request_imac^name!=task
function removeRelated(ciId, tableName) {

    // get a list of tables that are parent class to this one
    var tu1 = new TableUtils(tableName);
    var parentList= j2js(tu1.getTables());
    var parentQueryList = [];

    // gs.print('parentList(' + parentList.length + '): ' + parentList);
    // Build an encoded query string
    for (var i = 0; i < parentList.length; i++)
            parentQueryList.push('reference=' + parentList[i]);

    var parentQuery = parentQueryList.join('^OR');
    // gs.print('parentQuery=' + parentQuery);

    // get a list of all task tables (we don't want those)
    var tu2 = new TableUtils('task');
    var taskList = j2js(tu2.getTableExtensions());
    taskList.push('task'); // This isn't normally included
    var taskQueryList = [];

    //gs.print('taskList(' + taskList.length + '): ' + taskList);
    // Build an encoded query string
    for (var i = 0; i < taskList.length; i++)
            taskQueryList.push('name!=' + taskList[i]);

    var taskQuery = taskQueryList.join('^');
    //gs.print('taskQuery=' + taskQuery);

    var dGr = new GlideRecord('sys_dictionary');

    dGr.addQuery('internal_type', 'reference');
    dGr.addEncodedQuery(parentQuery);
    dGr.addEncodedQuery(taskQuery);
    dGr.orderBy('name');
    dGr.query();

    // gs.print('count=' + dGr.getRowCount());

    while (dGr.next()) {

            var au = new ArrayUtil();

            if (au.contains(knownExclude, dGr.getValue('name'))) {
                    gs.print('Ignoring: ' + dGr.getValue('name') + ' - forced exclude');
                    continue;
            }

            // Check if the table (name) has a field that references task (e.g. task_ci), if so, do not
clear it
            if (tableHasTaskReference(dGr.getValue('name'), taskList)) {
                    gs.print('Ignoring: ' + dGr.getValue('name') + ' - contains reference to task');
                    continue;
            }
```

®2015 ServiceNow, Inc. All rights reserved.

```
            gs.print('name=' + dGr.getValue('name') + ' element=' + dGr.getValue('element') + '
references=' + dGr.getValue('reference') + ' ciId=' + ciId);

            // Now query the known table/field to see if they reference the current CI
            var gr = new GlideRecord(dGr.getValue('name'));

            gr.addQuery(dGr.getValue('element'), ciId); // eg. cmdb_alias.cmdb_ci=your sys_id
            gr.query();

            while (gr.next()) {

                    // If this is a known M2M table, remove the whole record
                    if (au.contains(removeRecord, dGr.getValue('name'))) {
                            gr.deleteRecord();
                            gs.print('---Removing M2M record ' + dGr.getValue('name') + ' (' +
gr.getValue('sys_id') + ')');
                    } else {
                            // Just blank out the CI field
                            gr.setValue(dGr.getValue('element'), '');
                            gr.update();
                            gs.print('...Resetting ' + dGr.getValue('name') + '.' +
dGr.getValue('element') + ' from: ' + gr.getValue('sys_id'));
                    }
            }
    }
}

/*
 * tableHasTaskReference - determine if a given table has one or more fields that reference something in
the task hierarchy
 *                         This is to prevent m2m tables like task_ci from having a record w/o a CI
(functionally useless)
 *
 * @param tableName - name of the table we're looking at (e.g. task_ci)
 * @param taskList - list of task tables that might also be referencing this
 * @return boolean - true=one or more fields references a task
 *
 */
function tableHasTaskReference(tableName, taskList) {

    var taskQueryList = [];
    var count = 0;

    // Build an encoded query string
    for (var i = 0; i < taskList.length; i++)
            taskQueryList.push('reference=' + taskList[i]);

    var taskQuery = taskQueryList.join('^OR');

    var dict = new GlideAggregate('sys_dictionary');

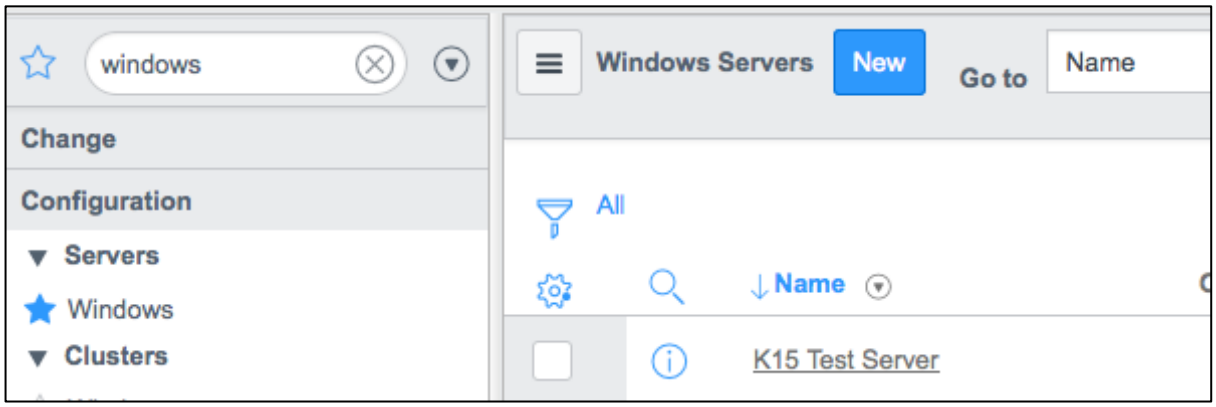
    dict.addAggregate('COUNT');
    dict.addQuery('name', tableName);
    dict.addQuery('internal_type', 'reference');
    dict.addEncodedQuery(taskQuery);

    dict.query();
    if (dict.next())
            count = dict.getAggregate('COUNT');

