Lab Guide

Discovery Optimized 301

Kiron Valluri & Tim Stammann

Default Login / Password: Knowledge17

admin / Knowledge17

itil / Knowledge17

employee / Knowledge17

Lab Sponsored by Amazon Web Services



This
Page
Intentionally
Left
Blank

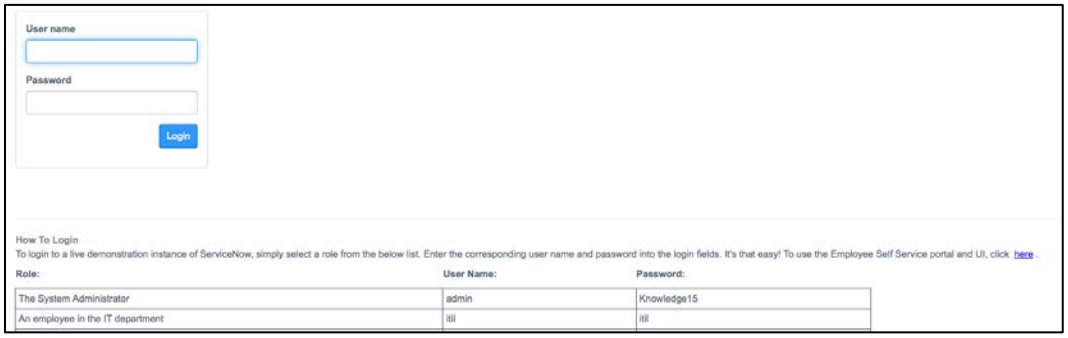
Lab Goal

This lab explains how to manage an incoming new CI. The process includes adding 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**.

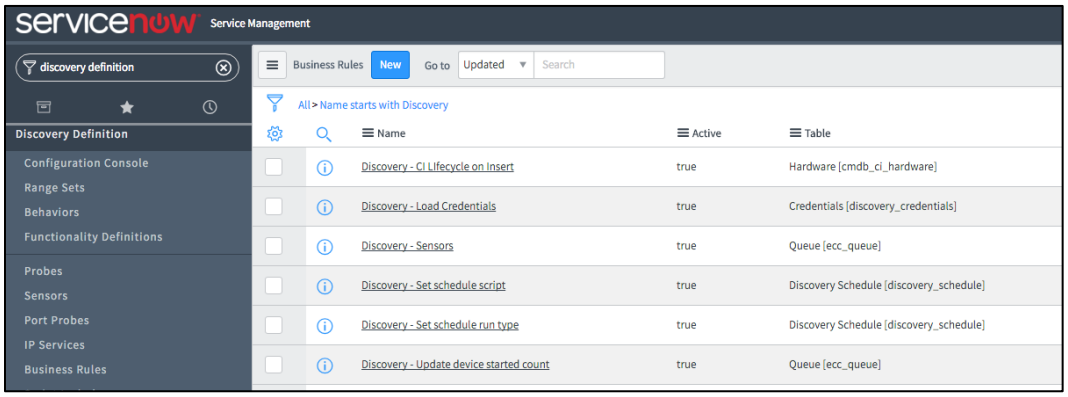


©2016 ServiceNow, Inc. All rights reserved.

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.

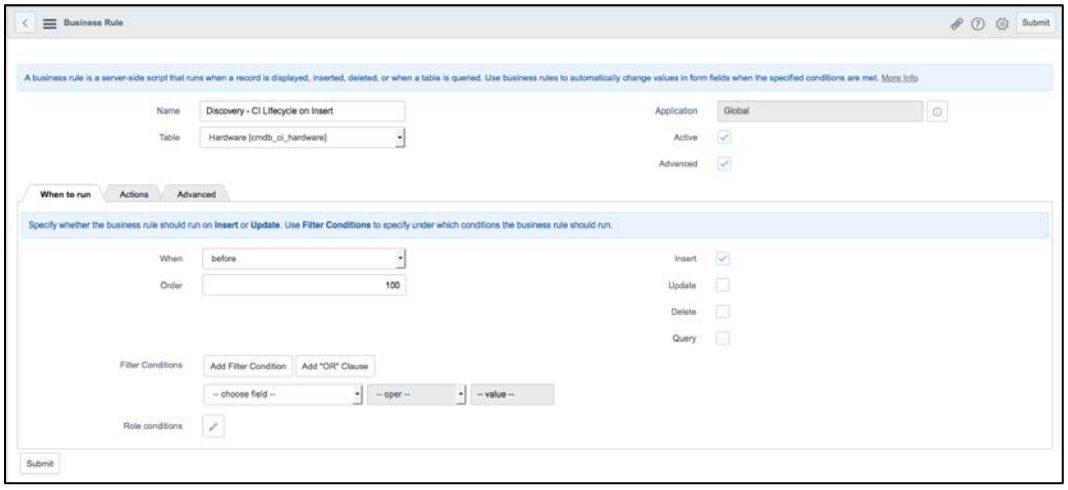


2. Select **Business Rules** and click **New**

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

   Name: **Discovery - CI Lifecycle on insert**

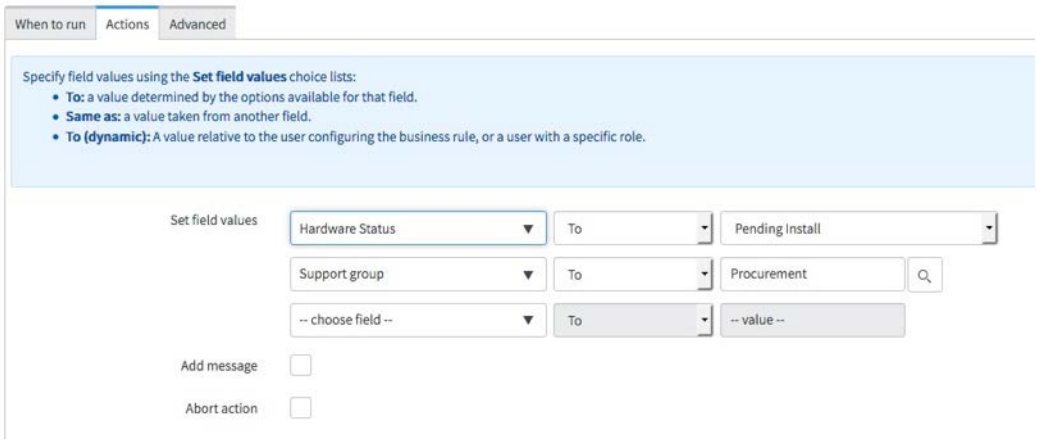
   (Use Discovery as the prefix to align with other discovery rules)
   Table: **Hardware [cmdb_ci_hardware]**
   Advanced: **checked**
   On **When to run** tab, set **When** to **before** and check **Insert**



©2016 ServiceNow, Inc. All rights reserved.

On the **Actions** tab:

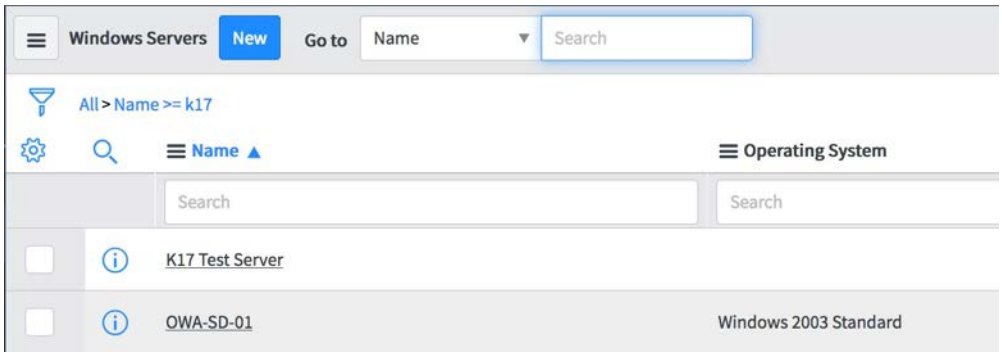
set field value: **Hardware Status To Pending Install**
set field value: **Support group To Procurement**



