knowledge14®

April 27–May 1, 2014 • Moscone Center • San Francisco, CA

Lab Guide

Scripting 301:
Advanced JavaScript
for ServiceNow

Oliver Schmitt & Matthias Martini

Lab instance: https://**<<name>>**.lab.service-now.com

Login / Passwords:

admin / Knowledge14

itil / Knowledge14

employee / Knowledge14

Lab Goal

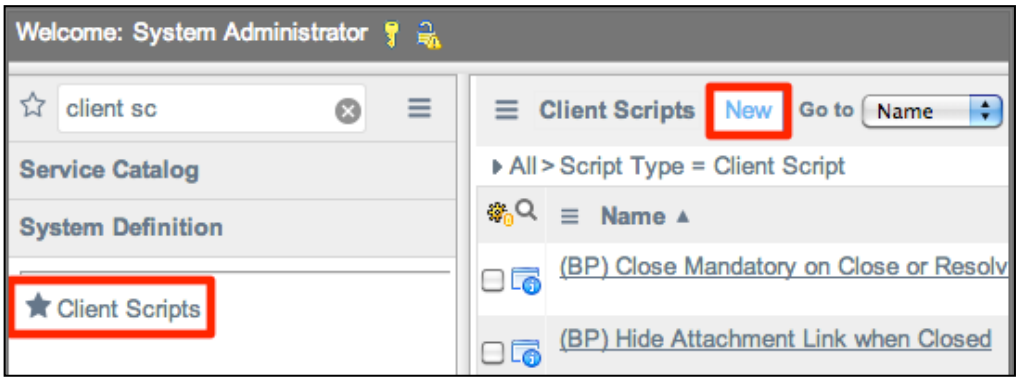
A simple GlideAjax call to retrieve information from the server and fill the short description of an existing ticket into the current one on the problem table.

Lab 1.1 Simple GlideAjax

Using GlideAjax

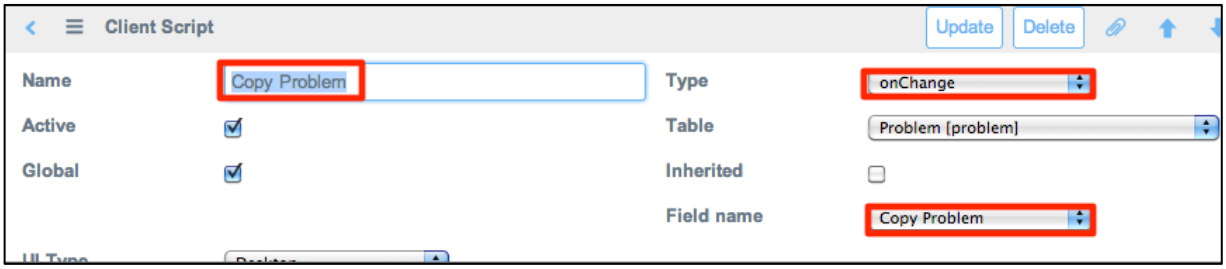
1. Login to the instance URL on the cover page using the provided **admin** credentials.

2. Navigate to **System Definition > Client Scripts** and click **New**.



©2014 ServiceNow, Inc. All rights reserved.

3. Set type to **onChange** and enter **Copy Problem** for the **Name** and **Field name**.



4. This script calls a server side script include and sets the short description on return:

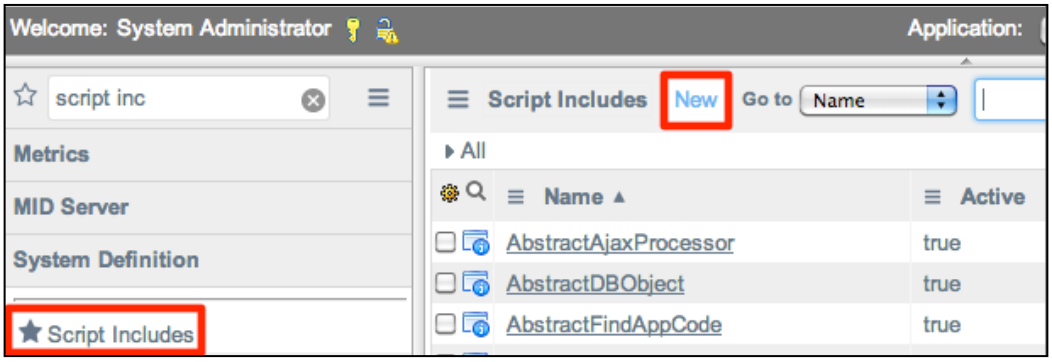
```
function onChange(control, oldValue, newValue, isLoading, isTemplate)
{
   if (isLoading || newValue == '') {
      return;
   }
   //send ajax
   var ga = new GlideAjax('AJAXProblemUtils');
   ga.addParam('sysparm_name','getAttributes');
   ga.addParam('sysparm_prb_id', newValue);
   ga.getXML(handleResponse);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

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

Result: The Ajax call will send the **`sysid`** of the selected PRB to the server and execute a script include called **`AJAXProblemUtils`** with the function **`getAttributes`**.

©2014 ServiceNow, Inc. All rights reserved.

5. Navigate to **System Definition > Script Includes** and click **New**.



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");
       }
  },

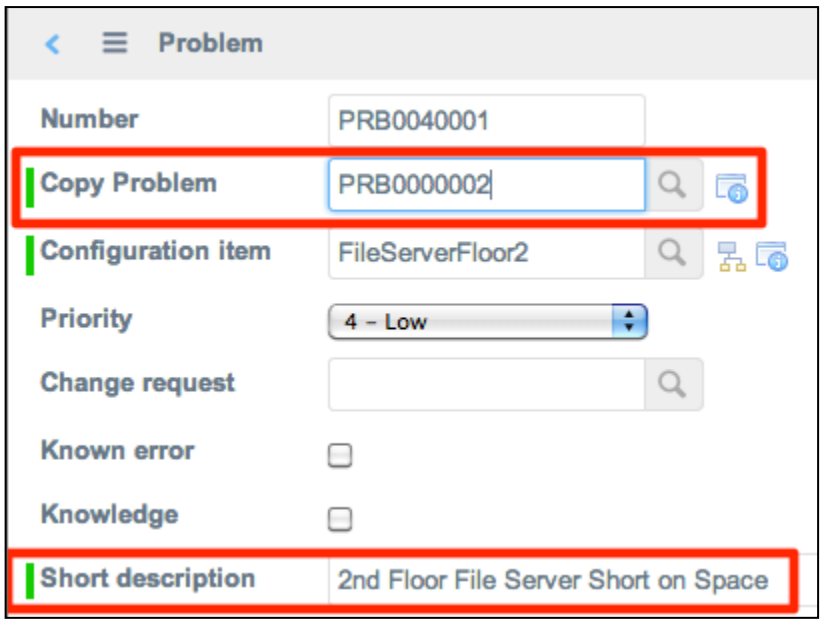
  type: "AJAXProblemUtils"
});
```

Result: With a given **`sysid`** the record will be fetched from the database and the short description will be returned to the client.