    return count != 0;
}
```

®2015 ServiceNow, Inc. All rights reserved.

5. Now see the magic! Return to the test windows system.



6. Set the **State** to **Retired** and **Save** the record. The **UI Action** becomes visible.



©2015 ServiceNow, Inc. All rights reserved.

Decision Points/Think About

The biggest value in this process is removing **installed software**. Following a proper ITAM process when a *physical* computer is retired, you now no longer have to deal with orphaned instances of Visio or Adobe in your CMDB when they are no longer installed. Why pay for something that is not deployed? This also cleans up other CIs that are not providing any additional value such as the running process. However, there are other important points to consider when dealing with retired CIs.

1. Would you rather just delete Virtual Machines if they are not needed for auditing or regulatory purposes?

2. What about setting business rules that alert if a CI is being rediscovered and its state is retired? Should a proper change be initiated that would update the state to something more appropriate?

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.

2. Click **I am done!**

®2015 ServiceNow, Inc. All rights reserved.

Lab Goal

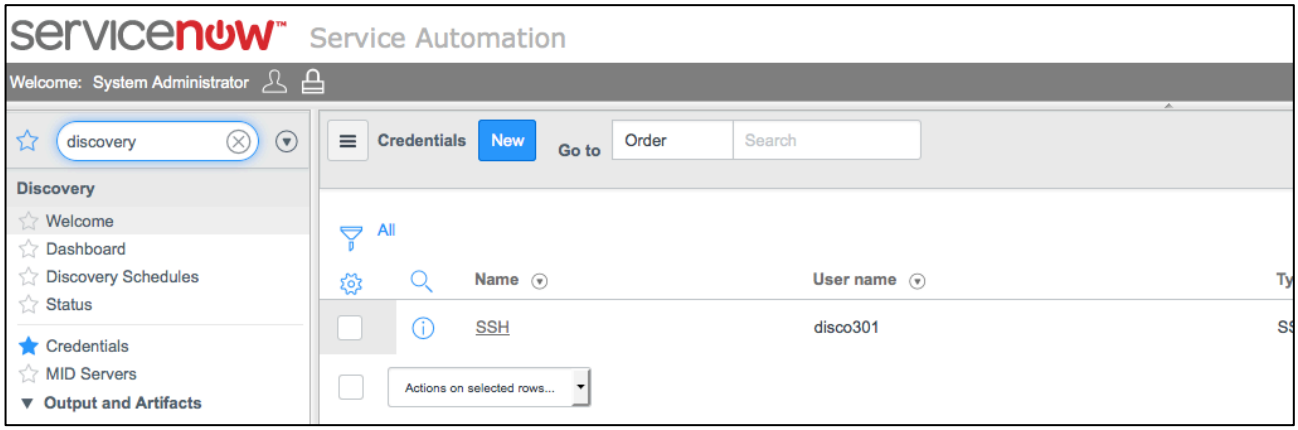
The previous labs looked at how individual parts of Discovery work. This lab explains how all the parts work together against an actual system that you discover that changes states as it goes through the lifecycle.

Lab 4
Discover the MID Server

Discover the MID Server

1. Enter **SSH** credentials

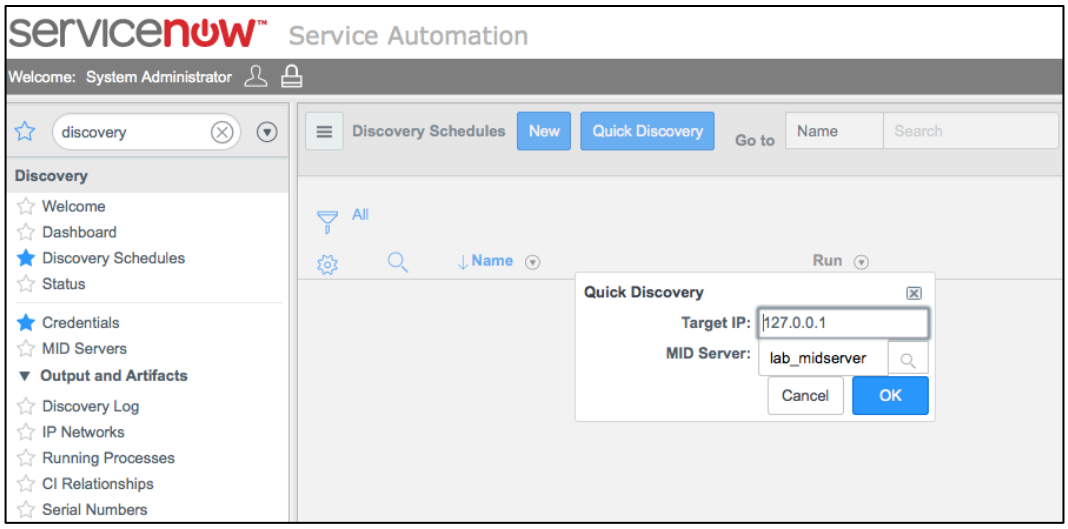
   user: **disco301**
