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

Lab Guide

Scripting 301: Advanced JavaScript for ServiceNow

Oliver Schmitt & Matthias Martini

Login / Passwords:

admin / Knowledge15

itil / Knowledge15

employee / Knowledge15

This

Page

Intentionally

Left

Blank

Lab Goal

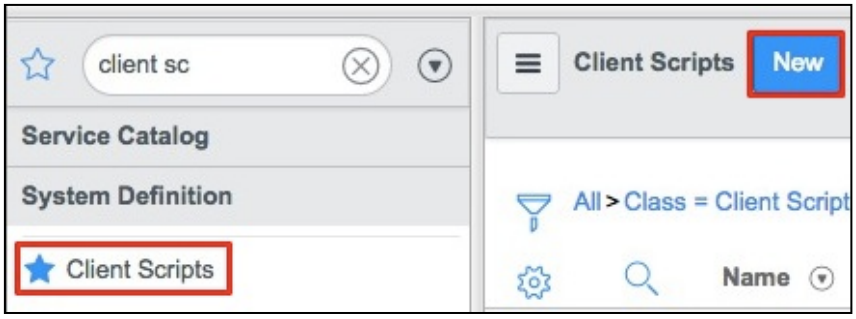
A simple **GlideAjax** call to retrieve information from the server and fill the short description of an existing ticket into the current one on the problem table.

This lab explains how to use the **GlideAjax** call to fill information from an existing ticket into a new one.

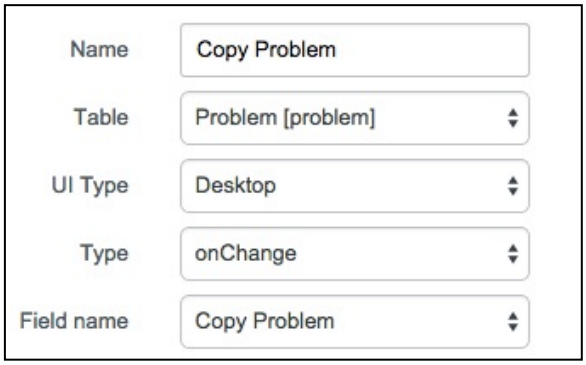
Lab 1.1
Simple GlideAjax

Use GlideAjax

1. Log on to a provided instance URL with the credentials on the cover.
2. Navigate to **System Definition > Client Scripts** and create a new Client Script.



3. The type should be **onChange** and the field name **Copy Problem** as well as the name.



©2015 ServiceNow, Inc. All rights reserved.

4. The script should call a server side script include and set the short description on return:

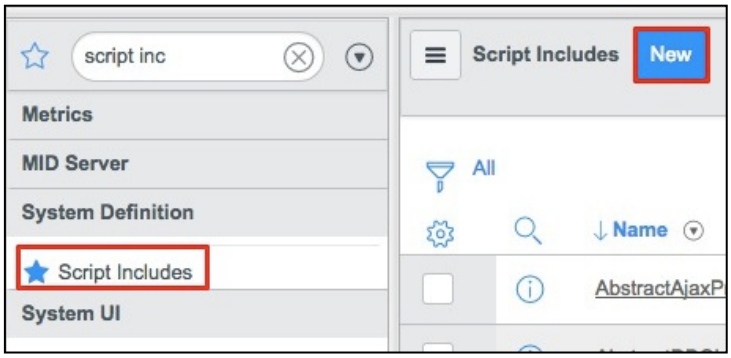
```
function onChange(control, oldValue, newValue, isLoading, isTemplate)
{
   if (isLoading || newValue == '') {
      return;
   }
   //send ajax
   var ga = new GlideAjax('AJAXProblemUtils');
   ga.addParam('sysparm_name','getAttributes');
   ga.addParam('sysparm_prb_id', newValue);
   ga.getXML(handleResponse);
}

function handleResponse(response)
{
      //process return
   var answer = response.responseXML.documentElement.getAttribute("answer");
   if (answer)
   {
         g_form.setValue("short_description", answer);
   }
}
```

Result: The Ajax call sends the **sysid** of the selected PRB to the server and executes a script include called **AJAXProblemUtils** with the function **getAttributes**.

©2015 ServiceNow, Inc. All rights reserved.

5. Navigate to **System Definition > Script Includes** and create a new script.



6. Set the following attributes for the new script include:

   Name**: AJAXProblemUtils**
   Client callable: **true**
   Description: **Server side functions for Problem Management**
   Script:

```
var AJAXProblemUtils = Class.create();

AJAXProblemUtils.prototype =  Object.extendsObject(AbstractAjaxProcessor, {

  getAttributes: function ()
  {
    //get parameter
       var prb = this.getParameter("sysparm_prb_id");
    //retrieve information
    var gr = new GlideRecord("problem");
    if (gr.get(prb))
    {
         return gr.getValue("short_description");
    }
  },

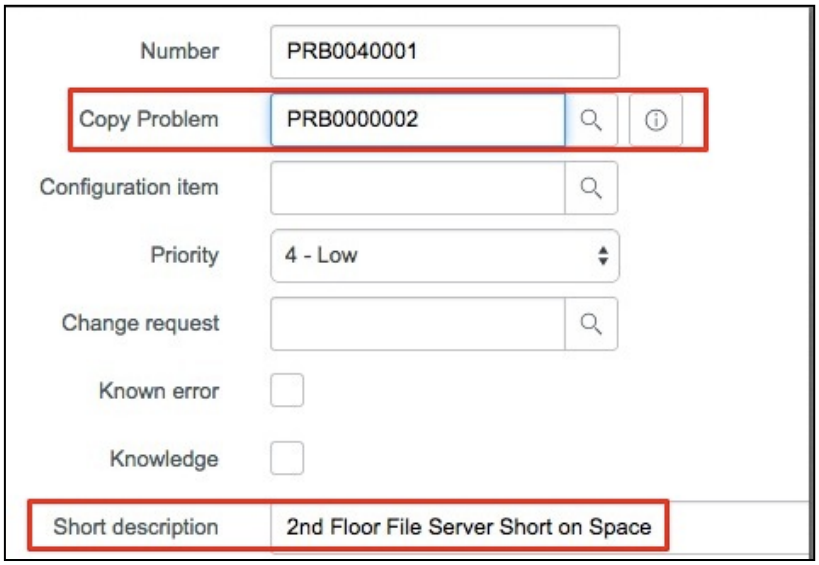
  type: "AJAXProblemUtils"
});
```

   Result: With a given **sysid**, the record is fetched from the database and the short description is returned to the client.

®2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Open a new or existing problem record and your screen should look the example:



®2015 ServiceNow, Inc. All rights reserved.

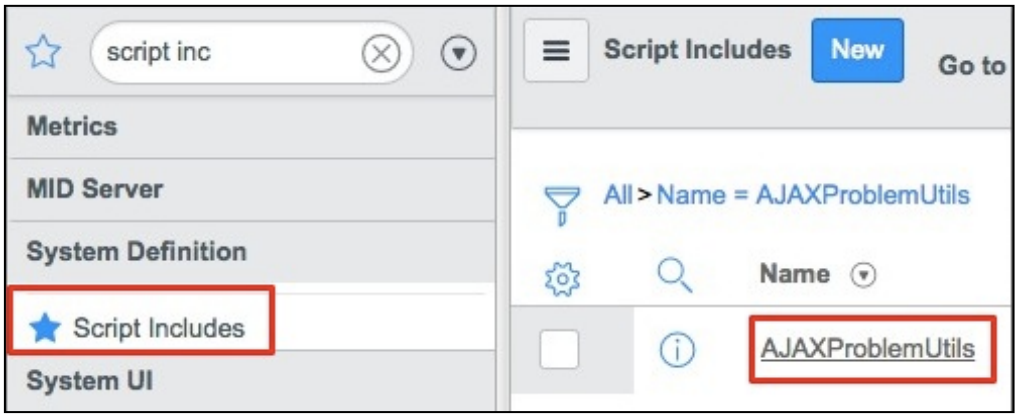
Lab Goal

This lab explains how to retrieve complex structures from the server and display it on the client side. To do this, it is necessary to extend the existing script in the following way.