©2014 ServiceNow, Inc. All rights reserved.

Lab Success Verification

Open a new or existing problem record and your screen should look like this:



©2014 ServiceNow, Inc. All rights reserved.

Lab Goal

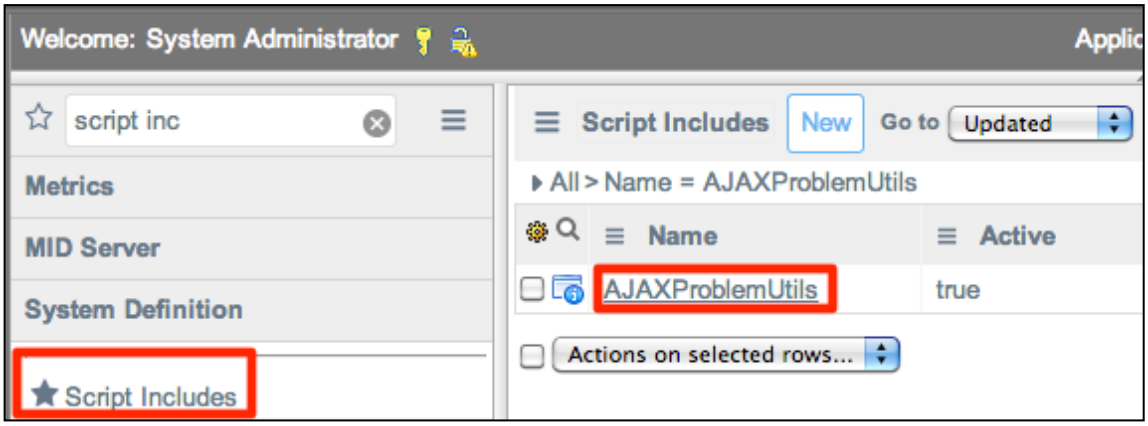
Retrieve complex structures from the server and display it on the client side by extending an existing script.

Lab 1.2
GlideAjax & JSON

GlideAjax and JSON

1. Login to the instance URL on the cover page using the provided **admin** credentials.

2. Navigate to **System Definition > Script Includes** and open **AJAXProblemUtils**.



©2014 ServiceNow, Inc. All rights reserved.

3. Extend the script with the following code:

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

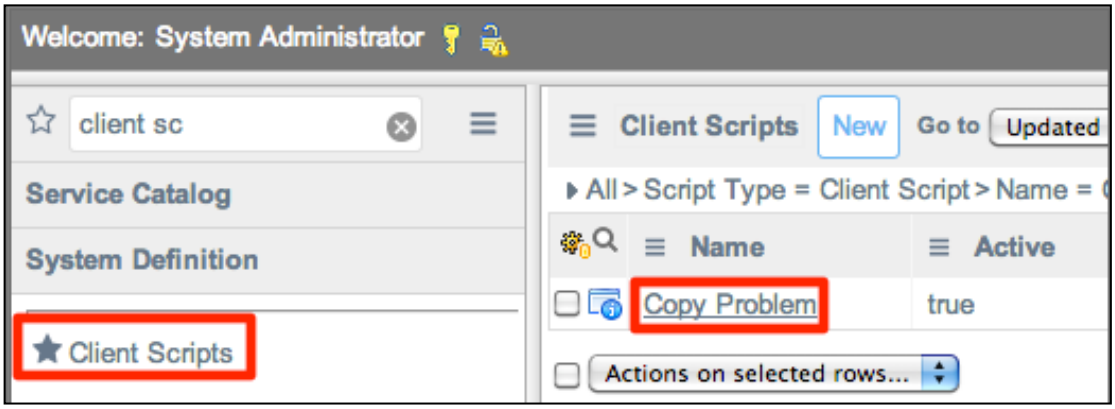
AJAXProblemUtils.prototype =  Object.extendsObject(AbstractAjaxProcessor, {

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

  type: "AJAXProblemUtils"
});
```

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.

4. Navigate to **System Definition > Client Scripts** and open **Copy Problem.** This script can be extended to handle the changes done on the server side.



©2014 ServiceNow, Inc. All rights reserved.

5. Change the script so it looks like the sample below:

Script:

```
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);
}

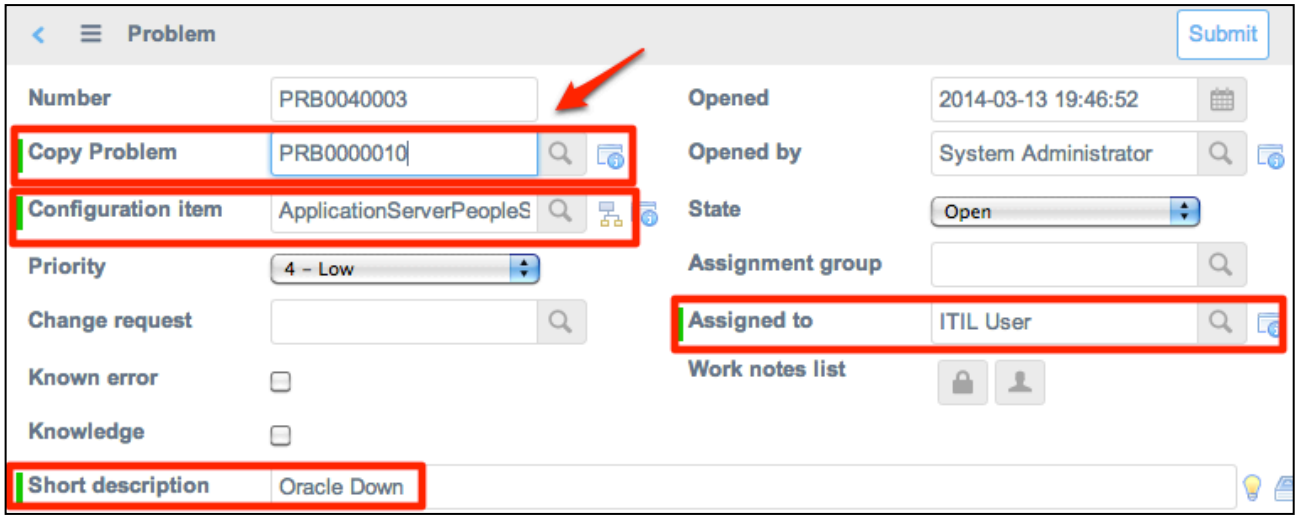
function handleResponse(response)
{
    //process return
    var answer = response.responseXML.documentElement.getAttribute("answer");
    if(answer)
    {
        var returneddata = answer.evalJSON(true);
        for(field in returneddata)
        {
            g_form.setValue(field, returneddata[field]);
        }
    }
}
```

Result: Each field will automatically be set if added to the server side script. The client does not need changing anymore and additional fields can be added. The **`evalJSON`** function will transform the string into an object.

©2014 ServiceNow, Inc. All rights reserved.

Lab Success Verification

Open a new or existing problem record and your screen should look like this:



©2014 ServiceNow, Inc. All rights reserved.

Lab Goal

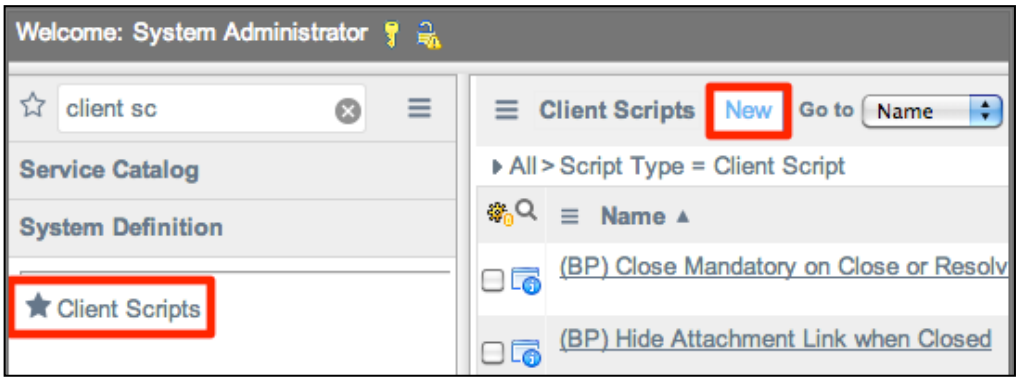
Script Includes allow you to build a hierarchy of reusable code. The goal of this lab is to show an approach for how this can be built on the client side as well.

Lab 2.1
Client Script Inheritance

Client Script Inheritance

1. Login to the instance URL on the cover page using the provided **admin** credentials.

2. Navigate to **System Definition > Client Scripts** and click **New**.



©2014 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
}

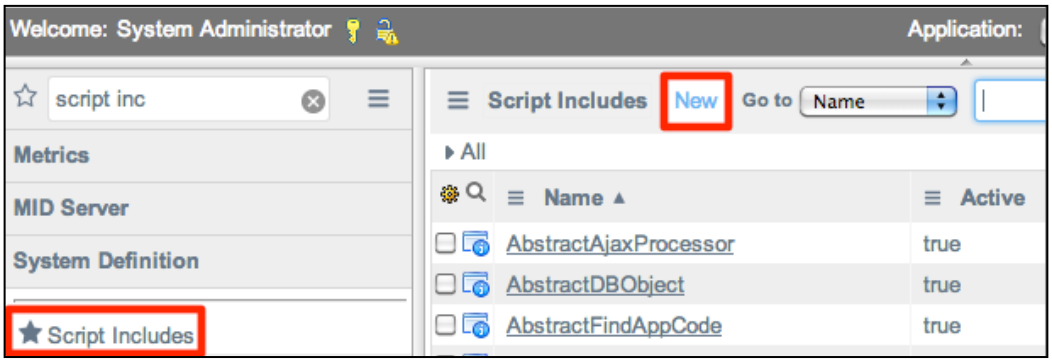
function checkForVIPUser(sysid)
{
   //send ajax
   var ga = new GlideAjax('AJAXTaskUtils');
   ga.addParam('sysparm_name','getVIP');
   ga.addParam('sysparm_user_id', sysid);
   ga.getXMLWait(); //sync call to get the answer

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

Result: By checking the **Inherited** field this script will be 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 will get deployed to the scope of the form.

©2014 ServiceNow, Inc. All rights reserved.

4. Navigate to **System Definition > Script Includes** and click **New**.



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, {

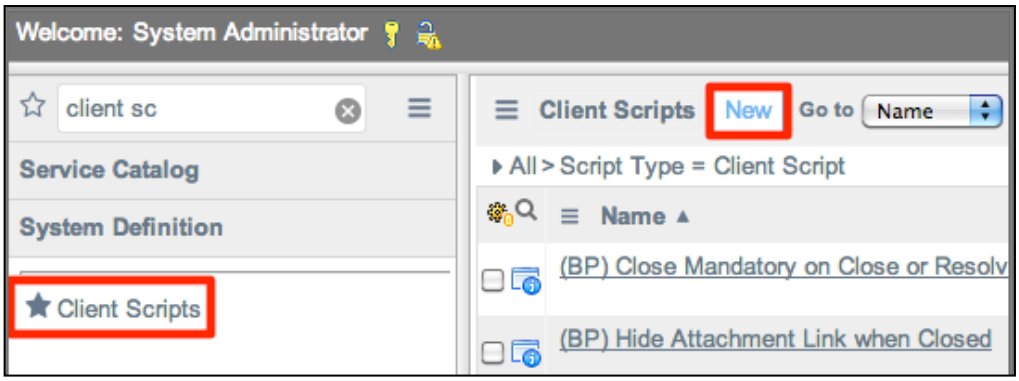
  getVIP: function()
  {
       //get parameter
       var id = this.getParameter("sysparm_user_id");
       //retrieve information
       var gr = new GlideRecord("sys_user");
       if (gr.get(id))
       {
            if(gr.getValue("vip"))
            {
                 return "true";
            }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

  type: "AJAXTaskUtils"
});
```

Result: Based on the users **`sysid`** the VIP status will be returned to the client.