   pass: **password**



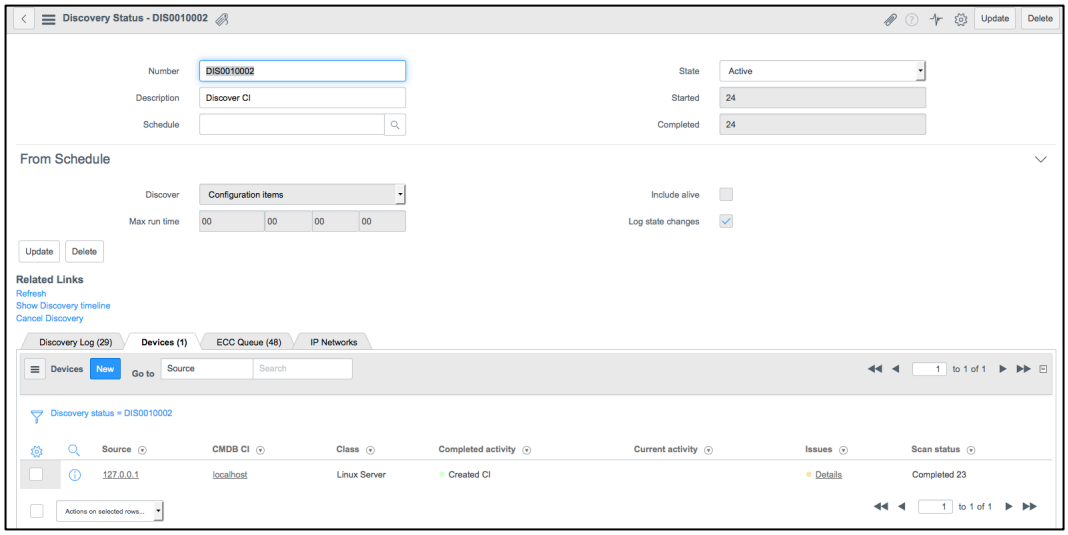
©2015 ServiceNow, Inc. All rights reserved.

2. Select **Discovery Schedules** then **Quick Discovery**. Set the information below and click **OK**.

   Target IP: **127.0.0.1**
   MID Server: **lab_midserver**

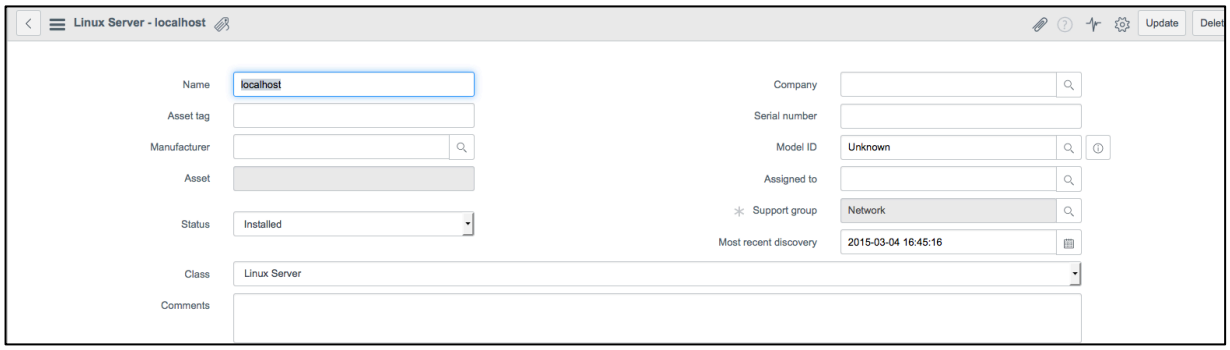


3. Review the **Discovery Status** Record and select the **localhost** CI.



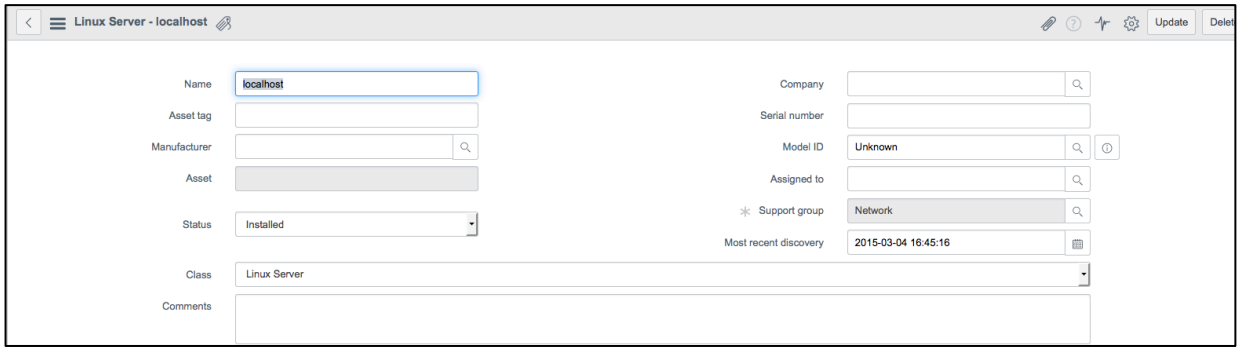
®2015 ServiceNow, Inc. All rights reserved.

4. Add fields to the Linux Server form: **Status(hardware_status)**, **Support Group**, and **Most Recent Discovery**.



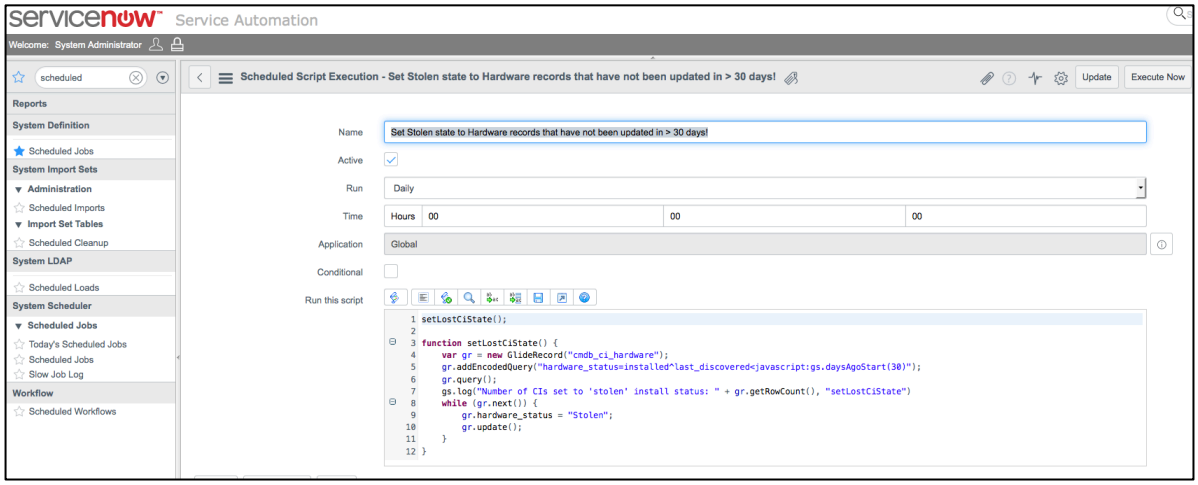
5. Change the **Status** field states to see the policies in effect.

6. Change the **Most Recent Discovery** date to something greater than 30 days ago. **Save** the record.

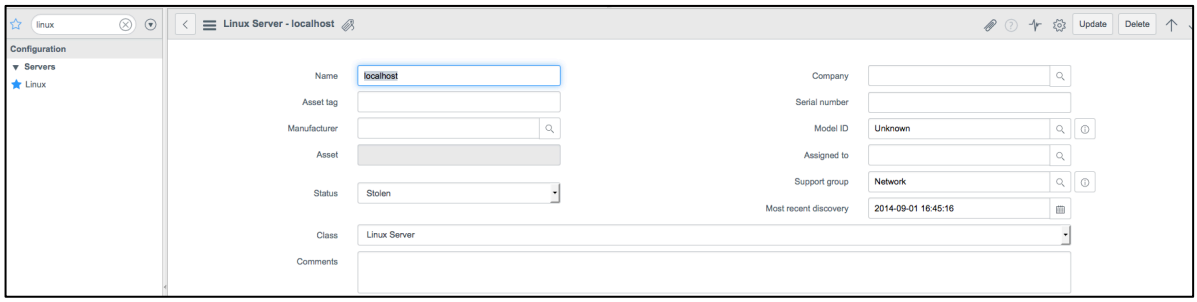


©2015 ServiceNow, Inc. All rights reserved.

7. Go to **Scheduled Jobs**. Choose and execute your scheduled job that sets the **stolen** state.



8. Return to the **Linux Server**. Choose **localhost** and verify that the item is in a **Stolen** state.

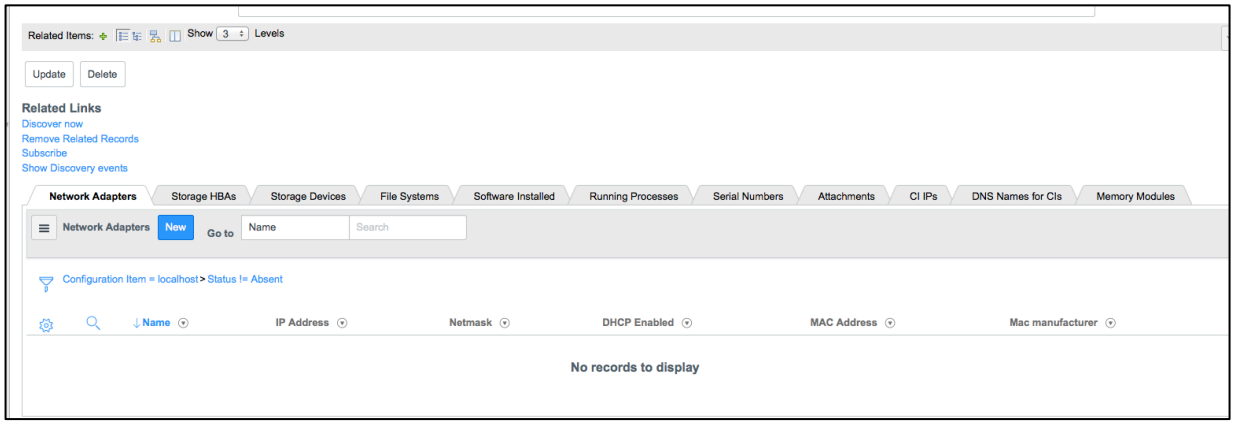


9. Change the State to **Retired. Save** the record.

10. Click the **Remove Related records** UI Action.

©2015 ServiceNow, Inc. All rights reserved.

11. Marvel at the beauty of no installed software or related items for retired systems.



Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.

2. Click **I am done!**

®2015 ServiceNow, Inc. All rights reserved.

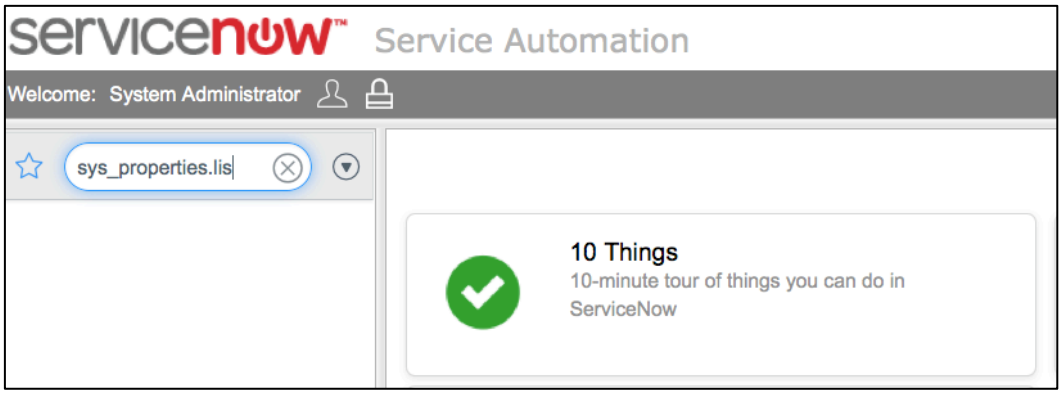
Lab Goal

This lab explains how to set a custom value so that a particular probe in the classifier gets triggered when the condition is met.