Lab 1.2
GlideAjax and JSON

GlideAjax and JSON

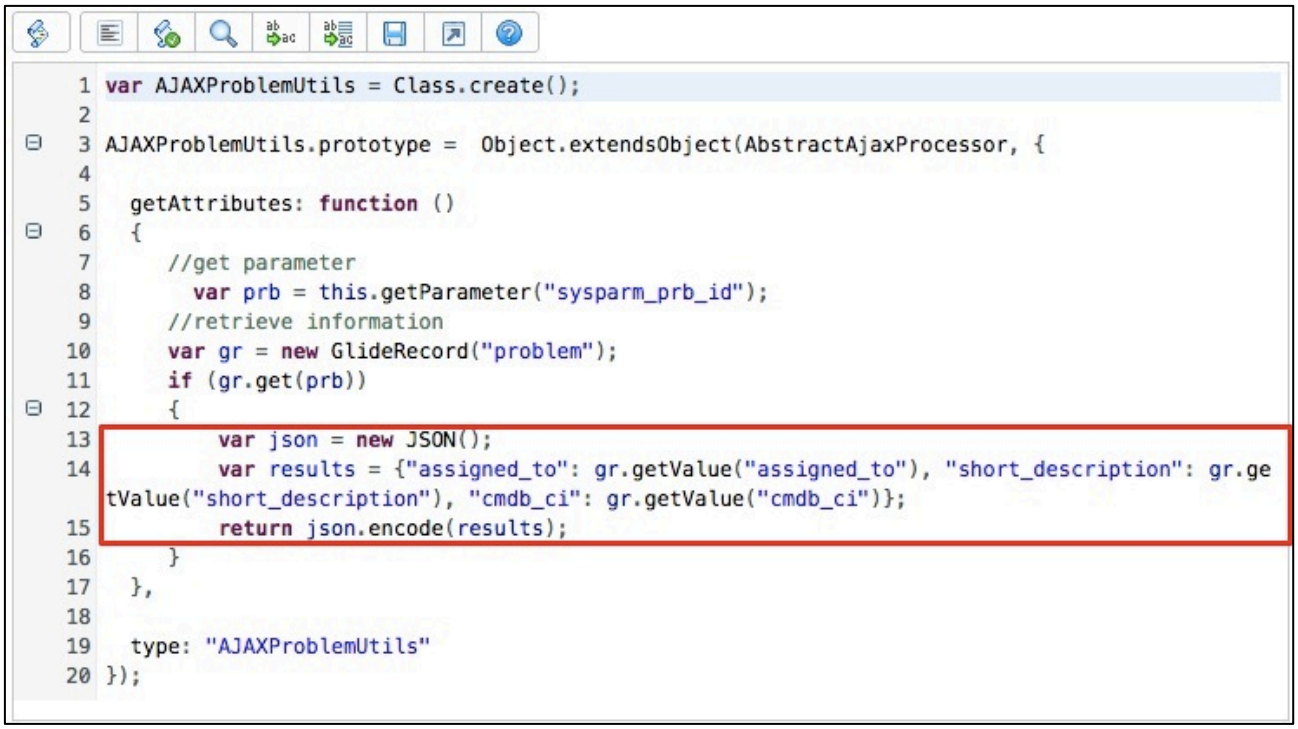
1. Log on to a provided instance URL with the credentials on the cover.
2. Navigate to **System Definition > Script Includes** and open the existing **AJAXProblemUtils**.



©2015 ServiceNow, Inc. All rights reserved.

3. Extend the script with this code to match the example.

```
            var json = new JSON();
            var results = {'assigned_to': gr.getValue('assigned_to'),
'short_description': gr.getValue('short_description'), 'cmdb_ci':
gr.getValue('cmdb_ci')};
            return json.encode(results);
```

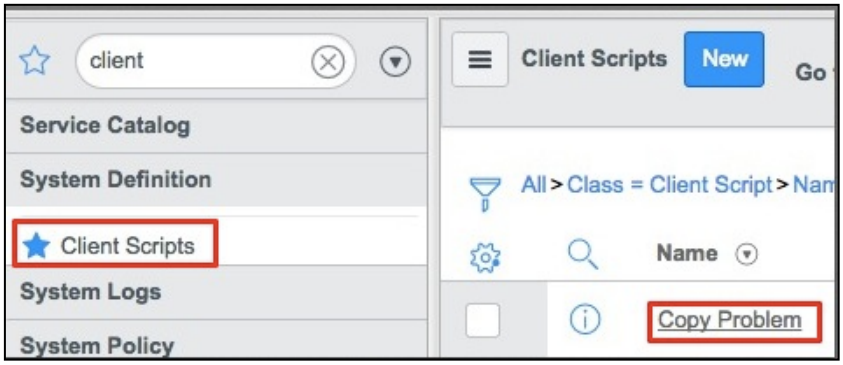


```
1  var AJAXProblemUtils = Class.create();
2
3  AJAXProblemUtils.prototype =  Object.extendsObject(AbstractAjaxProcessor, {
4
5    getAttributes: function ()
6    {
7       //get parameter
8         var prb = this.getParameter("sysparm_prb_id");
9       //retrieve information
10      var gr = new GlideRecord("problem");
11      if (gr.get(prb))
12      {
13          var json = new JSON();
14          var results = {"assigned_to": gr.getValue("assigned_to"), "short_description": gr.ge
    tValue("short_description"), "cmdb_ci": gr.getValue("cmdb_ci")};
15          return json.encode(results);
16      }
17   },
18
19   type: "AJAXProblemUtils"
20 });
```

Result: With the JSON Object, it is possible to encode an associative array as a single string and return multiple values in a possible complex structure. In this example the assigned person, short description, and the selected configuration item are returned at once.

©2015 ServiceNow, Inc. All rights reserved.