©2014 ServiceNow, Inc. All rights reserved.

6. This library can now be used to call every other child table of task. To use it, navigate to **System Definition > Client Scripts** and click **New**.



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;
   }

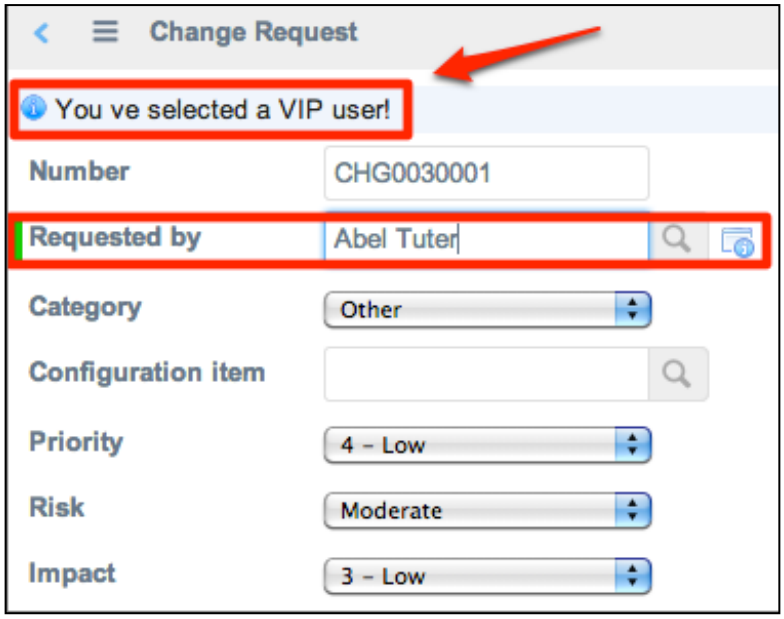
   if(checkForVIPUser(newValue))
   {
      g_form.addInfoMessage("You ve selected a VIP user!");
   }
}
```

Result: The library function checks to see if a user is a VIP. The same function could be used on the incident, problem, change_task etc. table.

©2014 ServiceNow, Inc. All rights reserved.

Lab Success Verification

Open a new or existing change record and your screen should look like this:



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

©2014 ServiceNow, Inc. All rights reserved.

Lab Goal

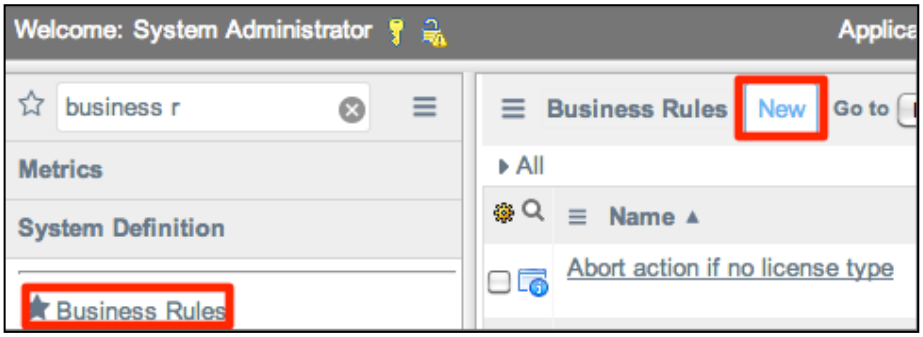
A server side calculation in an onDisplay business rule will be performed to filters available states on the problem form. The scratchpad will be used to transport the states that will be hidden.

Lab 3.1 onDisplay BR & scratchpad

Using onDisplay Business Rules and the scratchpad

1. Login to the instance URL on the cover page using the provided **admin** credentials.

2. Navigate to **System Definition > Business Rules** and click **New**.



©2014 ServiceNow, Inc. All rights reserved.

3. Setup the following:

   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;

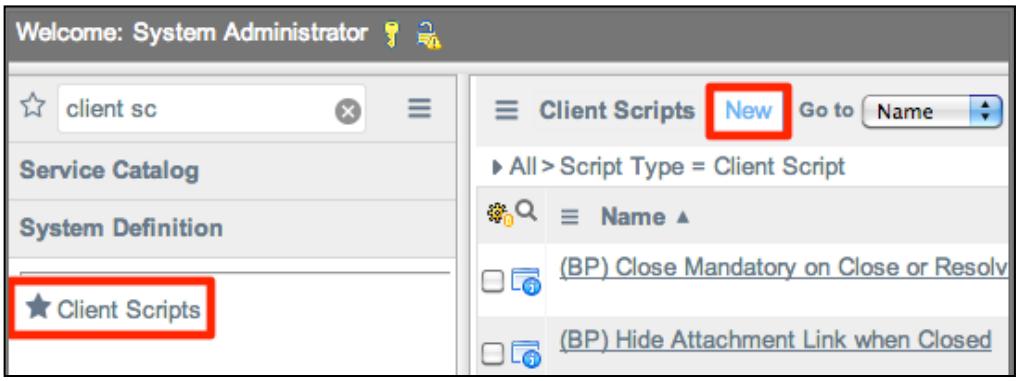
   return false;
}
```

©2014 ServiceNow, Inc. All rights reserved.