4. Click **Submit**.

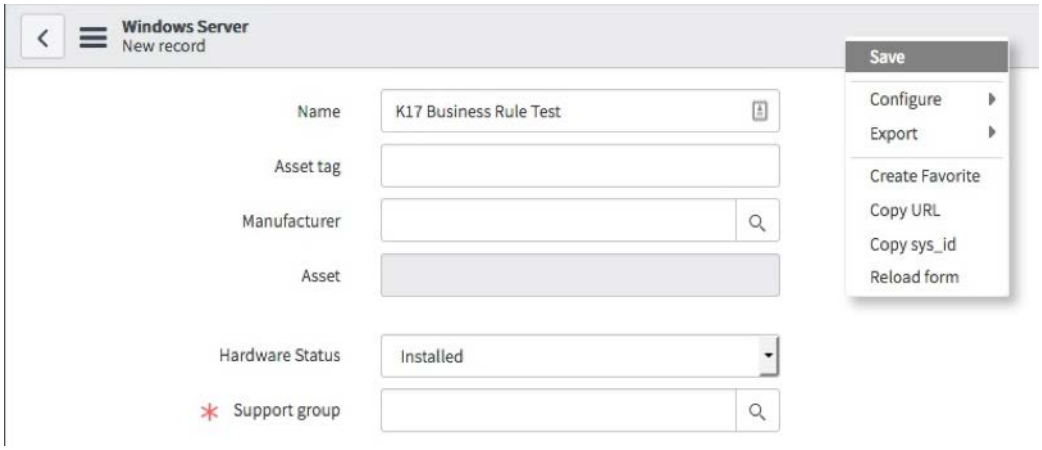
Lab Success Verification

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

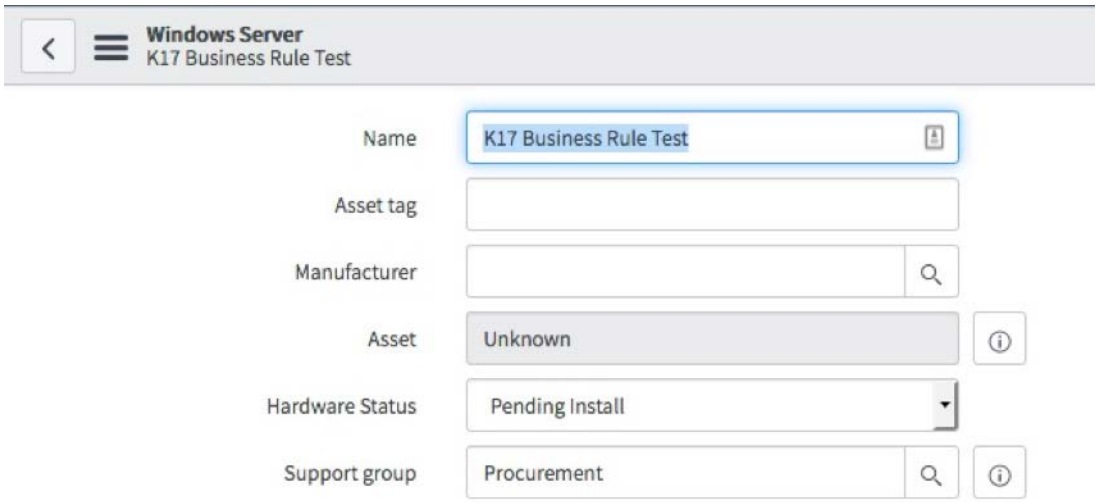


©2016 ServiceNow, Inc. All rights reserved.

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



3. Your result should look like the example.



®2016 ServiceNow, Inc. All rights reserved.

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 checkbox (**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]

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

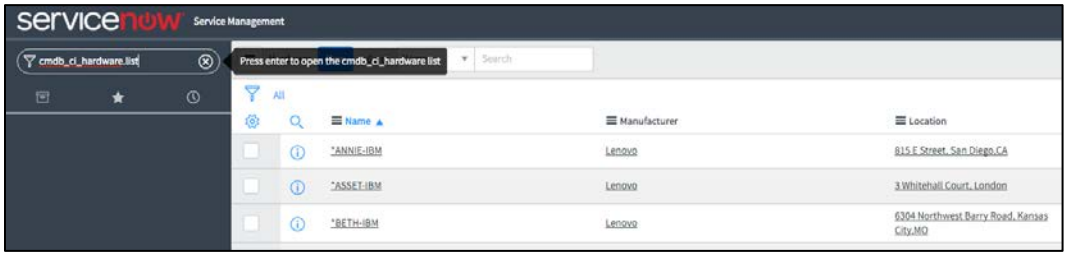
The Business Rule above created a new Hardware CI and set the state to Pending Install. This lab explains how the Procurement team can assign the CI to the appropriate support group through a mix of UI Policies and Client Scripts.

Lab 1.2
Ownership

Make a Support Group Mandatory

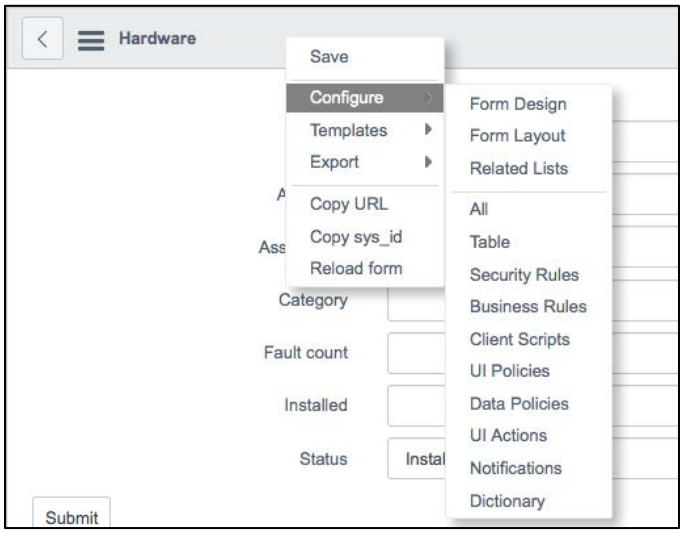
1. The Procurement group locks in a production support group when they move the hardware state to **Installed**. Do this on the base hardware table to apply it across all extended classes found by Discovery.

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

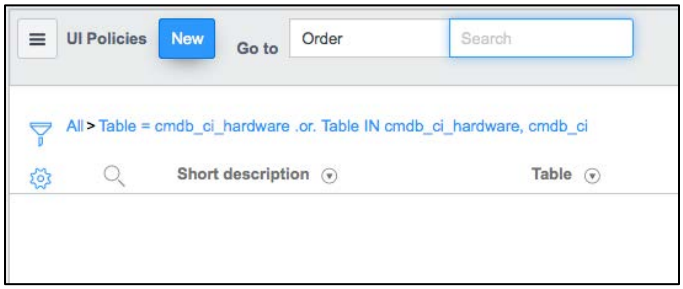


©2016 ServiceNow, Inc. All rights reserved.

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



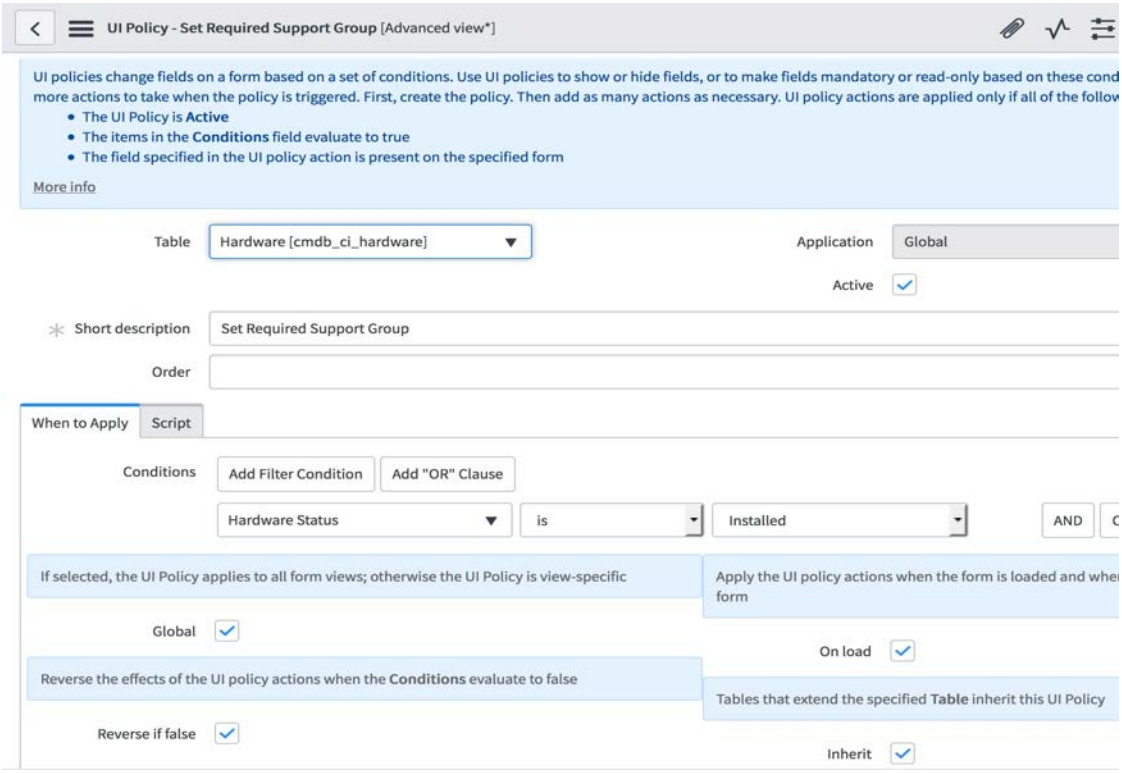
4. Click **New**.



®2016 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: **Hardware Status is Installed**
   Inherit: **checked** so the UI Policy applies to extended tables

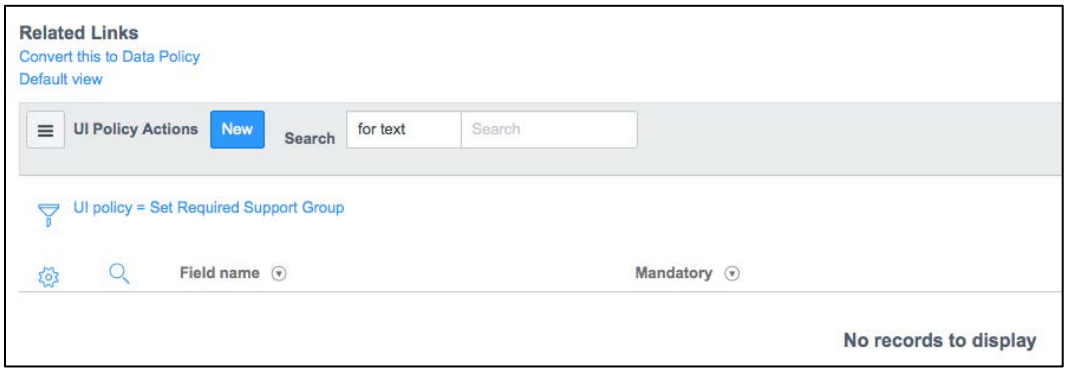


6. Right-click in the header bar and click **Save**.



©2016 ServiceNow, Inc. All rights reserved.