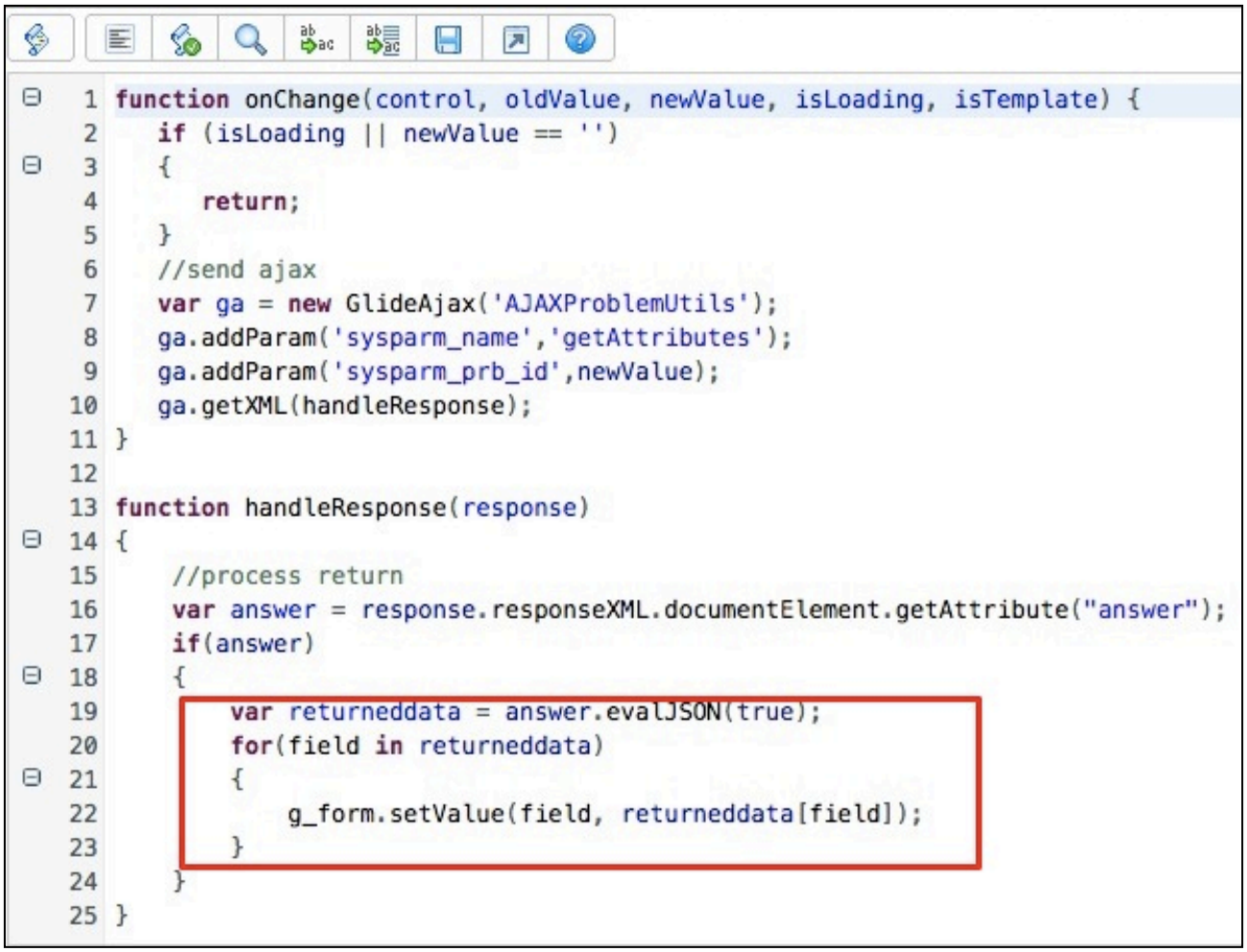
4. Navigate to **System Definition > Client Scripts** and open the existing script on the problem table with the name **Copy Problem.** It must be extended in order to handle the changes made on the server side.



®2015 ServiceNow, Inc. All rights reserved.

5. Change the script to match the example.

```
var returneddata = answer.evalJSON(true);
for(field in returneddata)
{
    g_form.setValue(field, returneddata[field]);
}
```



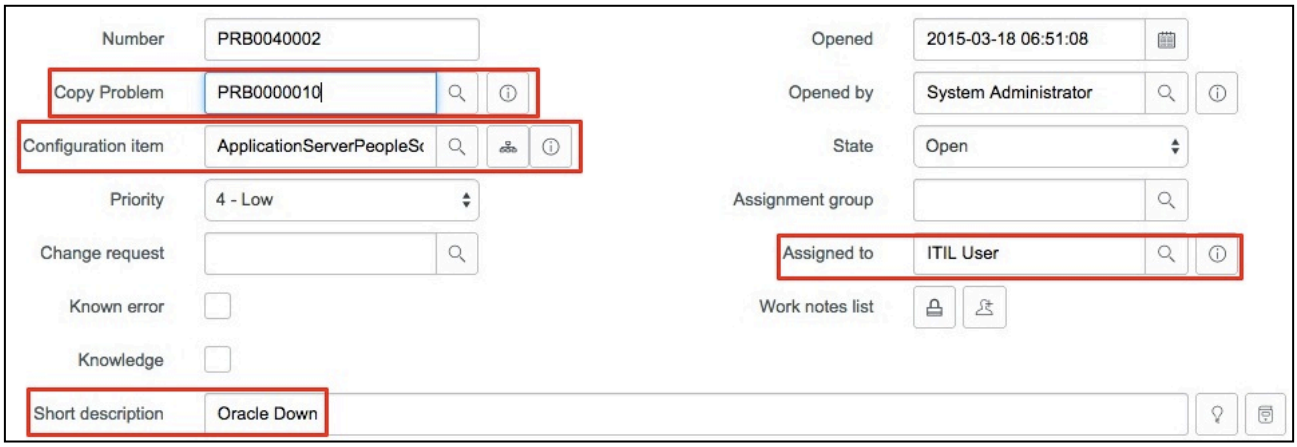
```
1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2     if (isLoading || newValue == '')
3     {
4        return;
5     }
6     //send ajax
7     var ga = new GlideAjax('AJAXProblemUtils');
8     ga.addParam('sysparm_name','getAttributes');
9     ga.addParam('sysparm_prb_id',newValue);
10    ga.getXML(handleResponse);
11 }
12
13 function handleResponse(response)
14 {
15     //process return
16     var answer = response.responseXML.documentElement.getAttribute("answer");
17     if(answer)
18     {
19        var returneddata = answer.evalJSON(true);
20        for(field in returneddata)
21        {
22            g_form.setValue(field, returneddata[field]);
23        }
24     }
25 }
```

Result: Each field automatically sets if added to the server side script. The client does not need other changing, and additional fields can be added. The **evalJSON** function transforms the string into an object.

©2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Open a new or existing problem record. The screen should look like the example.



©2015 ServiceNow, Inc. All rights reserved.

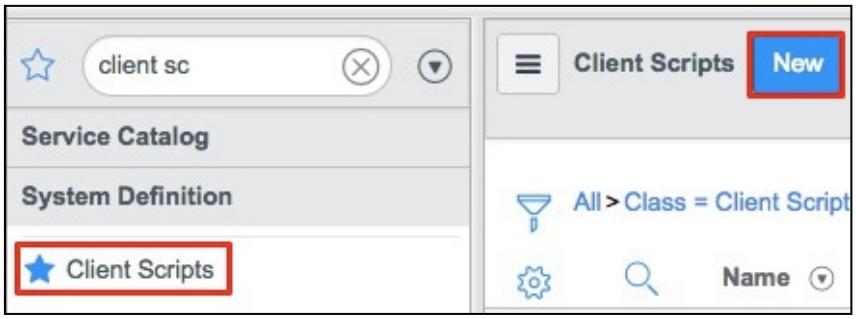
Lab Goal

Script Includes you to build a hierarchy of reusable code. This lab explains an approach to building a hierarchy of reusable code on the client side.

Lab 2.1
Client Script Inheritance

Client Script Inheritance

1. Log on to a provided instance URL with the credentials on the cover.
2. Navigate to **System Definition > Client Scripts** and create a new Client Script.



®2015 ServiceNow, Inc. All rights reserved.

3. Use these settings for the new record:

   Name: **Script Library**
   Type: **onLoad**
   Table: **Task [task]**
   Inherited: **true**
   Script:

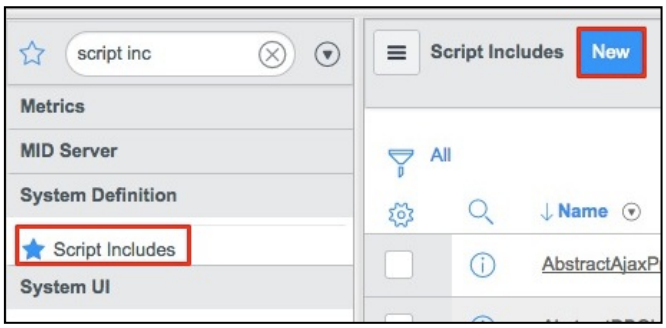
```
function onLoad() {
   //leave empty just to load the library
}

function checkForVIPUser(sysid)
{
   //send ajax
   var ga = new GlideAjax('AJAXTaskUtils');
   ga.addParam('sysparm_name','getVIP');
   ga.addParam('sysparm_user_id', sysid);
   ga.getXMLWait(); //sync call to get the answer

   var vip = ga.getAnswer();
   if(vip == 'true')
   {
         return true;
   }
}
```

   Result: By checking the **Inherited** field, this script is loaded on every child table of the task table and can be called. Leaving the **onLoad** function empty simply does not call something. All other functions are deployed to the scope of the form.