Result: The script can be found in **KB0010006** on the Knowledge Base and does the following:

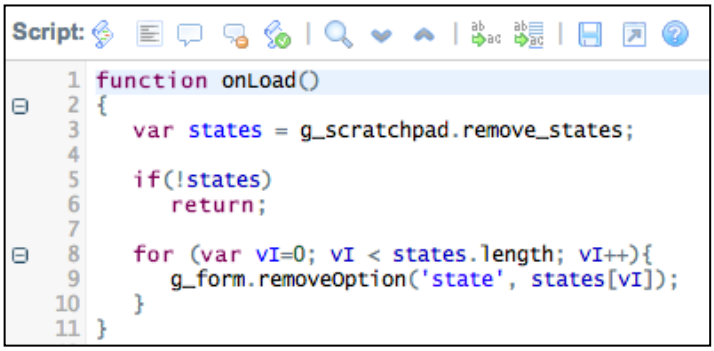
- **Pending Change** can be selected only if a Change Request is related to the record and still active.
- **Resolved/Closed** can only be selected by an ITIL Admin and if all Problem Tasks are closed.
- **Open & Known Error** can always be selected.

4. Navigate to **System Definition > Client Scripts** and click **New**.



5. Set the script up with the following settings:

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

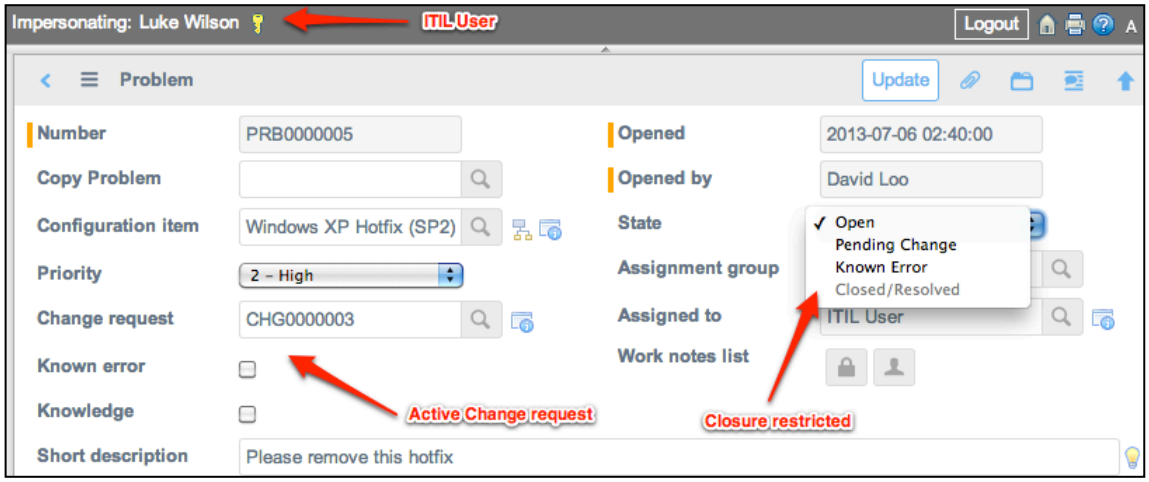


   Result: The client script will pick up the **remove_states** variable, which was set in the **onDisplay** business rule. Based on the values in the variable the states that are not valid based on the conditions are removed and not selectable anymore.

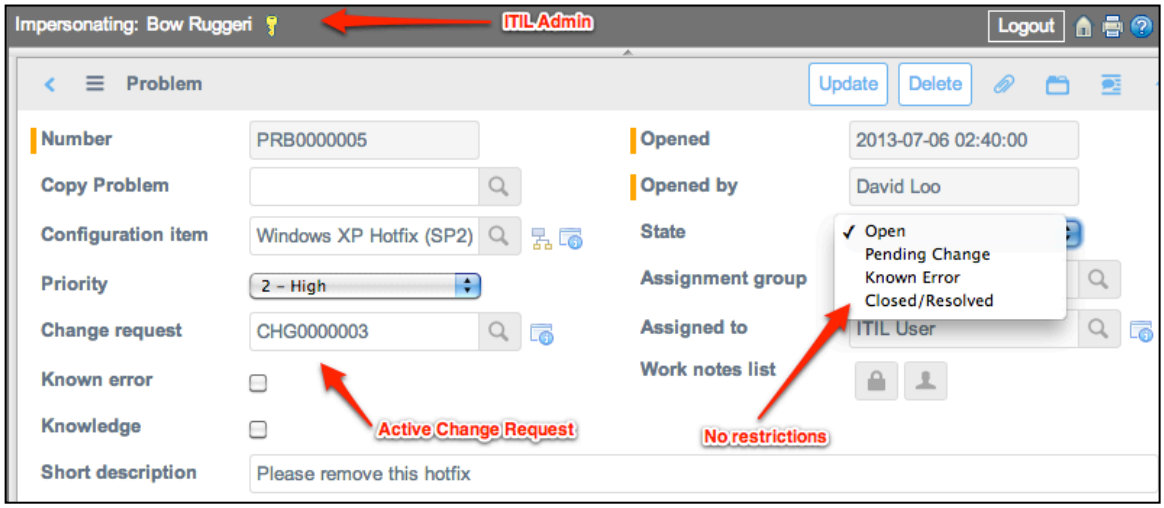
©2014 ServiceNow, Inc. All rights reserved.