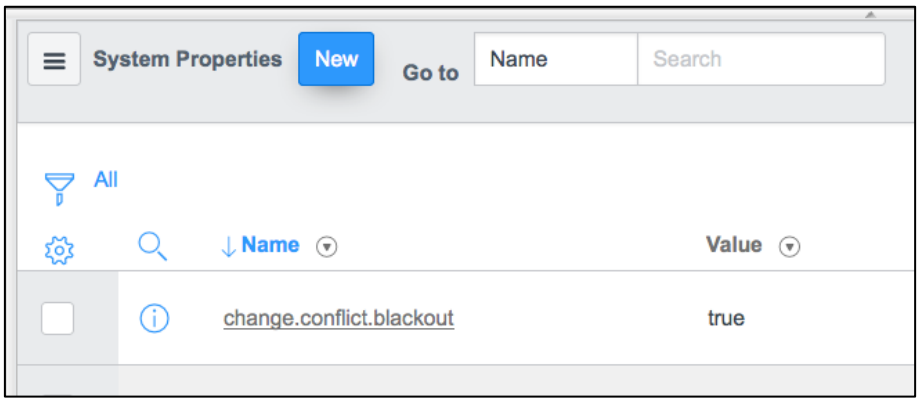
Lab 5
Create Custom Values

Create Additional Debug Logging

1. In the application search bar type in **sys_properties.list** to generate additional log messages to identify key pieces of processing data that you want to manipulate.



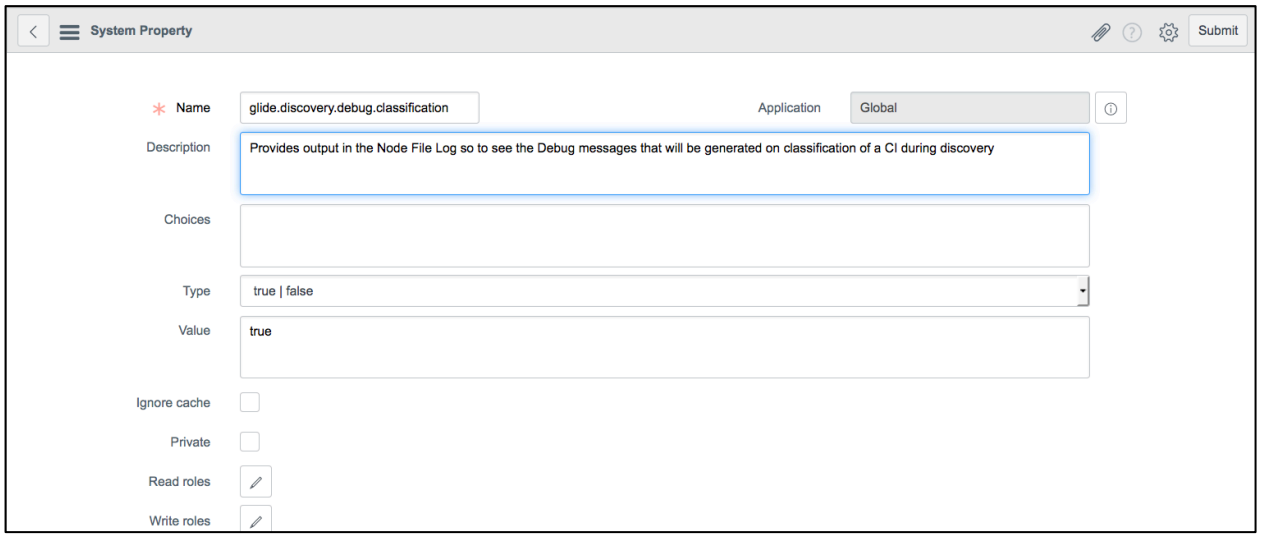
2. This brings up the list of all available properties. Click **New**.



©2015 ServiceNow, Inc. All rights reserved.

3. Enter the following:

   Name: **glide.discovery.debug.classification**
   Description: **(optional for the lab)**
   Type: **true | false**
   Value: **true**



4. Click **Submit**

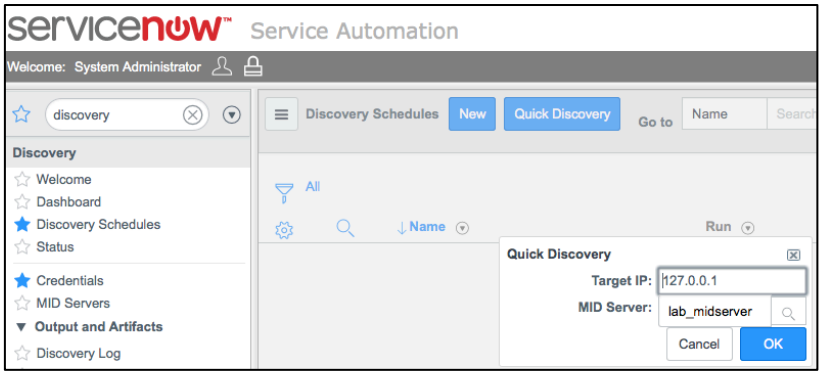
®2015 ServiceNow, Inc. All rights reserved.

Discover the lab_midserver Host

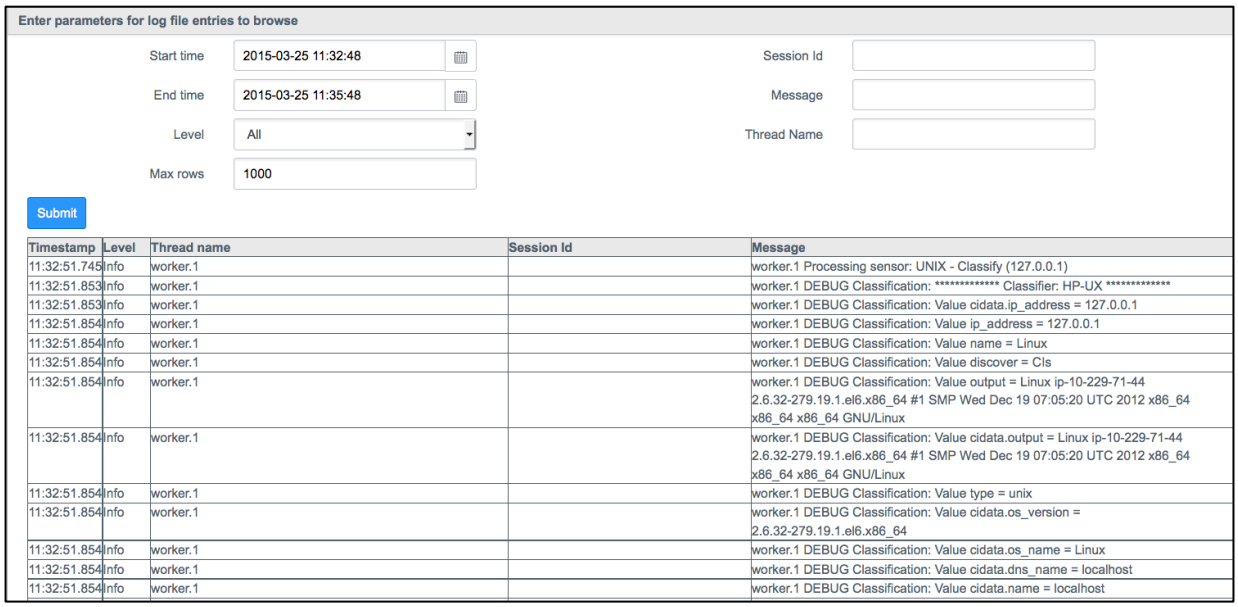
1. Navigate to **Discovery > Discovery Schedules** and select **Quick Discovery**.

   Target IP: **127.0.0.1**
   MID Server: **lab_midserver**

   Click **OK.**



2. After the discovery finishes, go to the **Node Log File Browser** and enter in the range of time that your last discovery ran. Click **Submit**.



Timestamp	Level	Thread name	Session Id	Message
11:32:51.745	Info	worker.1		worker.1 Processing sensor: UNIX - Classify (127.0.0.1)
11:32:51.853	Info	worker.1		worker.1 DEBUG Classification: ********** Classifier: HP-UX **********
11:32:51.853	Info	worker.1		worker.1 DEBUG Classification: Value cidata.ip_address = 127.0.0.1
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value ip_address = 127.0.0.1
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value name = Linux