4. Navigate to **System Definition > Script Includes** and create a new script.



©2015 ServiceNow, Inc. All rights reserved.

5. Set the following attributes for the new script include:

   Name: **AJAXTaskUtils**
   Client Callable: **true**
   Description: **Server side functions for Task**
   Script:

```
var AJAXTaskUtils = Class.create();

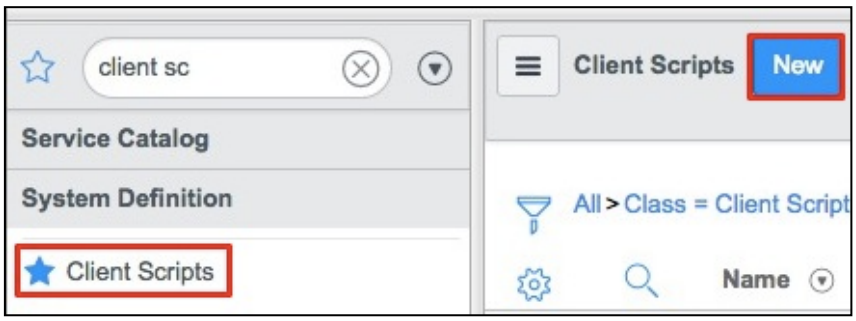
AJAXTaskUtils.prototype =  Object.extendsObject(AbstractAjaxProcessor, {

  getVIP: function()
  {
    //get parameter
       var id = this.getParameter("sysparm_user_id");
    //retrieve information
    var gr = new GlideRecord("sys_user");
    if (gr.get(id))
    {
          if(gr.getValue("vip") == true)
          {
               return "true";
          }
          else
          {
               return "false";
          }
    }
  },

  type: "AJAXTaskUtils"
});
```

   Result: Based on the users **sysid** the **VIP** status is returned to the client.

6. This library can now be used to call it on every other child table of task. Therefore navigate to **System Definition > Client Scripts** and create a new script.



©2015 ServiceNow, Inc. All rights reserved.

7. Set the following attributes for the new client script:

   Name: **Check for VIP**
   Table: **Change Request [change_request]**
   Field name: **Requested by**
   Script:

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

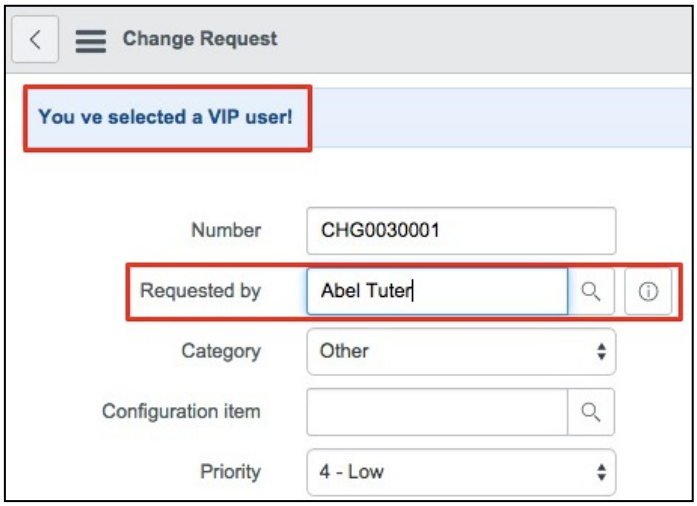
   if(checkForVIPUser(newValue))
   {
         g_form.addInfoMessage("You ve selected a VIP user!");
   }
}
```

   Result: We use the library function to check for the VIP. The same function can be used on the incident, problem, **change_task** (etc.) table.

©2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Open a new or existing change record. The screen should look like the example.



Note: Make sure that the user you selected has the VIP flag checked on the **User [sys_user]** table.

®2015 ServiceNow, Inc. All rights reserved.

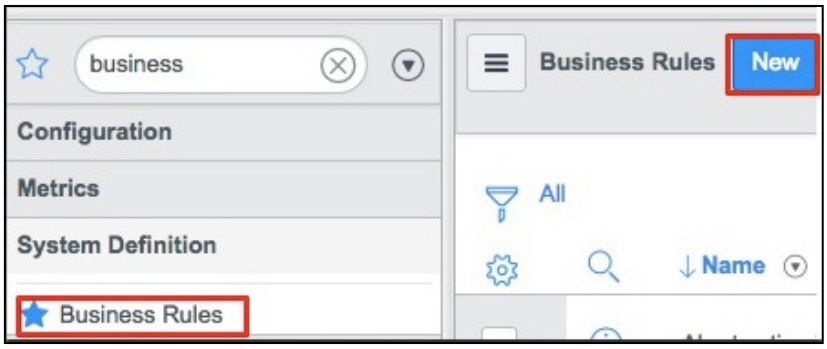
Lab Goal

This lab explains how to filter available server states and how to use a scratchpad to transport them. A server-side calculation in an **onDisplay** business rule is performed to filter the available states on the problem form. The scratchpad is used to transport the states, which are hidden.

Lab 3.1 onDisplay BR and Scratchpad

Using onDisplay Business Rules and the Scratchpad

1. Log on to a provided instance URL with the credentials on the cover.
2. Navigate to **System Definition > Business Rules** and create a new Business rule.



©2015 ServiceNow, Inc. All rights reserved.

3. Set up the following information:
   Name: **Filter Problem state**
   Table: **Problem [problem]**
   When: **display**
   Condition: **current.state.canWrite();**
   Script:

```
hideProblemStates(current);

function hideProblemStates(current)
{
   var hidden_states = [];

   if (current.state != '3' && !_activeChange())
   {
      hidden_states.push('3'); //Remove Pending Change
   }
   if (current.state != '4' && (!gs.hasRole('itil_admin') || _activeProbTasks()))
   {
      hidden_states.push('4'); //Remove Closed/Resolved
   }
   //save the states to hide in the scratchpad
   g_scratchpad.remove_states = hidden_states;
}

//check for an existing active RfC
function _activeChange(){
   if(current.rfc=='')
      return false;

   var chg = new GlideRecord('change_request');
   if (!chg.get(current.rfc))
      return false;

   return chg.active;
}
//check for existing and active Problem tasks
function _activeProbTasks(){
   var tsk = new GlideRecord('problem_task');
   tsk.addQuery('problem', current.sys_id);
   tsk.addQuery('active',true);
   tsk.query();
   if(tsk.next())
      return true;