Lab Success Verification

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

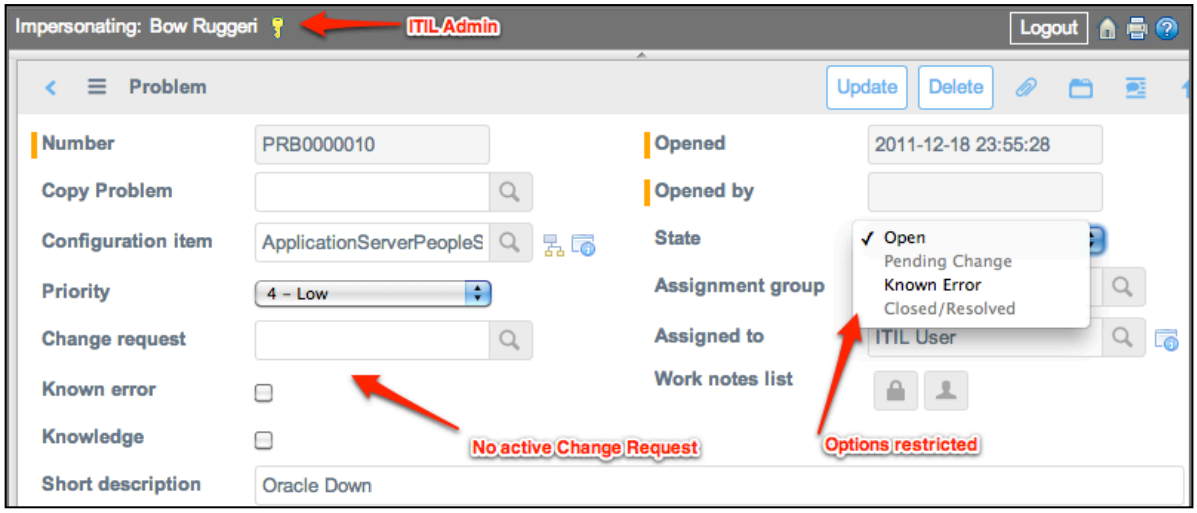


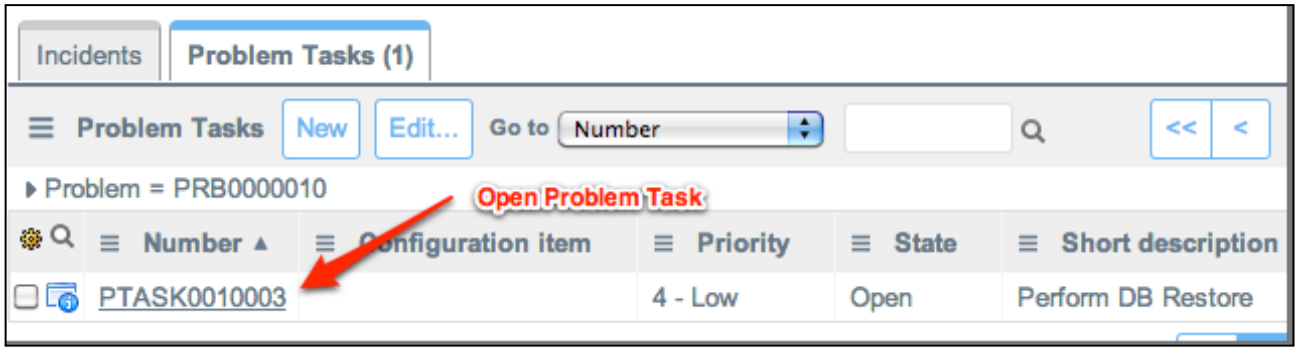
Impersonate **Bow Ruggeri** and double check that he can close the problem, as he has the ITIL Admin role. Since there is an active Change Request associated, the **Pending Change** is an available option.



©2014 ServiceNow, Inc. All rights reserved.

Open problem **PRB0000010** and you will see that Bow Ruggeri can't close the problem, as there is an open Problem Task. Since there is not Change Request associated, the **Pending Change** is not an option.





®2014 ServiceNow, Inc. All rights reserved.

Lab Goal

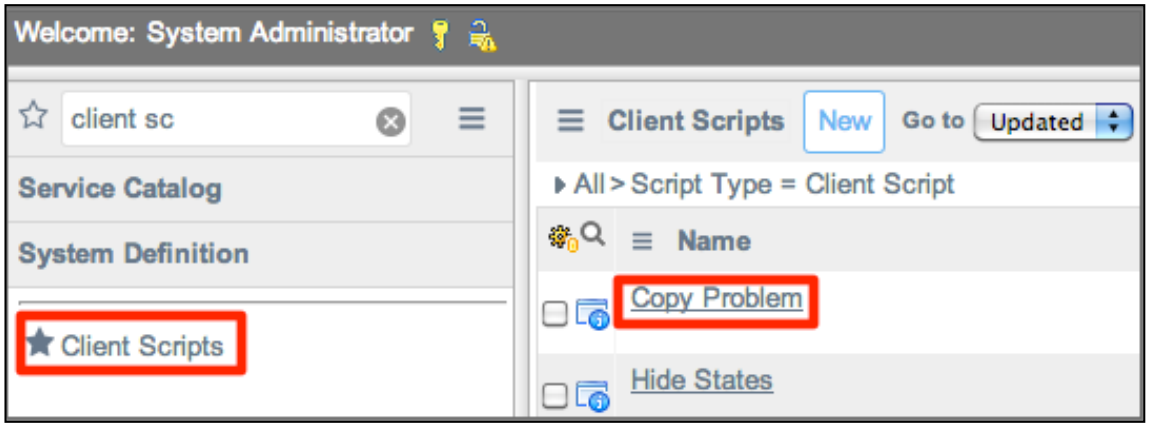
The existing use case of copying values from one record to another with GlideAjax will be extended to copy problem tasks as well. When the callback returns the related list should be reloaded without loading the whole form again.

Lab 4.1
Glide List

Using GlideList

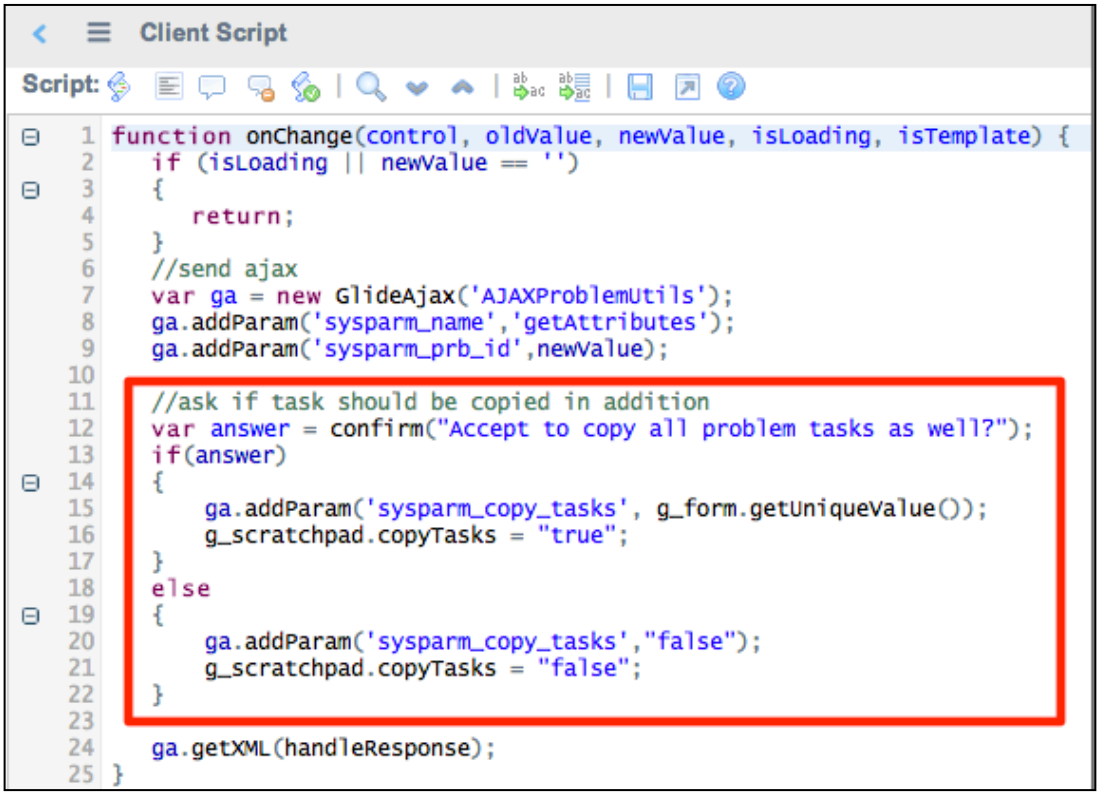
1. Login to the instance URL on the cover page using the provided **admin** credentials.