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



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



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



©2016 ServiceNow, Inc. All rights reserved.

Set the Support Group to Read-Only When a CI Is in Production

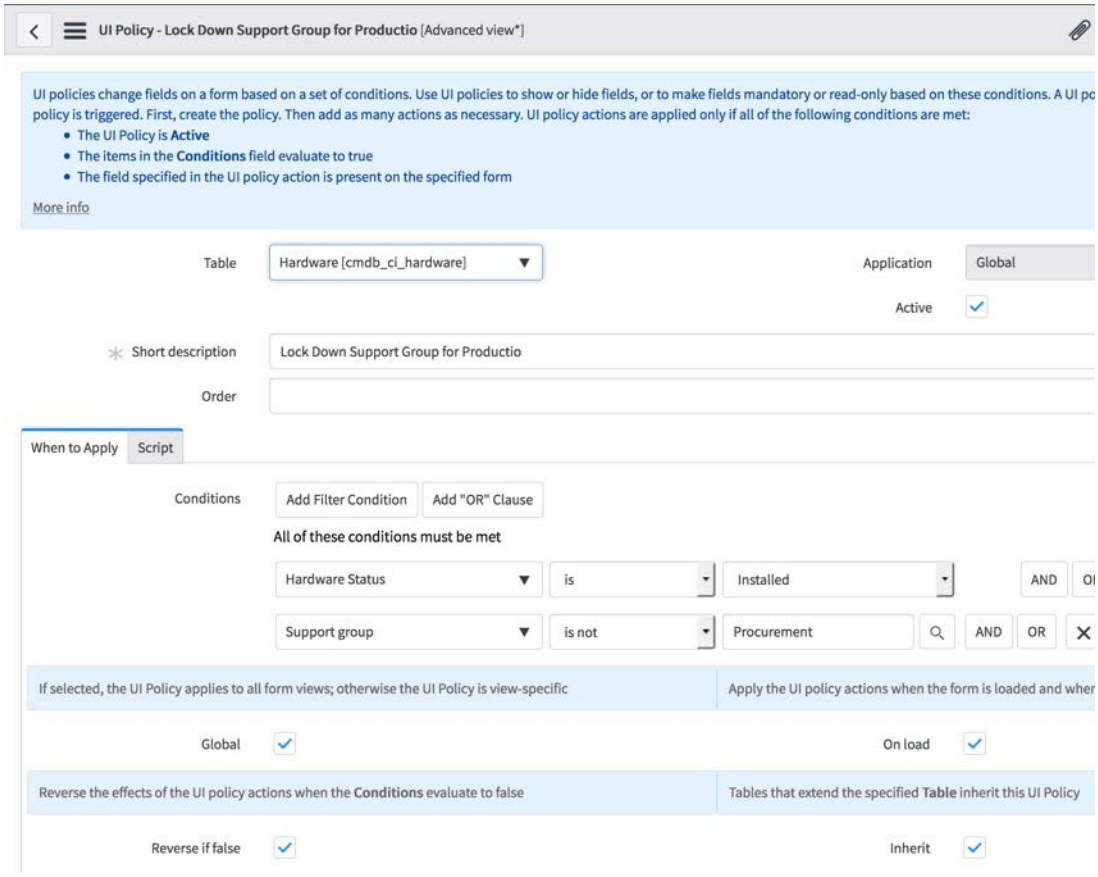
To prevent tampering of support group assignments, lock the support group field when a CI is **Installed**.

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

   Table: **Hardware [cmdb_ci_hardware]**
   Short description: **Lock Down Support Group for Production**

   On **When to Apply** tab,
   set condition: **Hardware Status is Installed**
   set condition: **Support group is not Procurement**

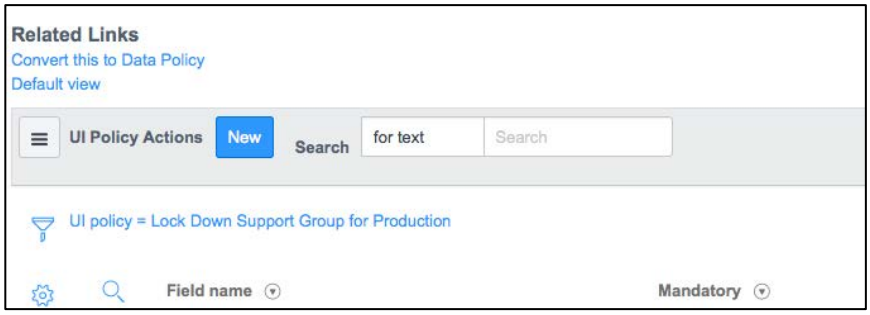
   Inherit: **checked** so the UI Policy applies to extended tables



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

©2016 ServiceNow, Inc. All rights reserved.

3. Click **New** UI Policy Action.



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

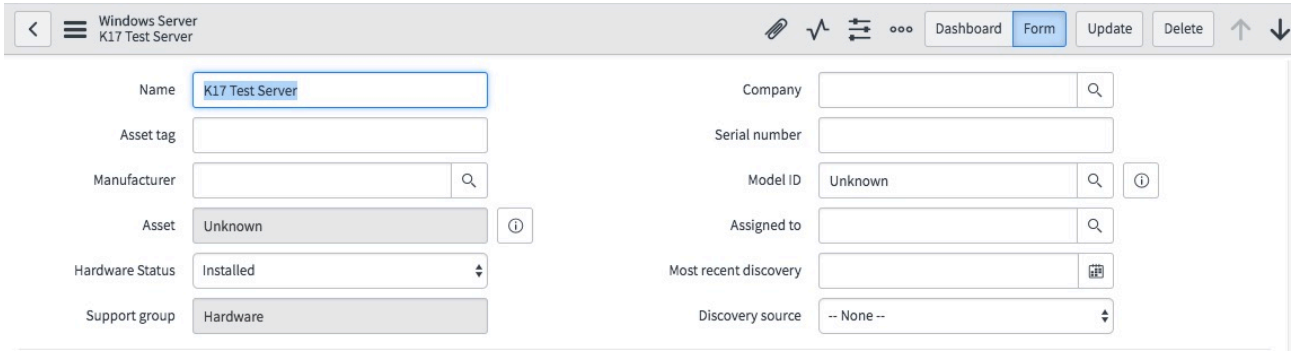
   Field name: **Support Group**
   Ready only: **True**



5. Select **Submit**.

Lab Success Verification

1. Navigate to Configuration > Servers > Windows. Click New and create a New Windows Server.
2. Change the **Hardware Status** to **Installed** and the **Support Group** to **Hardware**
3. Verify that **Support Group** is now a Mandatory field
4. Verify that the **Support Group** field is now Read-Only



©2016 ServiceNow, Inc. All rights reserved.