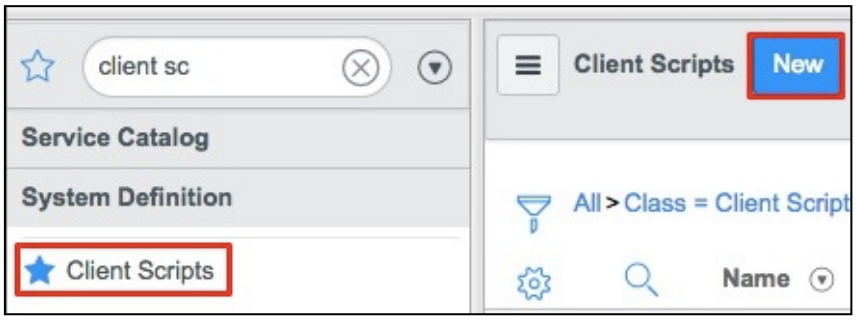
   return false;
}
```

©2015 ServiceNow, Inc. All rights reserved.

Result: The script can be added by typing **k15lab31** and pressing tab and does the following:

- **Pending Change** can be selected only if a Change Request is related to the record and still active.
- **Resolved/Closed** can only be selected by an ITIL Admin and if all Problem Tasks are closed.
- **Open** and **Known Error** can always be selected.

4. Navigate to **System Definition > Client Scripts** and create a new Client Script.



5. Set the script up with the following settings:

Name: **Hide States**
Table: **Problem [problem]**
Type: **onLoad**
Script:

```
function onLoad()
{
    var states = g_scratchpad.remove_states;

    if (!states)
       return;