2. Navigate to **System Definition > Client Script** and open **Copy Problem**.



©2014 ServiceNow, Inc. All rights reserved.

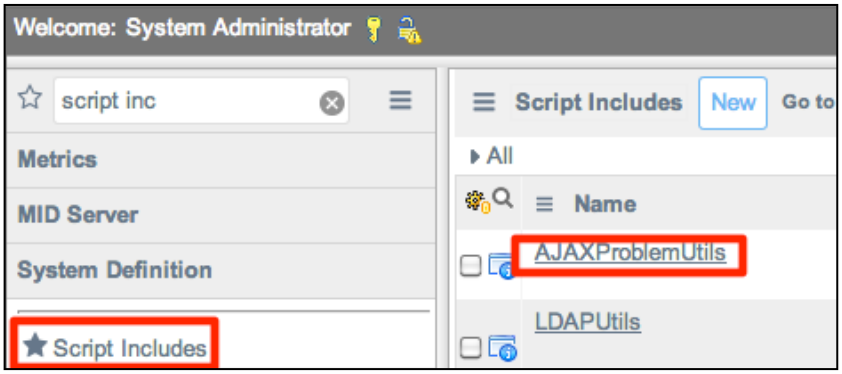
3. Extend the script by this:



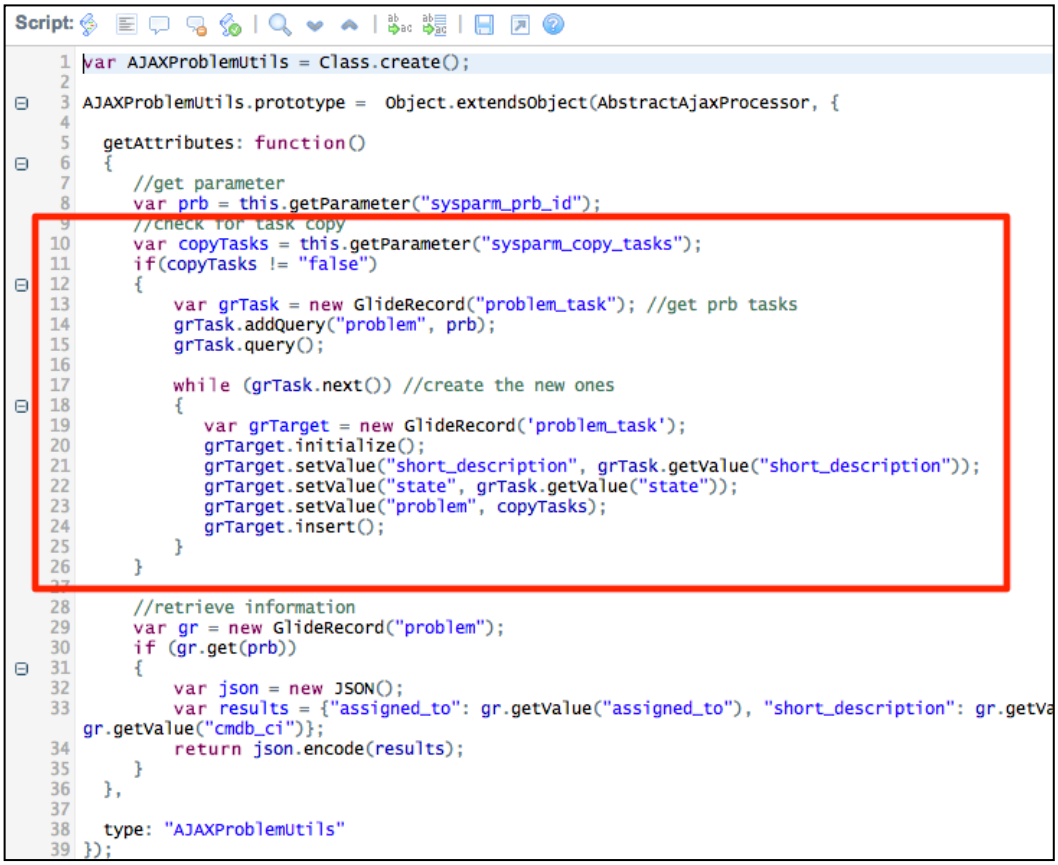
Result: A popup will ask if all related problem tasks of the selected record really should be copied. If the user agrees a parameter will be set in the Ajax call which identifies the record. The scratchpad variable helps to save the users answer in the callback function.

©2014 ServiceNow, Inc. All rights reserved.