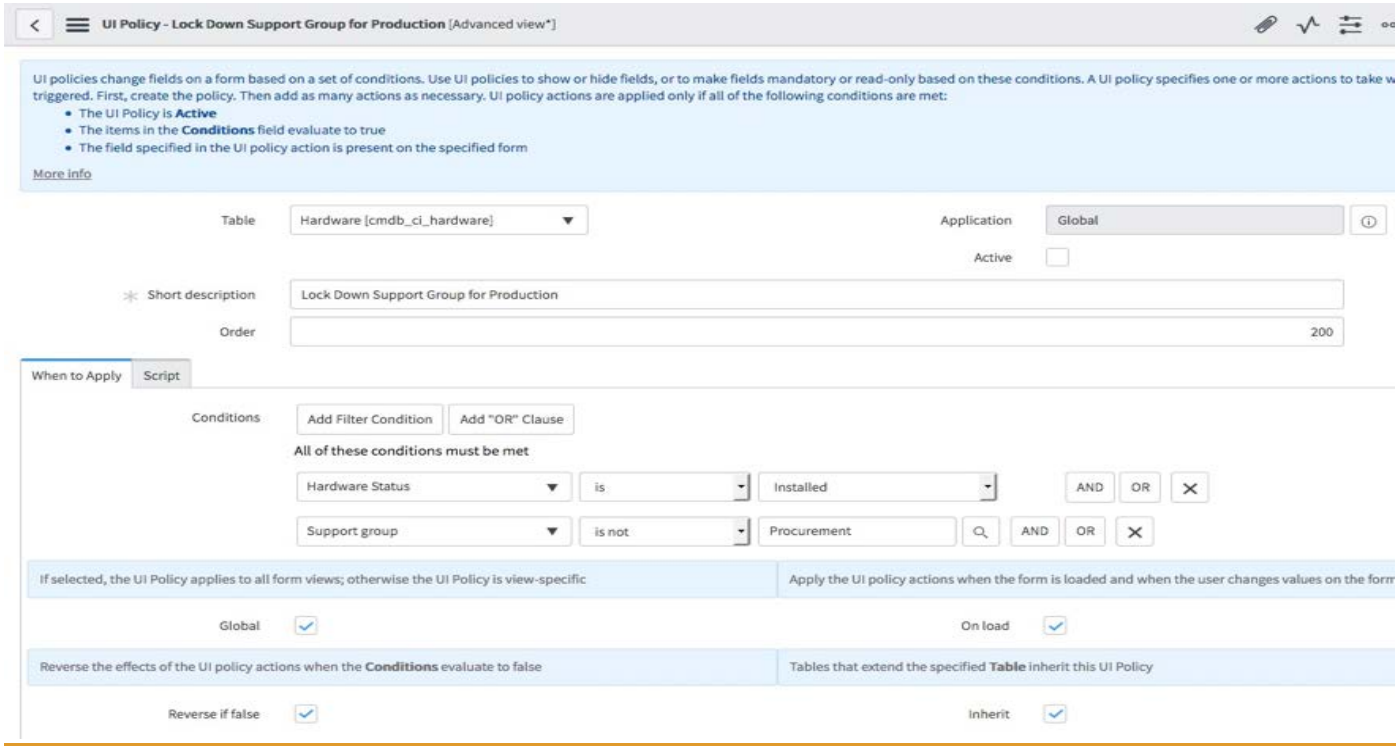
Decision Point/Think About

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 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', '');}
```



©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 2.1
Life as a CI

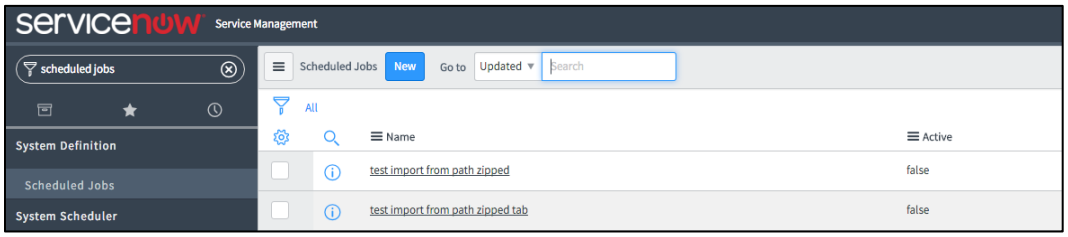
There is now a good process for setting up new CIs that are discovered and a solid method for allowing Procurement to assign responsible parties, while keeping them honest at the same time. This lab focuses on what happens to those CIs as they live in an environment. What happens when they disappear?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

To identify the lost CIs, a field called **Most Recent Discovery** can be used. 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.

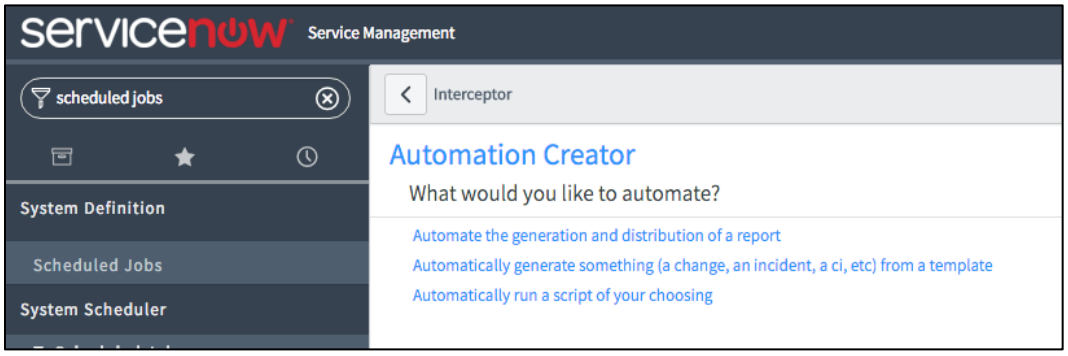
©2016 ServiceNow, Inc. All rights reserved.

Identify Lost CIs

1. Navigate to System Definition > Scheduled Jobs and click New.



2. Choose Automatically run a script of your choosing.



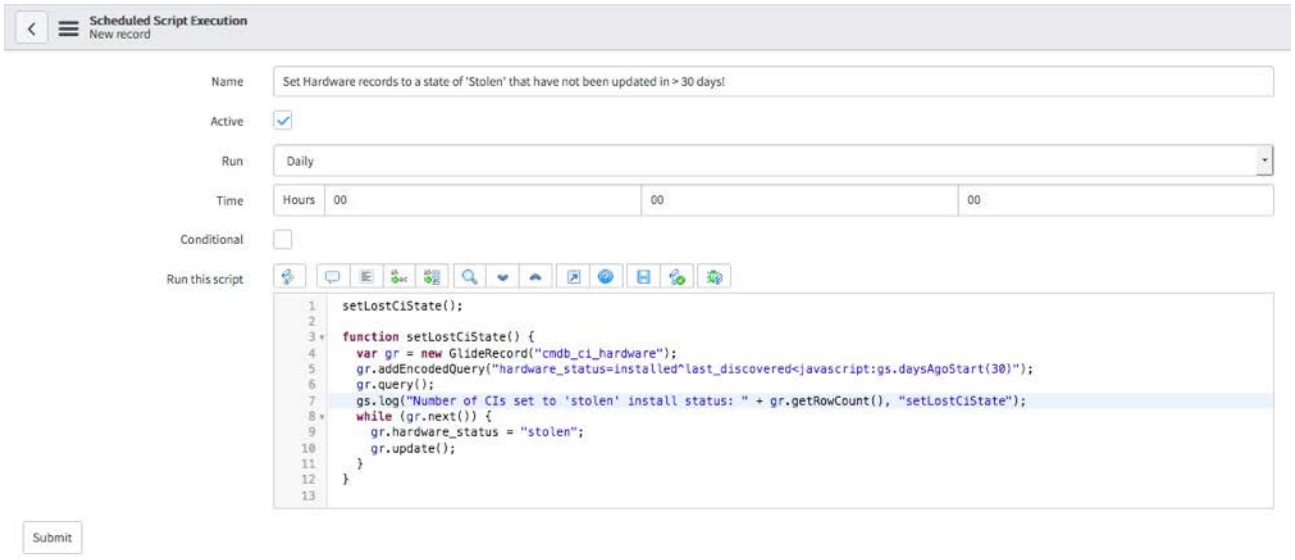
©2016 ServiceNow, Inc. All rights reserved.

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 Hardware records to a state of 'Stolen' 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();
  }
}
```

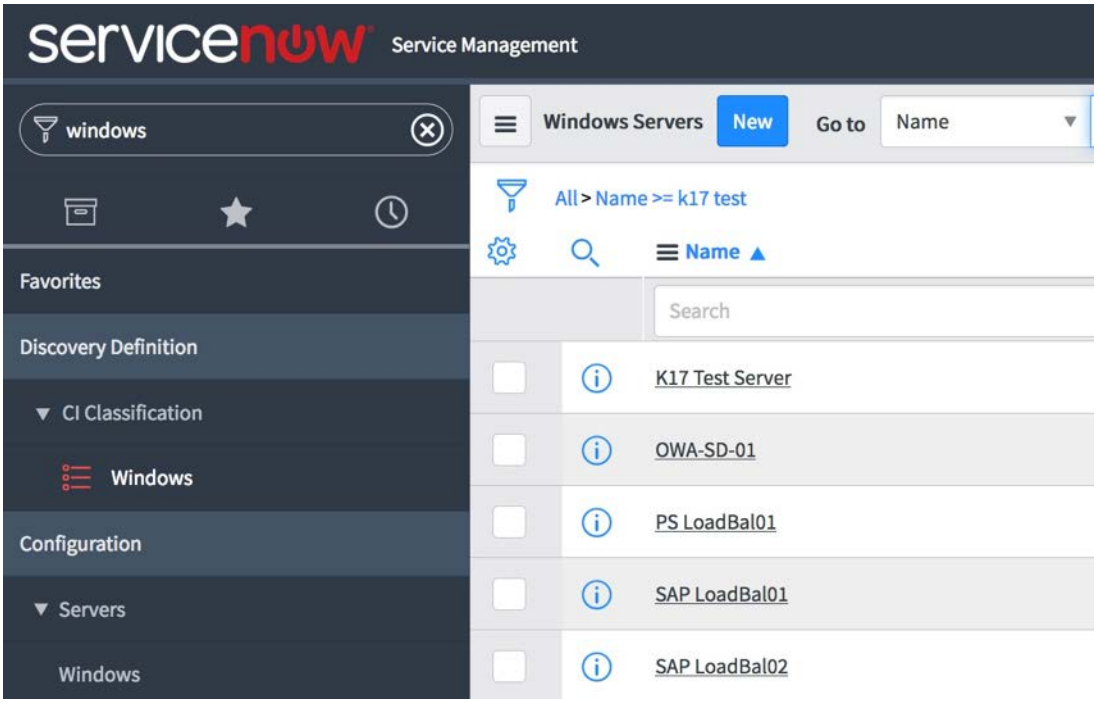


4. Click **Submit**.

©2016 ServiceNow, Inc. All rights reserved.