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value discover = Cls
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value output = Linux ip-10-229-71-44 2.6.32-279.19.1.el6.x86_64 #1 SMP Wed Dec 19 07:05:20 UTC 2012 x86_64 x86_64 x86_64 GNU/Linux
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value cidata.output = Linux ip-10-229-71-44 2.6.32-279.19.1.el6.x86_64 #1 SMP Wed Dec 19 07:05:20 UTC 2012 x86_64 x86_64 x86_64 GNU/Linux
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value type = unix
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value cidata.os_version = 2.6.32-279.19.1.el6.x86_64
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value cidata.os_name = Linux
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value cidata.dns_name = localhost
11:32:51.854	Info	worker.1		worker.1 DEBUG Classification: Value cidata.name = localhost

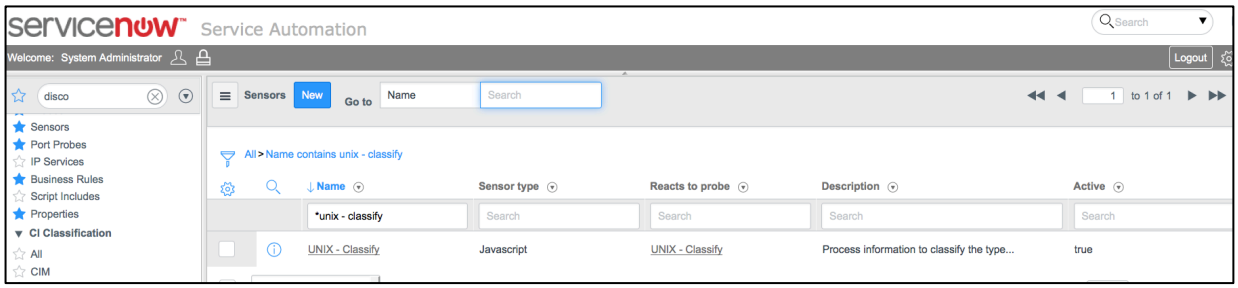
3. Note here the debug messages you asked for and the **ci.data** values that were generated.

4. Note the **ci.data.os_version = 2.6.32-279.19.1.el6.x86_64**.

®2015 ServiceNow, Inc. All rights reserved.

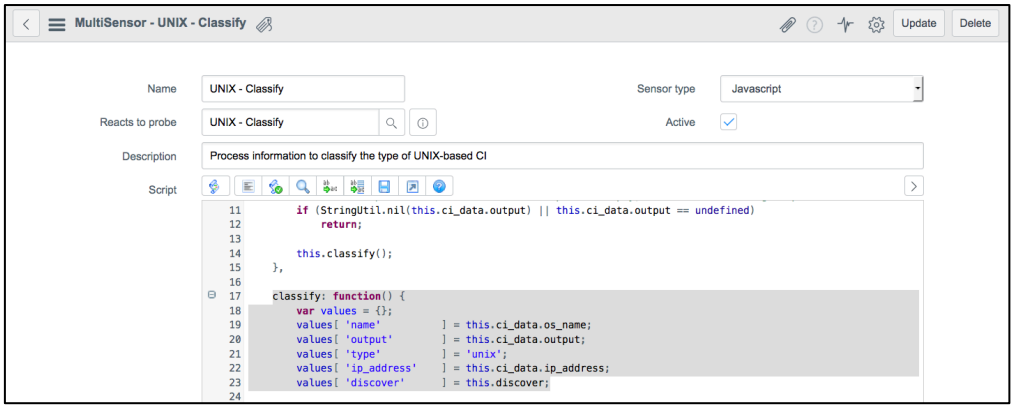
Define a Sensor

1. Navigate to **Discovery > Discovery Definition > Sensors** and open the **UNIX - Classify** sensor.



Note: Remember that it is a **best practice to create your own sensor**, copying the contents of the out-of-box sensor into your own and applying it to the appropriate probe. But, for the purposes of the lab, just modify the out-of-box **Unix - Classify** sensor.

2. Note lines 17-23, where the value is set based on the **ci.data**.



3. Add the highlighted line for **os** version to the bottom.