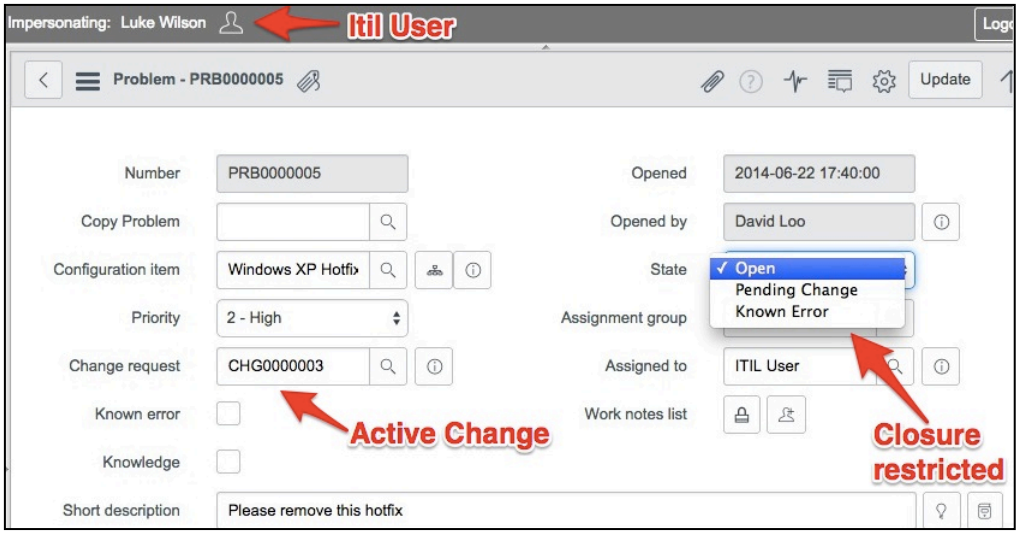
    for (var vI=0; vI < states.length; vI++){
        g_form.removeOption('state', states[vI]);
    }
}
```

Result: The client script picks up the **remove_states** variable, which was set in the **onDisplay** business rule. Based on the values in the variable, states that are not valid (based on the conditions) are removed and no longer selectable.

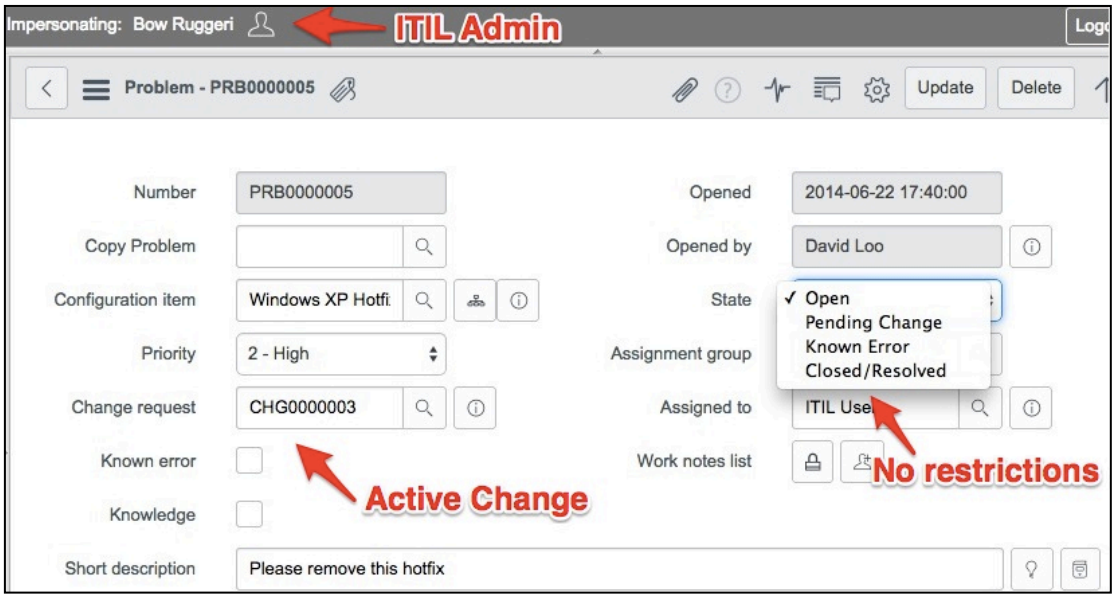
©2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Impersonate **Luke Wilson**. You should see that Luke is prohibited from closing the problem, because he does not have the **itil_admin** role. Since there is an active associated Change Request, **Pending Change** is an available state option.

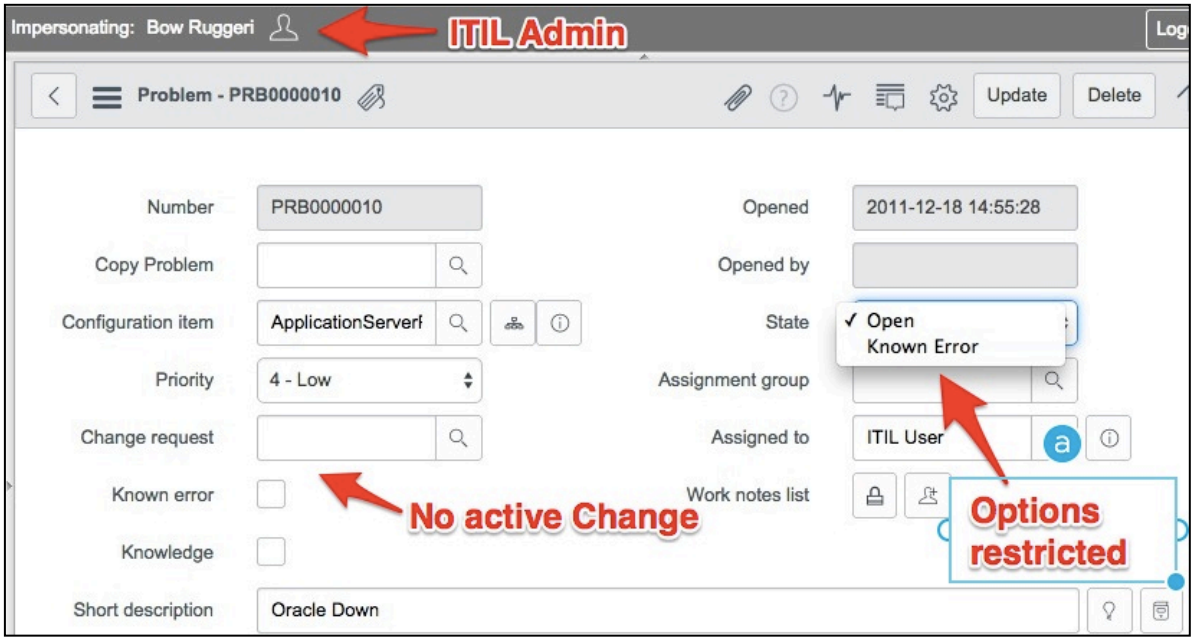


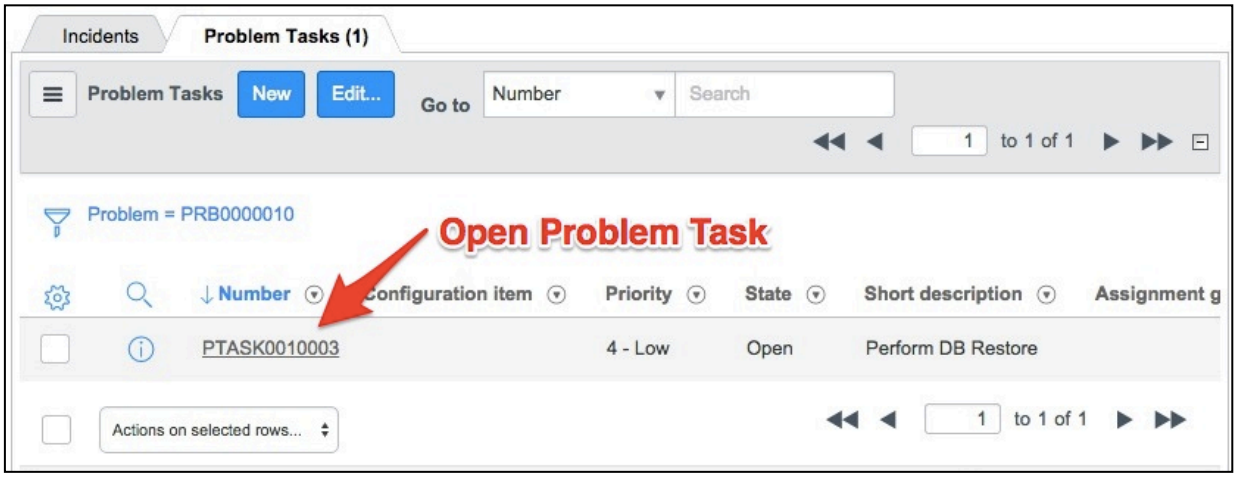
2. Impersonate **Bow Ruggeri** and confirm that he can close the problem, because he has the **ITIL Admin** role. Since there is an active associated Change Request, the **Pending Change** is an available option.



©2015 ServiceNow, Inc. All rights reserved.

3. Open problem **PRB0000010**. Note that Bow Ruggeri cannot close the problem, because there is an open Problem Task. Since there is no associated Change Request, **Pending Change** is not an option.





©2015 ServiceNow, Inc. All rights reserved.

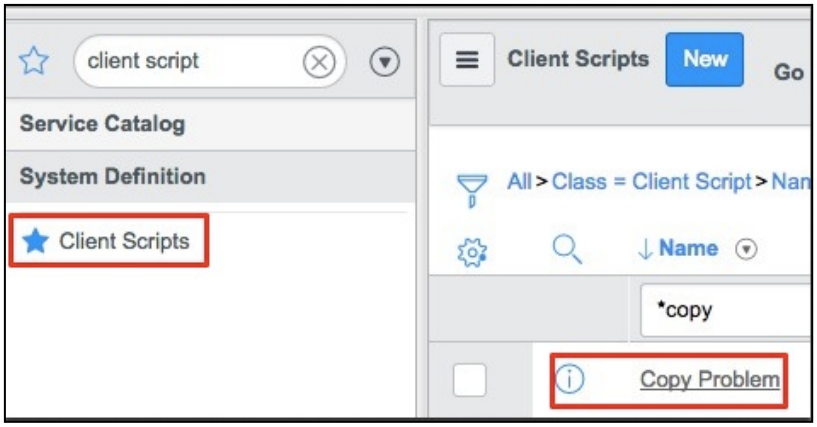
Lab Goal

This lab explains how to use the existing use case of copying values from one record to another with **GlideAjax** to copying problem tasks. When the callback returns, the related list should be reloaded without reloading the whole form.

Lab 4.1
Glide List

Using GlideList

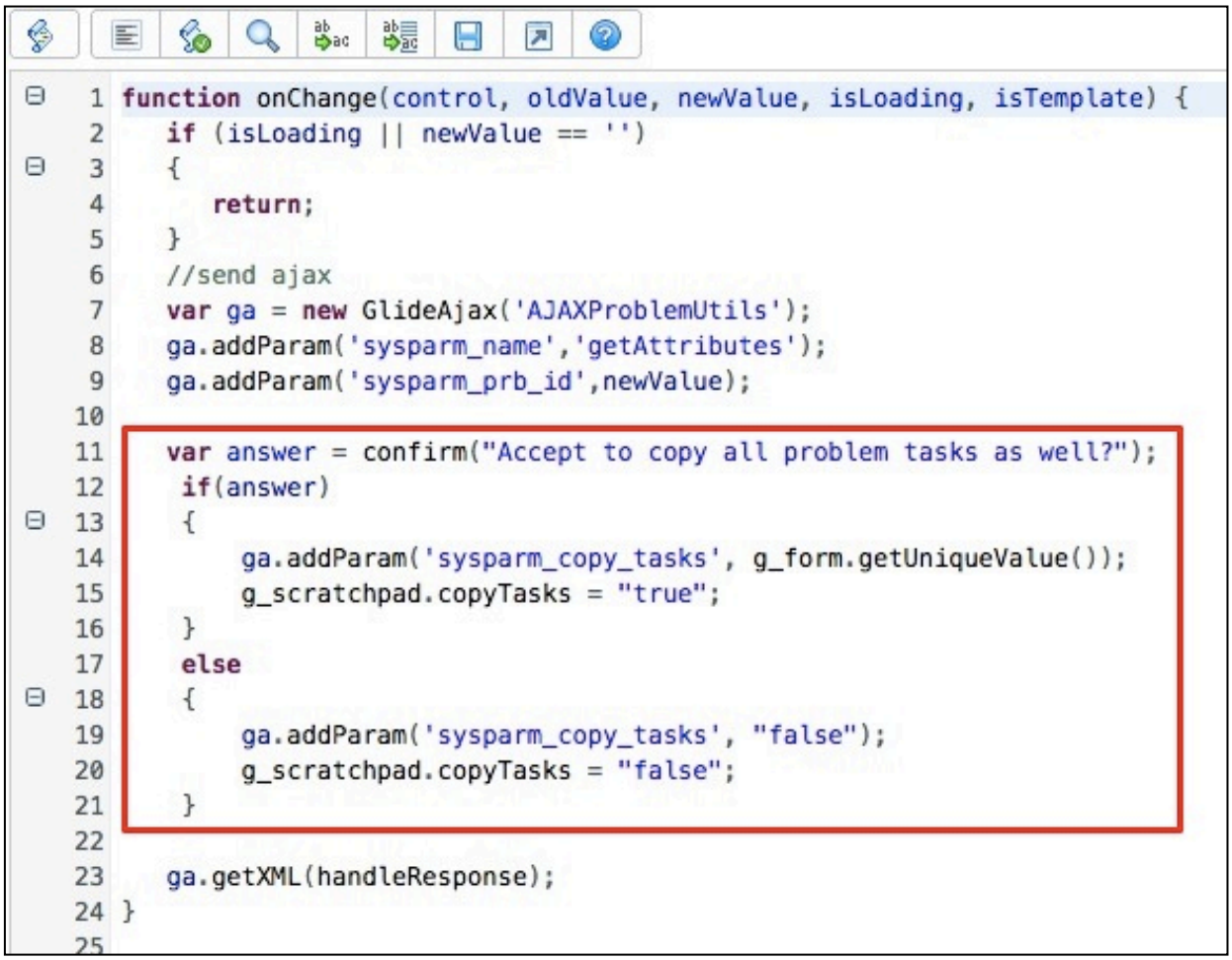
1. Log on to a provided instance URL with the credentials on the cover.
2. Navigate to **System Definition > Client Script** and open the script **Copy Problem**.



®2015 ServiceNow, Inc. All rights reserved.

3. Extend the script with this code to match the example.