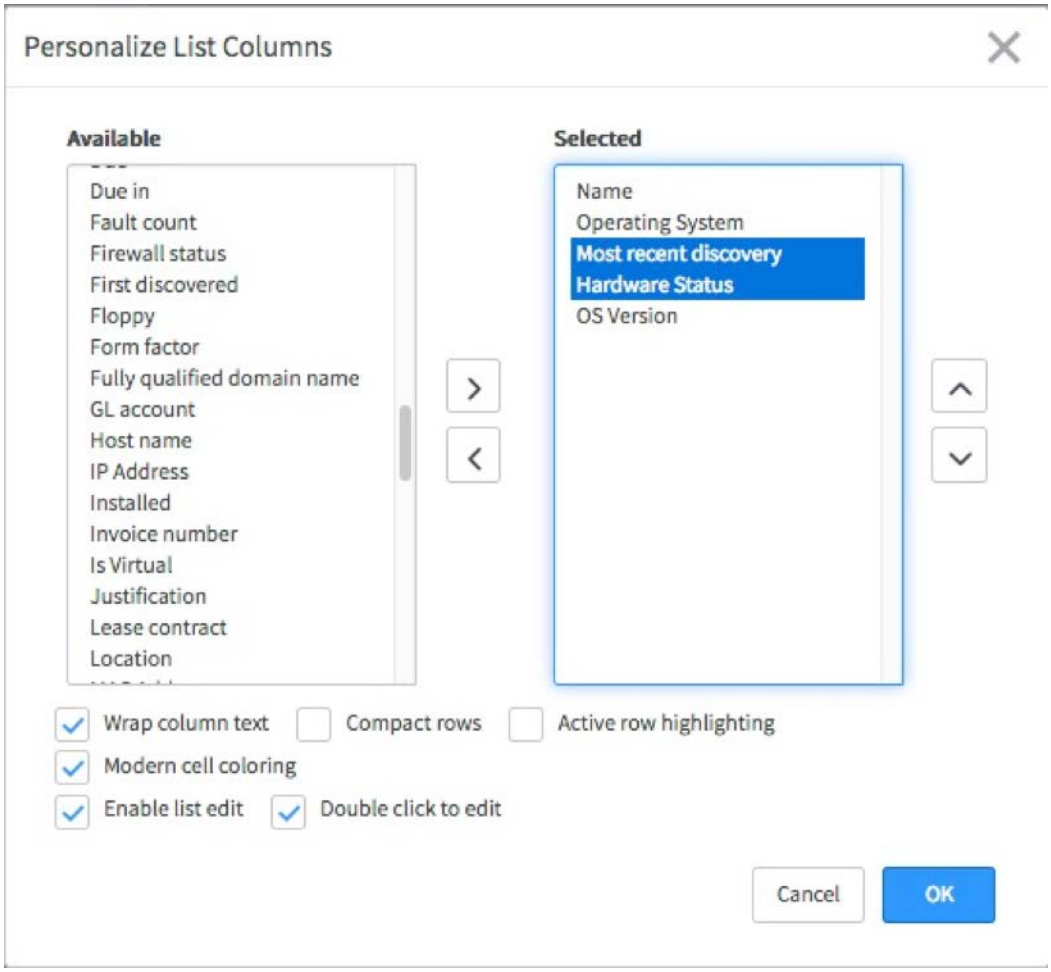
Lab Success Verification

1. Navigate to Configuration > Servers > Windows.



©2016 ServiceNow, Inc. All rights reserved.

2. Modify the view to include Most recent discovery and Hardware Status.

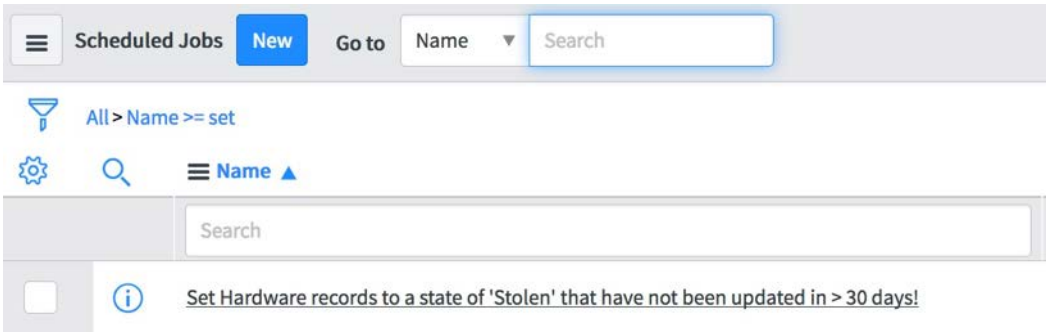


3. Using the Windows Server you created in the previous lab, set the most recent discovery date in the past – last year.

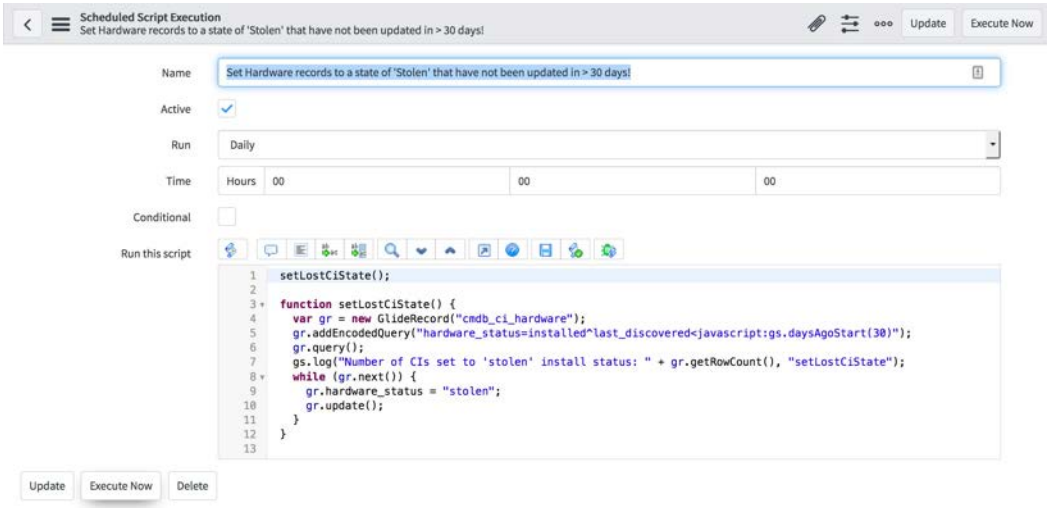


©2016 ServiceNow, Inc. All rights reserved.

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



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



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



And the asset:



©2016 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.

What should you do when a device goes missing? You could send an email notification to the support group or create an incident assigned to the support group looking for information about the CI.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 2.2
Life as an App CI

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.

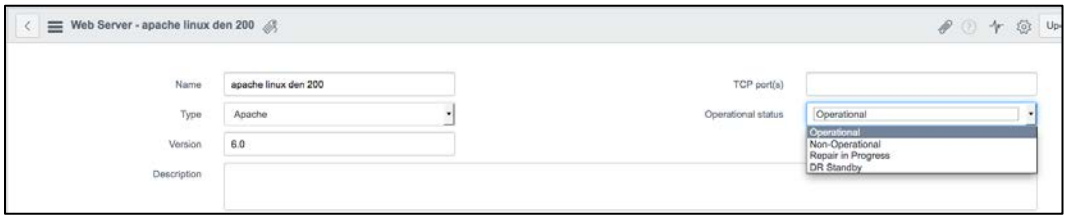
Set CI Status in the Application Table (cmdb_ci_appl)

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. Use the out-of-box **Operational** status for this exercise.



3. On the record of your choosing, set the **most recent discovery** value to sometime in the past—for example, last year.

4. Navigate to **System Definition > Scheduled Jobs** and click **New**.

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

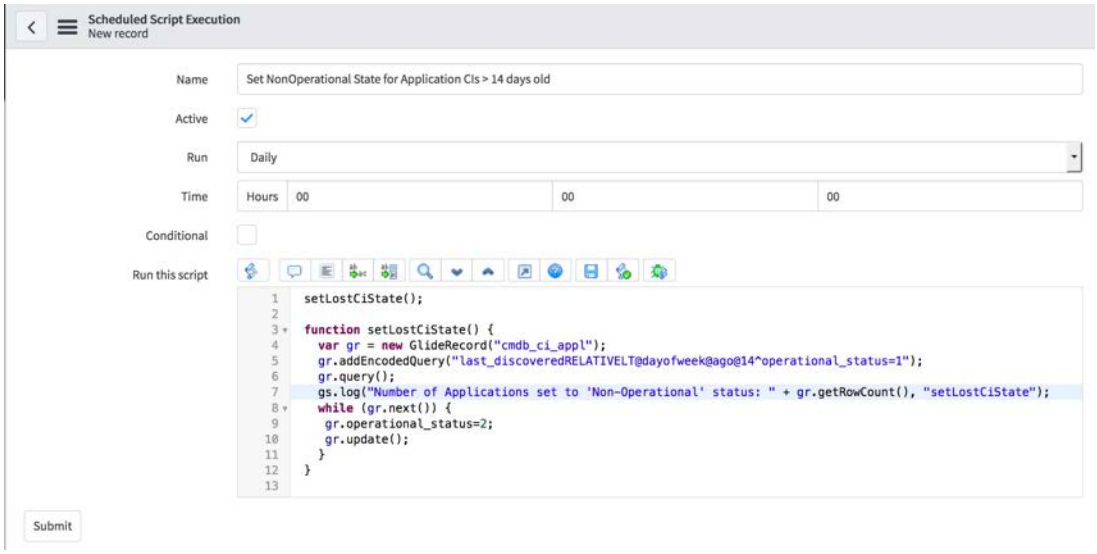
©2016 ServiceNow, Inc. All rights reserved.