```
classify: function() {
        var values = {};
        values[ 'name'          ] = this.ci_data.os_name;
        values[ 'output'        ] = this.ci_data.output;
        values[ 'type'          ] = 'unix';
        values[ 'ip_address'    ] = this.ci_data.ip_address;
        values[ 'discover'      ] = this.discover;
        values[ 'osversion'     ] = this.ci_data.os_version;
```

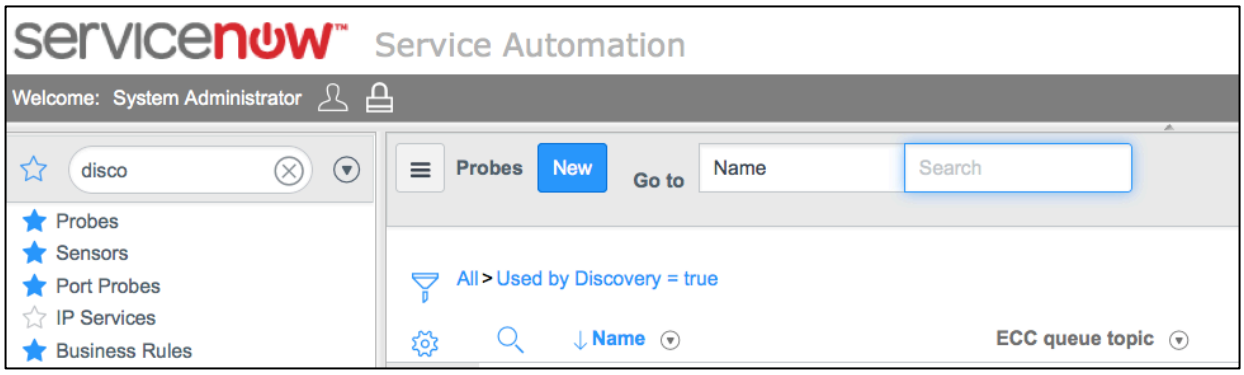
4. Click **Submit**.

Now you have a value you can call in the probe condition script called **osversion** based on values pulled from classification.

©2015 ServiceNow, Inc. All rights reserved.

Create a New Probe.

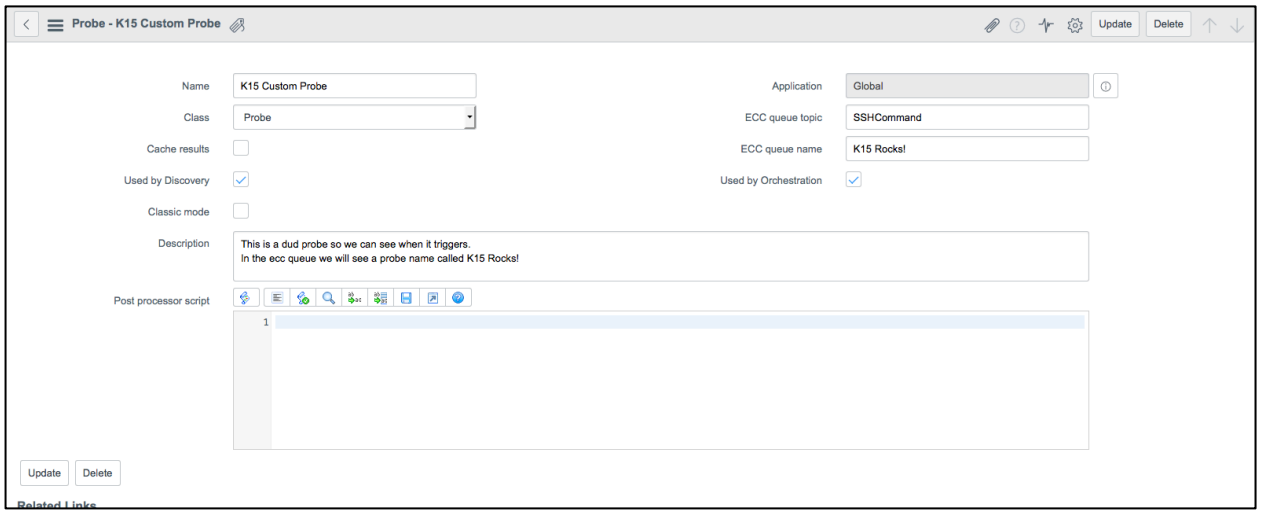
1. Navigate to **Discovery Definition > Probes**. Click **New**.



2. Enter values for the probe:

   Name: **K15 Custom Probe**
   ECC queue topic: **SSHCommand**
   ECC queue name: **K15 Rocks!**