```
var answer = confirm('Accept to copy all problem tasks as well?');
if(answer)
{
    ga.addParam('sysparm_copy_tasks', g_form.getUniqueValue());
    g_scratchpad.copyTasks = 'true';
}
else
{
    ga.addParam('sysparm_copy_tasks', 'false');
    g_scratchpad.copyTasks = 'false';
}
```

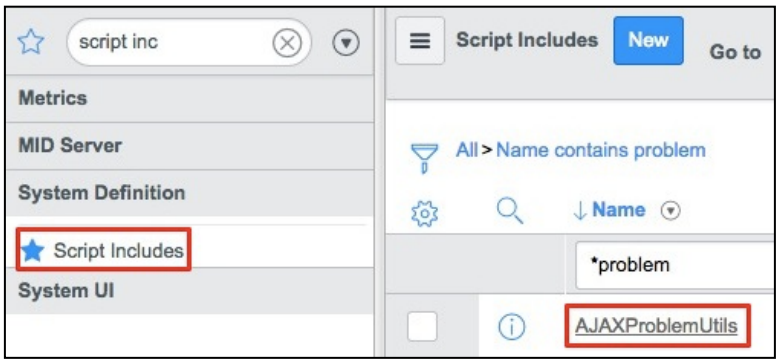


```
1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2     if (isLoading || newValue == '')
3     {
4        return;
5     }
6     //send ajax
7     var ga = new GlideAjax('AJAXProblemUtils');
8     ga.addParam('sysparm_name','getAttributes');
9     ga.addParam('sysparm_prb_id',newValue);
10
11    var answer = confirm("Accept to copy all problem tasks as well?");
12     if(answer)
13     {
14         ga.addParam('sysparm_copy_tasks', g_form.getUniqueValue());
15         g_scratchpad.copyTasks = "true";
16     }
17     else
18     {
19         ga.addParam('sysparm_copy_tasks', "false");
20         g_scratchpad.copyTasks = "false";
21     }
22
23    ga.getXML(handleResponse);
24 }
25
```

Result: A popup asks if all related problem tasks of the selected record should be copied. If the user agrees, a parameter is set in the Ajax call that identifies the record. The scratchpad variable helps to save the users answer in the callback function.

©2015 ServiceNow, Inc. All rights reserved.

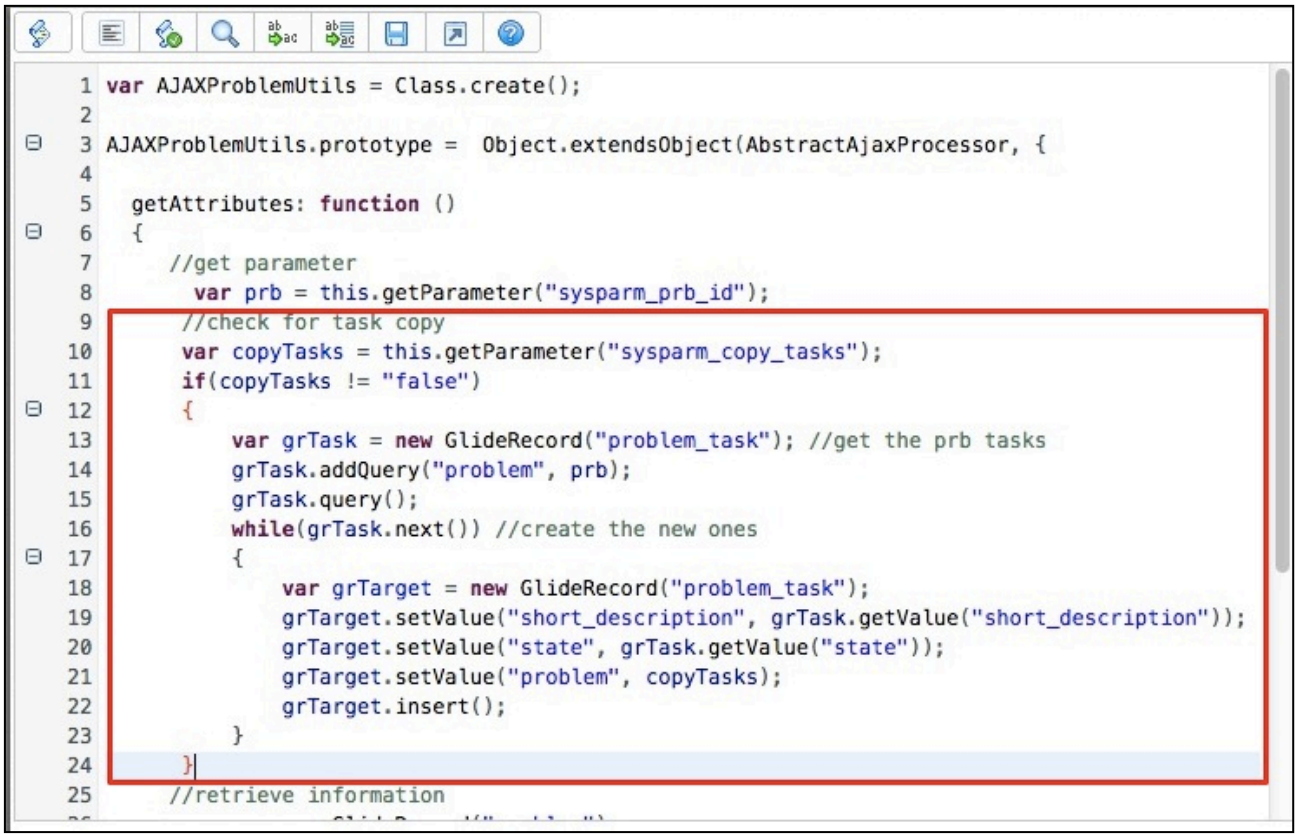
4. Navigate to **System Definition > Script Includes** and open the script **AJAXProblemUtils**.



©2015 ServiceNow, Inc. All rights reserved.

5. Extend the script with this code to match the example:

```
        //check for task copy
        var copyTasks = this.getParameter('sysparm_copy_tasks');
        if(copyTasks != "false")
        {
            var grTask = new GlideRecord('problem_task'); //get the prb task
            grTask.addQuery('problem', prb);
            grTask.query();
            while(grTask.next()) //create the new ones
            {
                var grTarget = new GlideRecord("problem_task");
                grTarget.setValue('short_description', grTask.getValue('short_description'));
                grTarget.setValue('state', grTask.getValue('state'));
                grTarget.setValue('problem', copyTasks);
                grTarget.insert();
            }
        }
```

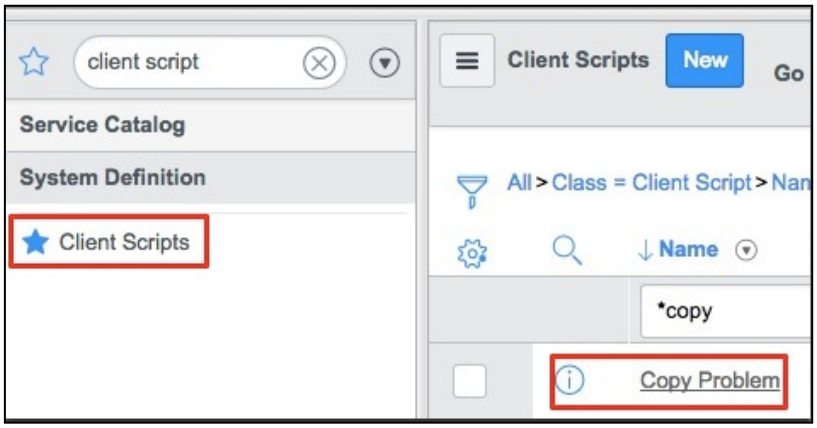