6. 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();
  }
}
```



7. Right-click the gray header bar, then **Save**.

8. Click **Execute now**.

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



©2016 ServiceNow, Inc. All rights reserved.

Decision Point/Think About

Why not remove the Application CIs that have not shown themselves in a while? Remember, applications are not assets. They spin up and down all the time! There might be no reason to keep the clutter. Your process can determine that it is okay to remove them after 30 days, for example.

It then becomes just another scheduled job that looks through the Application table, then removes 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 more than 30 days ' + myCMDBApps.name + ' deleted');
myCMDBApps.deleteRecord();
}
```

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 3.0 Time to Say Goodbye

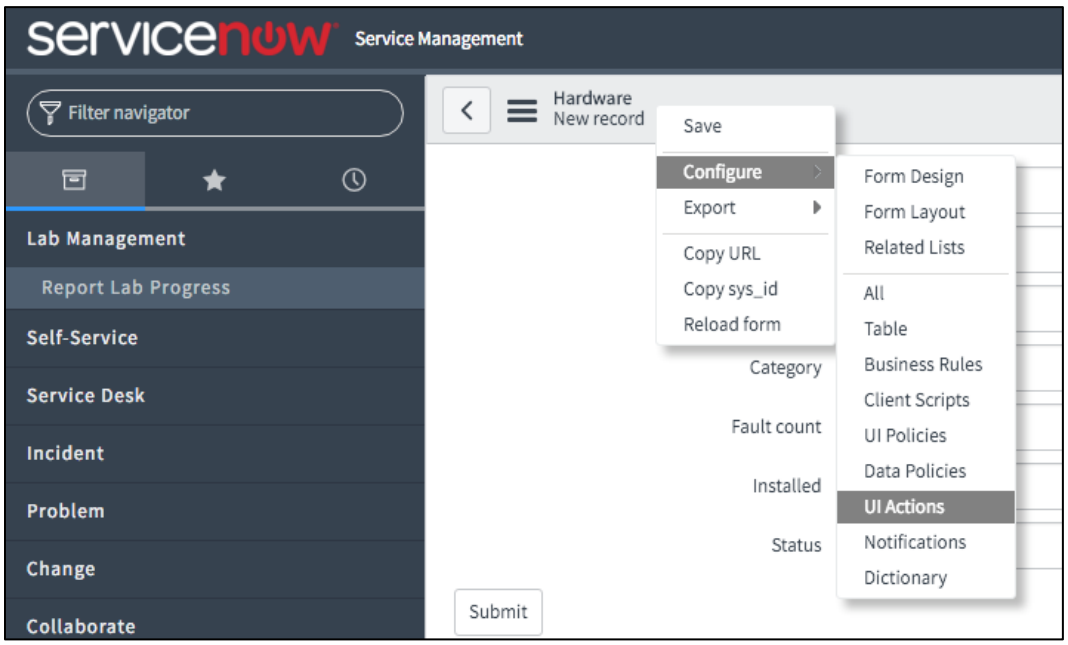
Life has been great having the CIs around, but now it is time to say goodbye (queue Bocelli's "Time to Say Goodbye," or even better, the Will Ferrell version from *Step Brothers*).

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.

Create UI Action to Remove Related Records When a CI Is Retired

1. In the Application Search Bar, type **cmdb_ci_hardware.list** to return quickly to the **Hardware Table.**

2. Click **New**. Right-click the header bar and select **Configure > UI Actions**.

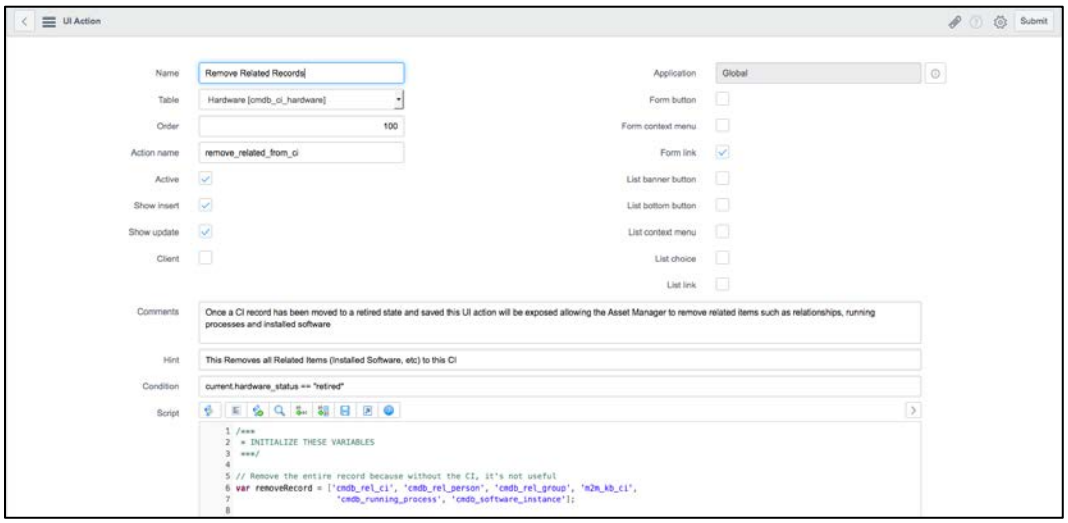


3. Click **New**.

©2016 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
 *
 * @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
 *
```

©2016 ServiceNow, Inc. All rights reserved.

```
 */

 // 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;
		}

		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
```

©2016 ServiceNow, Inc. All rights reserved.

```
		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');

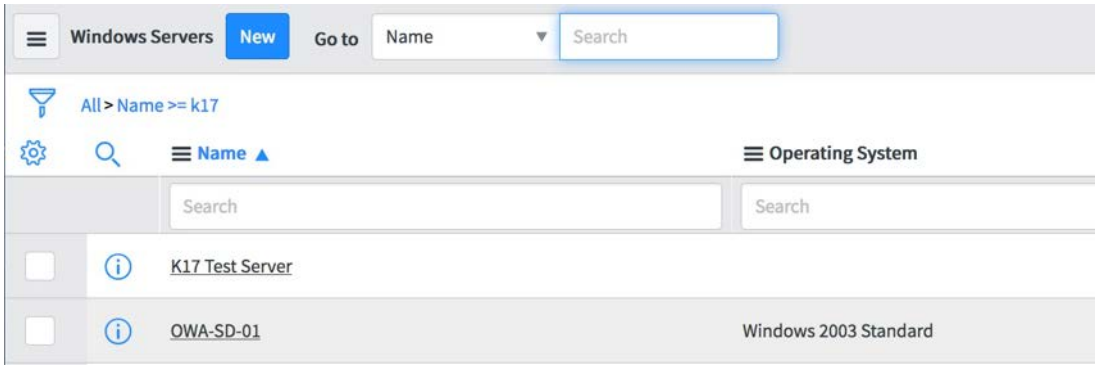
	dict.addAggregate('COUNT');
	dict.addQuery('name', tableName);
	dict.addQuery('internal_type', 'reference');
	dict.addEncodedQuery(taskQuery);

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

	return count != 0;
}
```

©2016 ServiceNow, Inc. All rights reserved.

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



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



©2016 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 means you 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.

- Would you rather just delete Virtual Machines if they were not needed for auditing or regulatory purposes?
- 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?

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

This lab explains how to track unplanned changes to computer or application records that are being discovered.

One of the Godfathers of Discovery, Aleck Lin, helped identify a quick and effective business rule that tracks the attribute changes on a CI as Discovery updates a record. If an identified attribute does get modified, we create an unplanned change record.

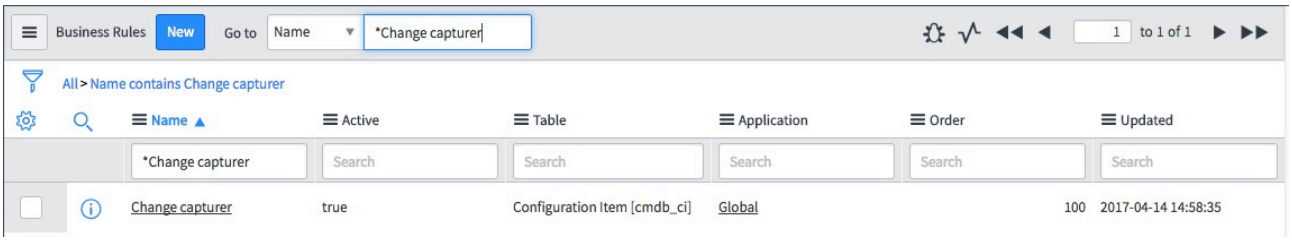
Lab 4.0
Who Said You Could Change?

It's Alive!

1. The first order of business is setting a business rule on the hardware table that checks to see, on update, if a device has been modified.

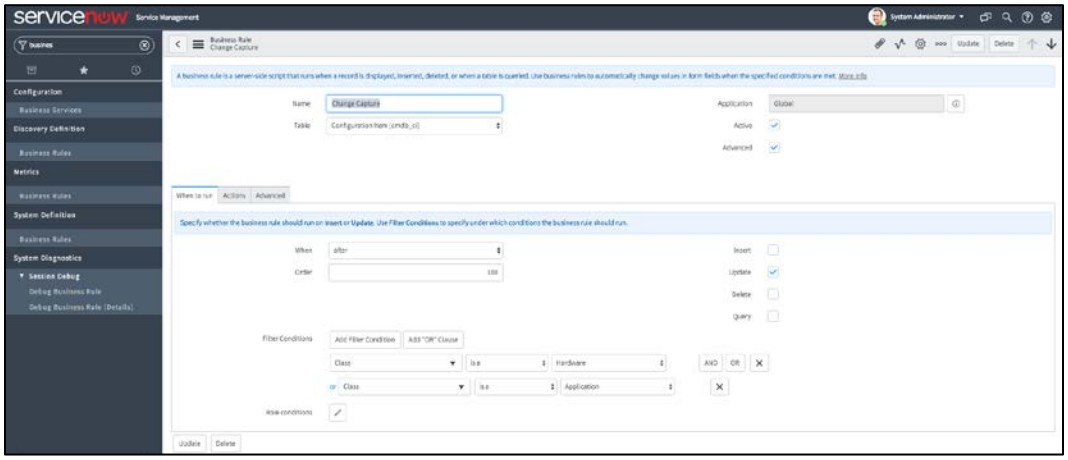
2. Navigate to System Definition > Business Rules.

3. Search for Business Rule **Change Capturer**.



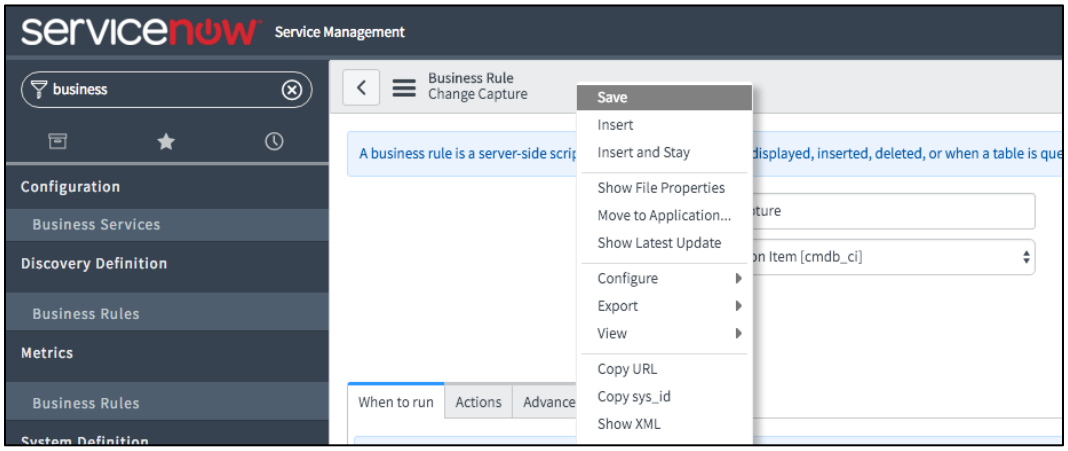
4. Make sure the Business Rule is **Active.**

©2016 ServiceNow, Inc. All rights reserved.



Decision Point/Think About: What if only the MID Server's Role should be the one impacting this action? What are the pros/cons for this impact in your processes?

5. Right-click on the gray bar above and select **Save**.



6. Select the **advanced** tab.

7. Let's start reviewing our script. To begin:

©2016 ServiceNow, Inc. All rights reserved.

8. Shall we dance? This measures line 6 above if we want to care if the record is part of a service.