   Description: **(Optional)**

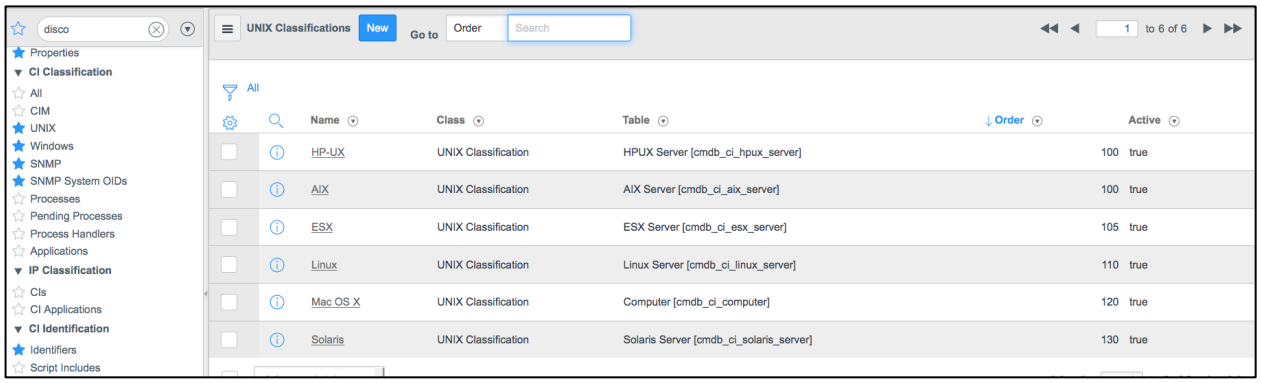
   Click **Submit**.



©2015 ServiceNow, Inc. All rights reserved.

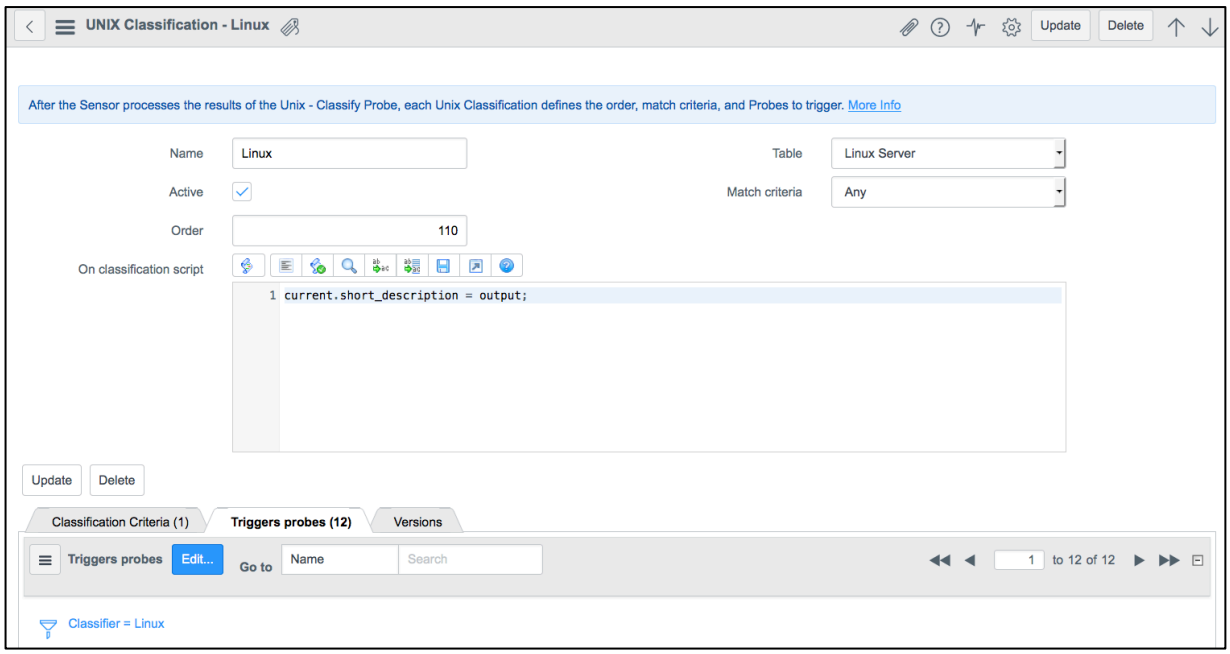
Add the Probe to the Linux Classifier

1. Navigate to **Discovery Definition > CI Classification > UNIX**.



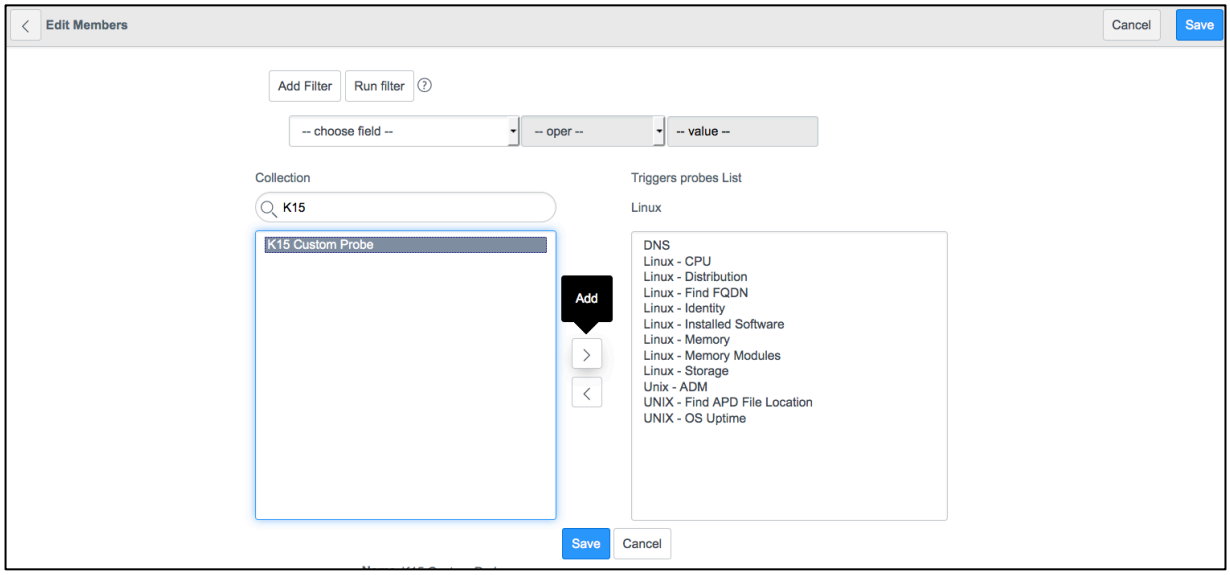
2. Select **Linux**.

3. Open the **Triggers probes** section and select **Edit**.



©2015 ServiceNow, Inc. All rights reserved.

4. Search Collection for **K15** to find the **K15 Custom probe**. Add it the **Triggers probes List**.



5. Click **Save**.

®2015 ServiceNow, Inc. All rights reserved.

Add a Condition Script

1. Select the **K15** Custom Probe from the Classifier Probe list.
2. Add to the condition script: **values.get('osversion')=='2.6.32-279.19.1.el6.x86_64'**



3. Click **Update** and confirm its existence.