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



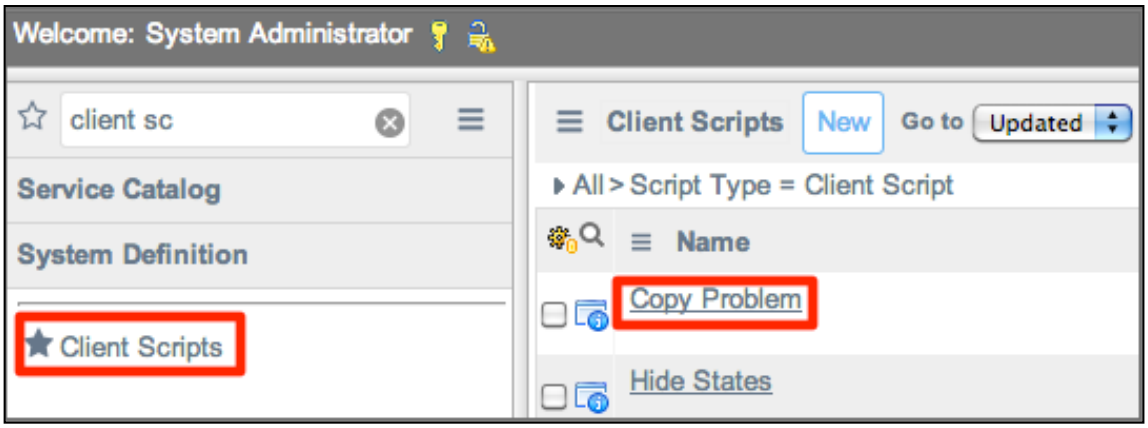
5. Extend the script like this:



Result: If the **copy_task parameter** is set all problem tasks will be copied over.

©2014 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 **Copy Problem**.



7. The callback function have to be extended by this:

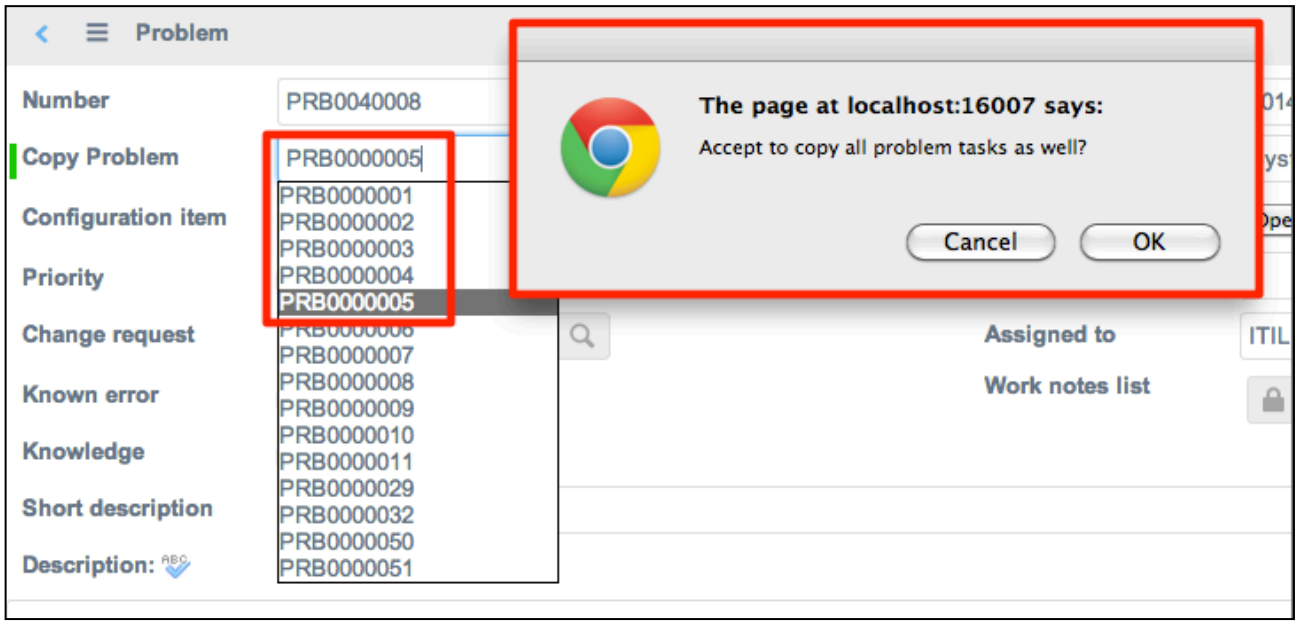
```
function handleResponse(response)
{
    //process return
    var answer = response.responseXML.documentElement.getAttribute("answer");
    if(answer)
    {
        var returneddata = answer.evalJSON(true);
        for(field in returneddata)
        {
            g_form.setValue(field, returneddata[field]);
        }
        if(g_scratchpad.copyTasks == "true") //reload related list
        {
            GlideList2.get('problem.problem_task.problem').refresh(1);
        }
    }
}
```

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

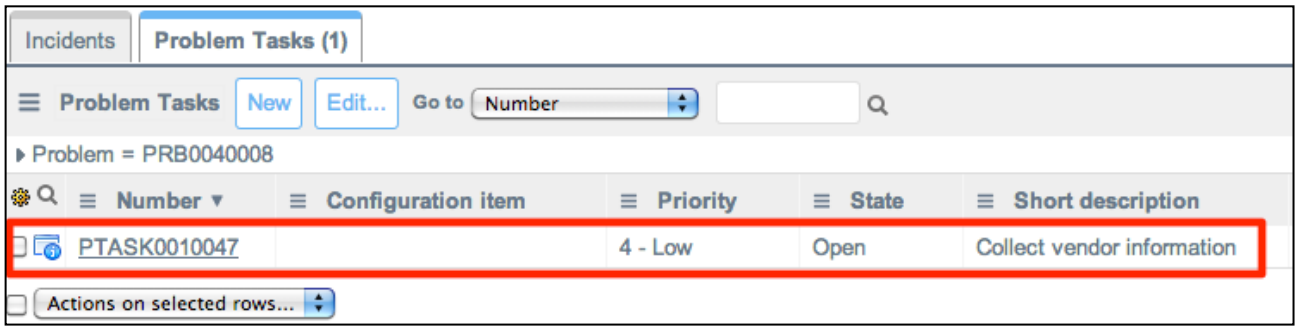
©2014 ServiceNow, Inc. All rights reserved.

Lab Success Verification

Open a problem record and enter the number of an existing problem, which has related problem tasks, in the **Copy Problem** field.



Accept the popup and check the problem tasks related list.



©2014 ServiceNow, Inc. All rights reserved.