```
12
13      // If we want to even move forward at all...
14      if (shallWeDance())
15          letsDance();
16
17      function shallWeDance() {
18          if (serviceMatters && partOfService()) {
19              //gs.log("Exiting change validation script because it is not part of a business service.");
20              return false;
21          }
22
23          if (!isThereAnyChange())
24              return false;
25
26          return true;
27      }
28
29      function partOfService() {
30          var gr = new GlideRecord("svc_ci_assoc");
31          gr.addQuery("ci_id", current.sys_id);
32          gr.setLimit(1);
33          gr.query();
34          if (gr.next())
35              return true;
36
37          return false;
38      }
39
```

9. Time to dance. Let's look up if there's an associated change record.

```
39
40      function isThereAnyChange() {
41          //Placeholder for now... will need to be more fancy later...
42          changedFieldNames = getNameOfFieldsThatChanged(current);
43
44          if (changedFieldNames.length == 0)
45              return false;
46
47          return true;
48      }
49
50      function letsDance() {
51          // Must return a boolean value to indicate if this change was authorized or not
52          var now = gs.nowNoTZ();
53
54          var message = _generateMessage(current);
55
56          var changeDetails = message;
57          var result = false;
58
59          var mutex = new SelfCleaningMutex('isValidChange_' + current.sys_id);
60          if (!mutex.get()) {
61              gs.log('Change Capturer Script: FATAL unable to obtain mutex, bailing out.');
62              return;
63          }
64          try {
65              var change = findScheduledChange();
66              if (change) {
67                  commentOnChange();
68                  result = true;
69              } else {
70                  change = findUnplannedChange() || createUnplannedChange();
71                  if (change) {
72                      commentOnChange();
73                  }
74              }
75          } finally {
76              mutex.release();
77          }
78
79          return result;
80
81
```

©2016 ServiceNow, Inc. All rights reserved.

10. Do the work that we defined in line 65 above.

```
80
81
82      function findScheduledChange() {
83          // Quite simplistic, only looks for App/hardware CIs being affected.
84          // When we take more alert patterns into account , more complex relationships to CHG may be needed.
85          // Even now, is a planned change on the underlying host sufficient to correlate on? or MUST it be the app?
86
87          // Find any change to CI
88          var chg = new GlideRecord('change_request');
89          chg.addActiveQuery();
90          chg.addQuery('cmdb_ci', current.sys_id);
91          chg.addQuery('start_date', '<', now);
92          chg.addQuery('end_date', '>', now);
93          chg.query();
94          if (chg.next()) {
95              return chg;
96          }
97
98          // Find any change where CI is an affected CI
99          var tci = new GlideRecord('task_ci');
100         tci.addQuery('ci_item', current.sys_id);
101         tci.addQuery('task.sys_class_name', 'change_request');
102         tci.addQuery('task.active', true);
103         tci.query();
104         while (tci.next()) {
105             if (chg.get(tci.task) && chg.start_date < now && chg.end_date > now) {
106                 return chg;
107             }
108         }
109     }
110
```

11. Add to comments if we found a change.

```
110
111     function commentOnChange() {
112         var comment = 'Detected change. ' + changeDetails;
113
114         var existingComments = change.comments.getJournalEntry(-1).split('\n\n');
115         for (var i = 0; i < existingComments.length; i++) {
116             var existingComment = existingComments[i];
117
118             existingCommentMod = existingComment.replace(/\n/g, '');
119             commentMod = comment.replace(/\n/g,'');
120
121             if (existingCommentMod.indexOf(commentMod) >= 0) {
122                 return;
123             }
124         }
125
126         change.comments = comment;
127         change.update();
128     }
129
```

12. More work in searching for our unplanned change.

```
129
130     function findUnplannedChange() {
131         var chg = new GlideRecord('change_request');
132         chg.addActiveQuery();
133         chg.addQuery('cmdb_ci', current.sys_id);
134         chg.addQuery('type', 'Unplanned');
135         chg.addQuery('sys_created_on', '<=', now);
136         chg.query();
137         if (chg.next()) {
138             // If we found a *different* change to the same CI, make it known
139             if (chg.short_description != message) {
140                 chg.short_description = 'Multiple changes detected - see comments for details';
141                 if (chg.description.indexOf('Initial change: ') != 0) {
142                     chg.description = 'Initial change: ' + chg.description;
143                 }
144                 chg.update();
145             }
146             return chg;
147         }
148     }
149
```

©2016 ServiceNow, Inc. All rights reserved.

13. Do the actual work.

```
149
150         function createUnplannedChange() {
151             var chg = new GlideRecord('change_request');
152             chg.setValue('cmdb_ci', current.sys_id);
153             chg.setValue('short_description', message);
154             chg.setValue('description', 'Detected a change outside of a scheduled change: ' + changeDetails);
155             chg.setValue('type', 'Unplanned');
156             chg.insert();
157             return chg;
158         }
159
```

14. Make the message.