```
1  var AJAXProblemUtils = Class.create();
2
3  AJAXProblemUtils.prototype =  Object.extendsObject(AbstractAjaxProcessor, {
4
5    getAttributes: function ()
6    {
7       //get parameter
8         var prb = this.getParameter("sysparm_prb_id");
9       //check for task copy
10      var copyTasks = this.getParameter("sysparm_copy_tasks");
11      if(copyTasks != "false")
12      {
13          var grTask = new GlideRecord("problem_task"); //get the prb tasks
14          grTask.addQuery("problem", prb);
15          grTask.query();
16          while(grTask.next()) //create the new ones
17          {
18              var grTarget = new GlideRecord("problem_task");
19              grTarget.setValue("short_description", grTask.getValue("short_description"));
20              grTarget.setValue("state", grTask.getValue("state"));
21              grTarget.setValue("problem", copyTasks);
22              grTarget.insert();
23          }
24      }
25    //retrieve information
```

Result: If the **copy_task** parameter is set, all problem tasks are copied over.

©2015 ServiceNow, Inc. All rights reserved.

6. The last step is to go back to the "**Copy Problem**" Client Script to adjust the callback function. Navigate to **System Definition > Client Script** and open it.



7. The callback function has to be extended to match the example:

```
if (g_scratchpad.copyTasks == 'true') //reload list
{
    GlideList2.get('problem.problem_task.problem').refresh(1);
}
```

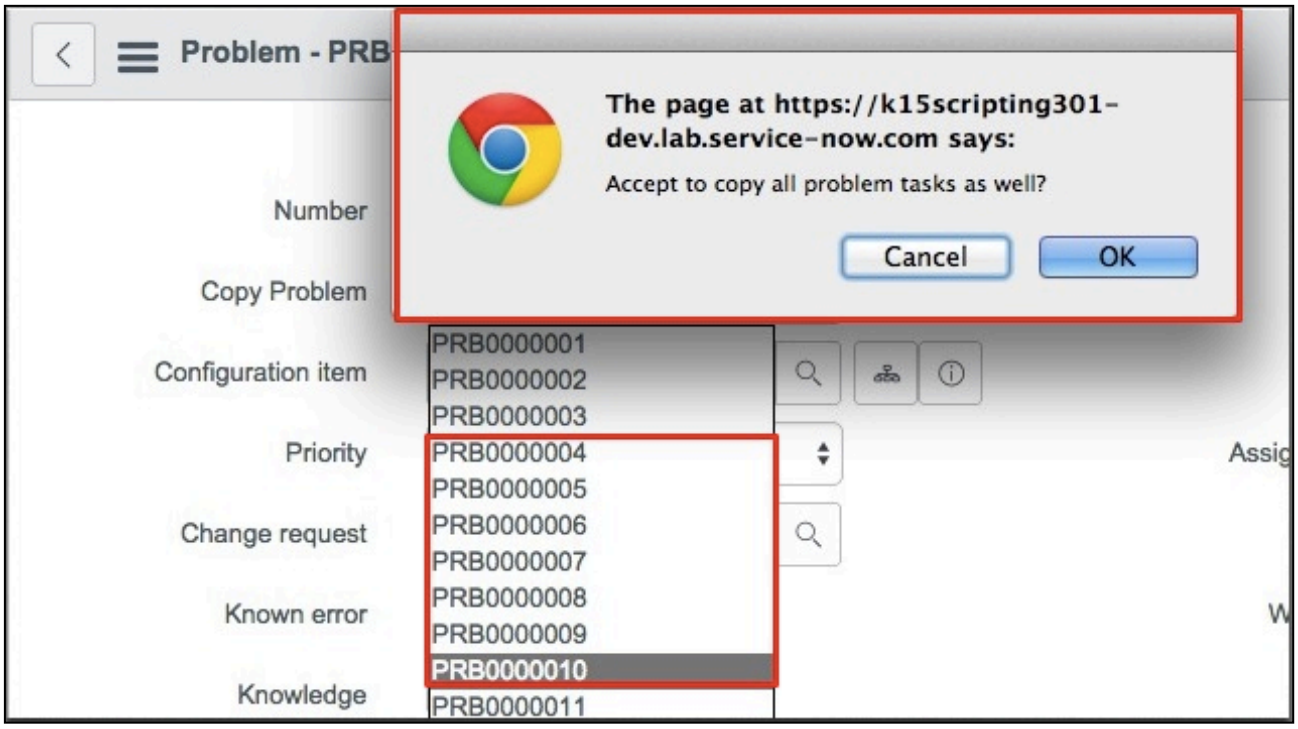
```
26 function handleResponse(response)
27 {
28     //process return
29     var answer = response.responseXML.documentElement.getAttribute("answer");
30     if(answer)
31     {
32         var returneddata = answer.evalJSON(true);
33         for(field in returneddata)
34         {
35             g_form.setValue(field, returneddata[field]);
36         }
37         if(g_scratchpad.copyTasks == "true") //reload list
38         {
39             GlideList2.get("problem.problem_task.problem").refresh(1);
40         }
41     }
42 }
```

Result: The result from the user input has been saved in the scratchpad variable and will be tested. **GlideList2** is a client-side object that allows the related lists to be controlled. The related list for the problem tasks is retrieved and reloaded. This updates only the reloaded list, not the whole form. The notation for a related list is **table.target_table.reference**.

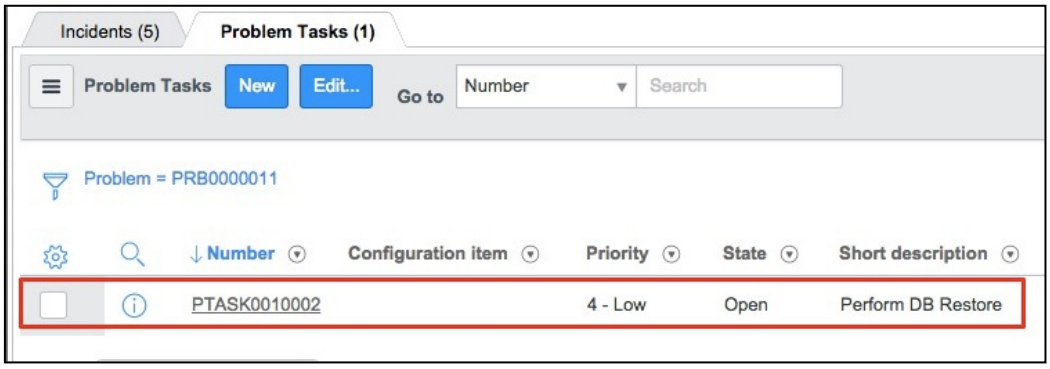
©2015 ServiceNow, Inc. All rights reserved.

Lab Success Verification

1. Open a problem record and enter the number of an existing problem that has related problem tasks into the problem record **Copy Problem** field.



2. Accept the popup and check the Problem Tasks related list.



©2015 ServiceNow, Inc. All rights reserved.