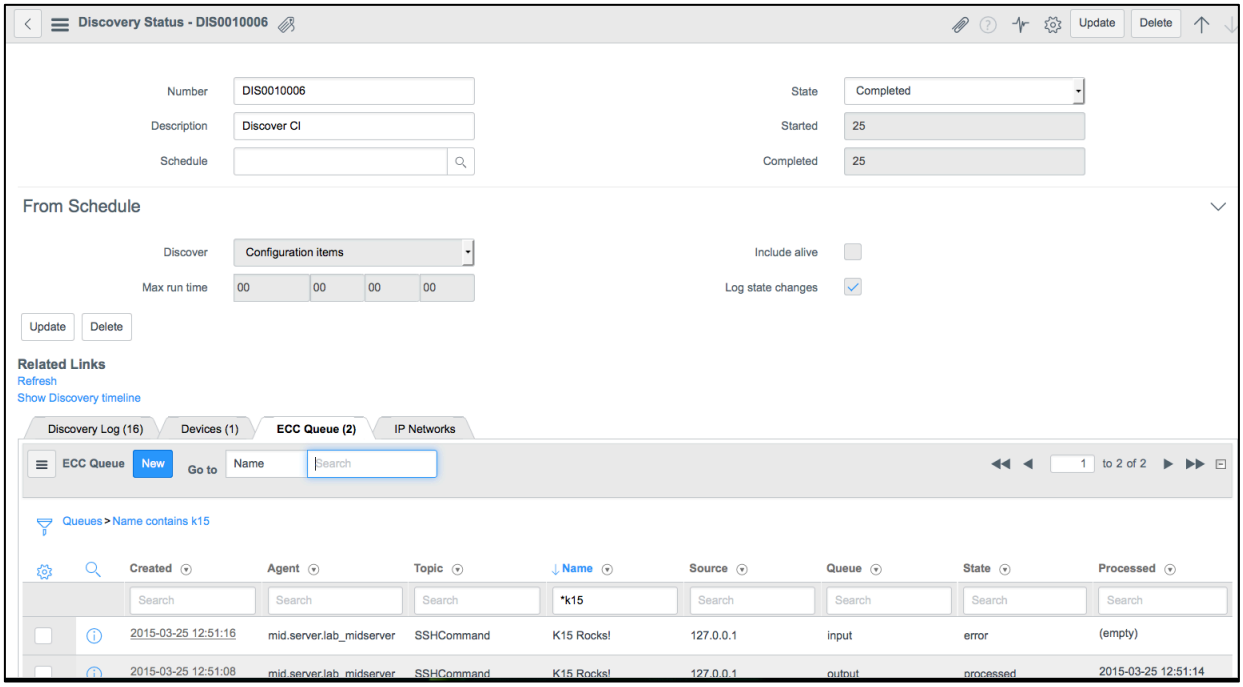


®2015 ServiceNow, Inc. All rights reserved.

Test the Condition Script

1. Discover **lab_midserver** again.

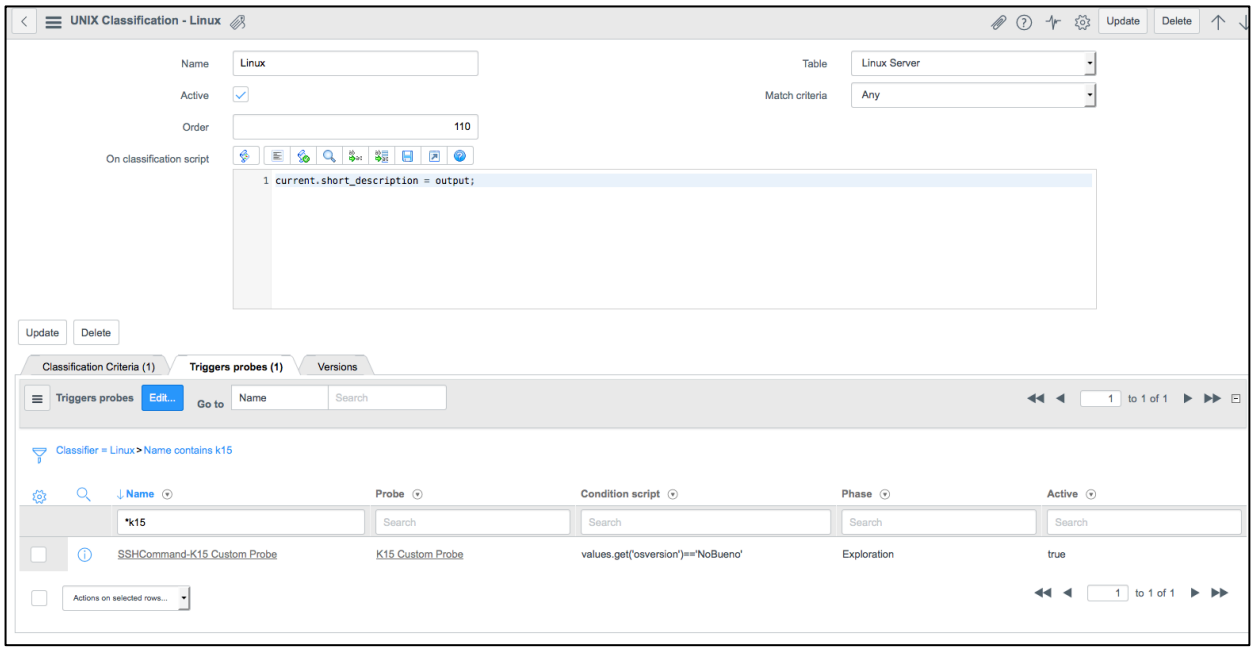
2. Check the status record and confirm that you see the probe triggered.



®2015 ServiceNow, Inc. All rights reserved.

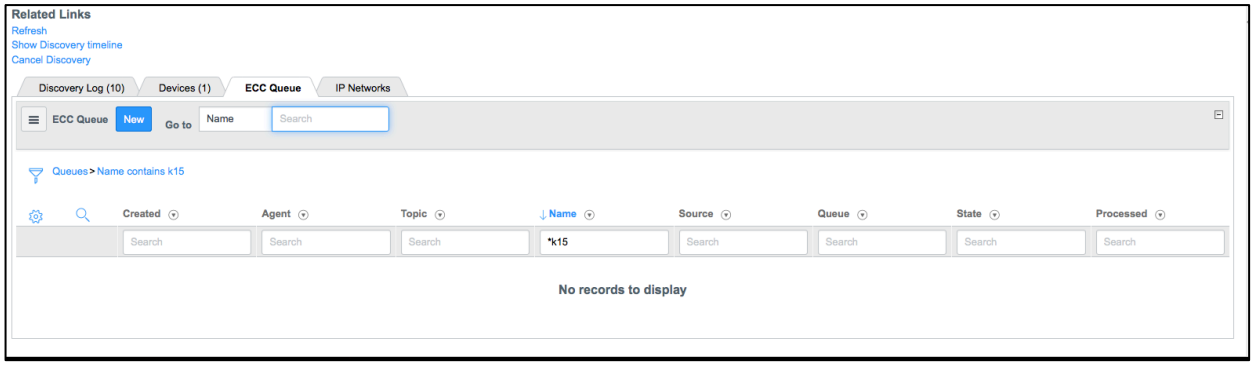
Test the Condition Script with a Bogus Value

1. Change the condition script to something random like **'NoBueno'**.



2. Discover **lab_midserver** again.

3. Identify that no K15 probe was triggered.



Best practice: disable the debug property in **sys_properties**; you do not need all that extra logging when it is not being used. For the lab, do not worry about it unless you want the practice.

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.
2. Click **I am done!**

©2015 ServiceNow, Inc. All rights reserved.