```
159
160         function _generateMessage() {
161             var plural = (changedFieldNames.length > 1);
162
163             var msg = "The following field" + (plural ? "s " : " ") + (plural ? " were" : " was") + " changed by " + current.sys_updated_by + ":\n";
164             for (var i=0; i<changedFieldNames.length; i++) {
165                 var fieldName = changedFieldNames[i]+'';
166
167                 var newValue = gs.nil(current[fieldName]) ? "''" : "'" + current[fieldName] + "'";
168                 var oldValue = gs.nil(previous[fieldName]) ? "''" : "'" + previous[fieldName] + "'";
169
170                 msg += fieldName + " is updated to " + newValue + " (Previously " + oldValue + ")\n";
171             }
172
173             return msg;
174         }
175
176     }
177
```

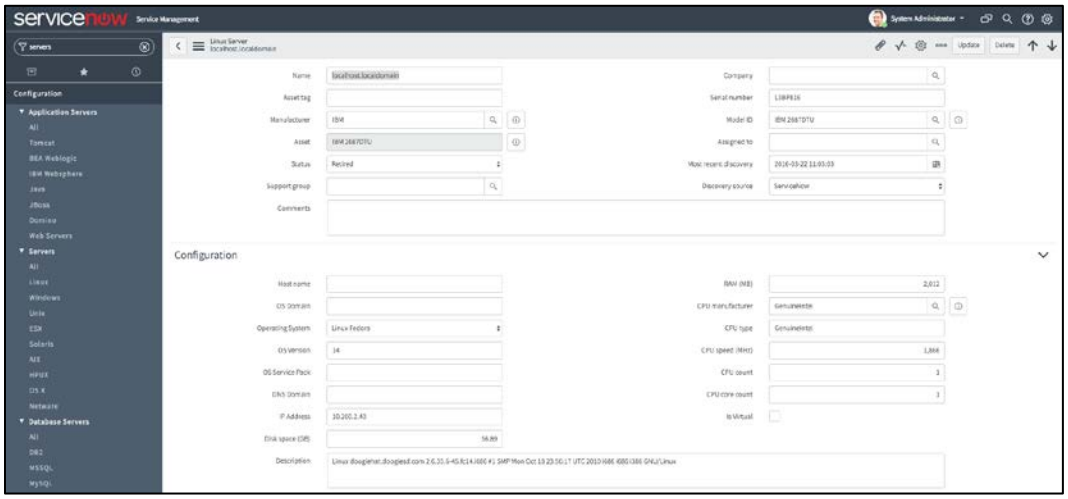
®2016 ServiceNow, Inc. All rights reserved.

15. And find what we want to fill in that message.

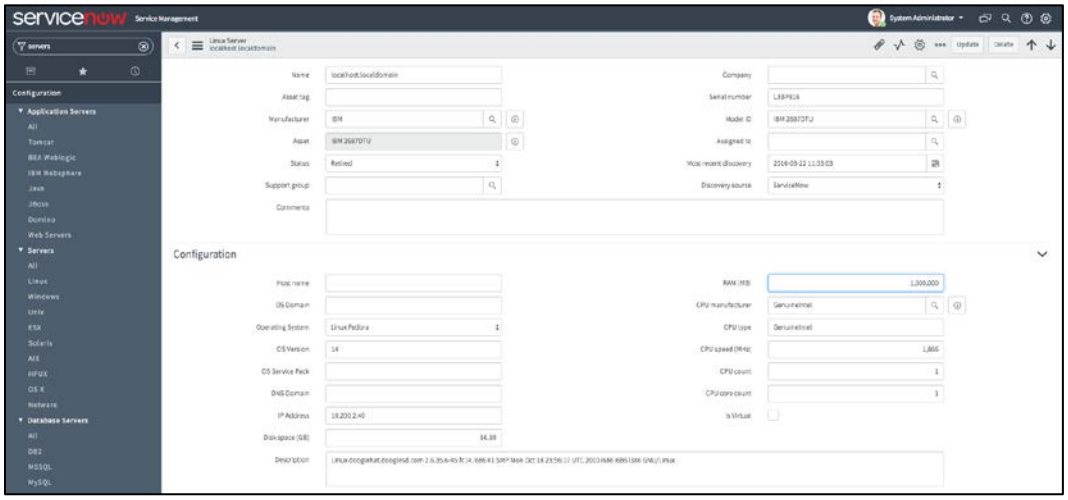
```
177
178     function getLabelOfFieldsThatChanged(gr) {
179         var changes = [];
180
181         if (gr == null || !gr.isValid())
182             return changes;
183
184         var gru = GlideScriptRecordUtil.get(gr);
185         var whatChanged = gru.getChangedFields();
186         for (var i = 0; i < whatChanged.size(); i++) {
187             var f = whatChanged.get(i);
188             if(ignoreAttributes[f])
189                 continue;
190             changes.push(f);
191         }
192
193         return changes;
194     }
195
196     function getNameOfFieldsThatChanged(gr) {
197         var changes = [];
198
199         if (gr == null || !gr.isValid())
200             return changes;
201
202         var gru = GlideScriptRecordUtil.get(gr);
203         var whatChanged = gru.getChangedFieldNames();
204         for (var i = 0; i < whatChanged.size(); i++) {
205             var f = whatChanged.get(i);
206             if(ignoreAttributes[f])
207                 continue;
208             changes.push(f);
209         }
210
211         return changes;
212     }
213
214
215
216 })(current, previous);
```

©2016 ServiceNow, Inc. All rights reserved.

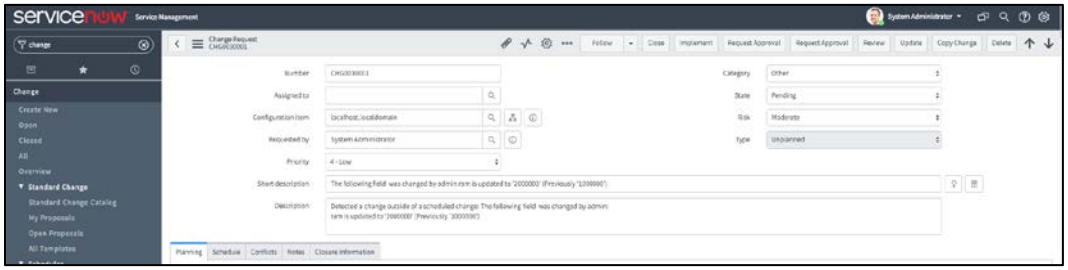
16. Test this out! Go to the **Linux Server** table and select **lnux100**.



17. Change any value on the form, say RAM? Give it a million MB and select **Update**.



18. Then go to all change records and see the one created today!

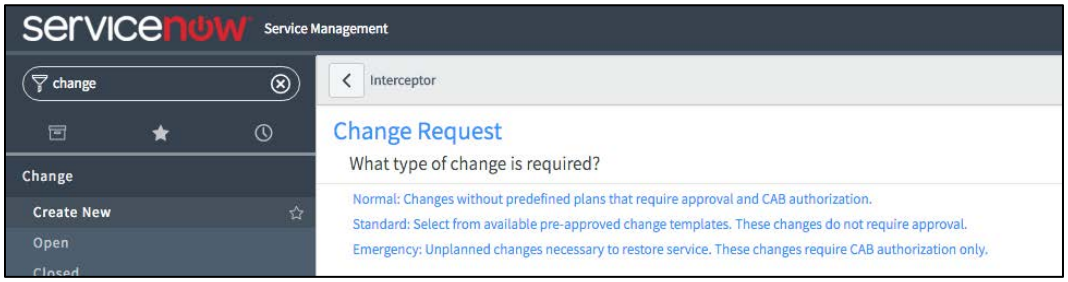


©2016 ServiceNow, Inc. All rights reserved.

19. Now create a change against an Application CI to test. Use **apache linux den 200**.

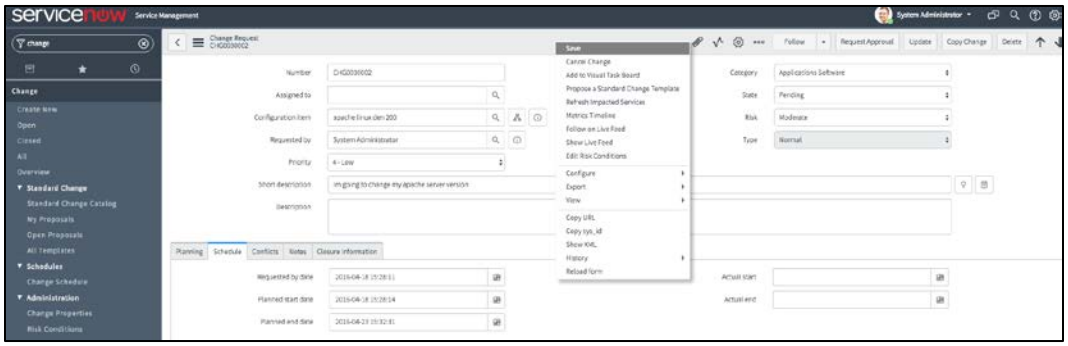


20. Select **Create New Change**, **Normal**.



21. In the Configuration Item select **apache linux den 200**.

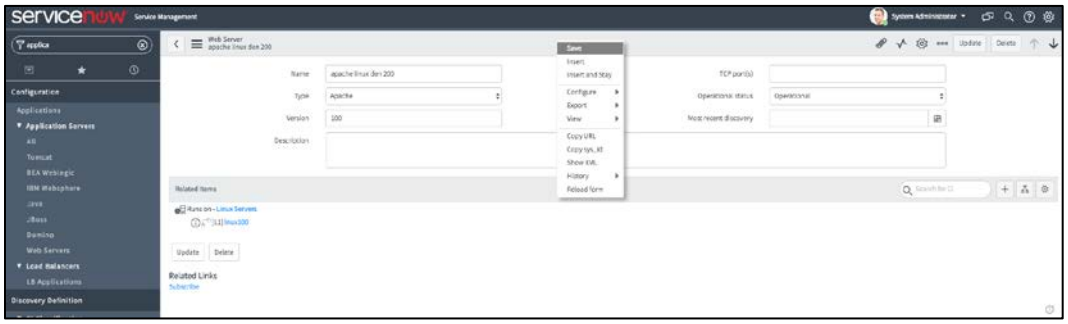
22. Select the schedule tab and select **today** in the **Planned start date** and sometime in the future for the **Planned end date**.



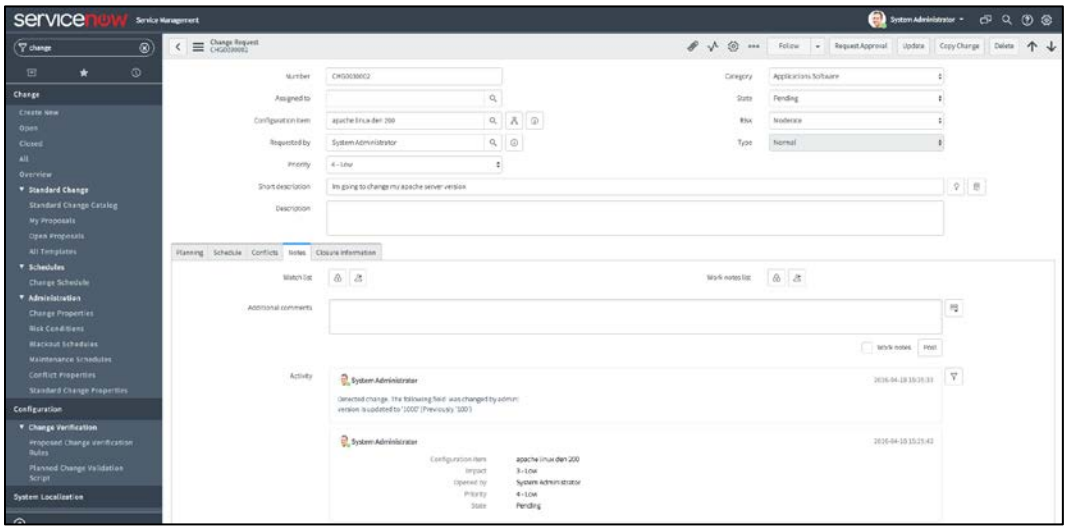
23. Select **Save** or **Submit**.

©2016 ServiceNow, Inc. All rights reserved.

24. Select the Apache application in applications and make some change to the record. Here we have changed the version to 1000, but any change to the record will work. Click **Save** or **Submit**.



25. See that your existing change was updated!



®2016 ServiceNow, Inc. All rights reserved.