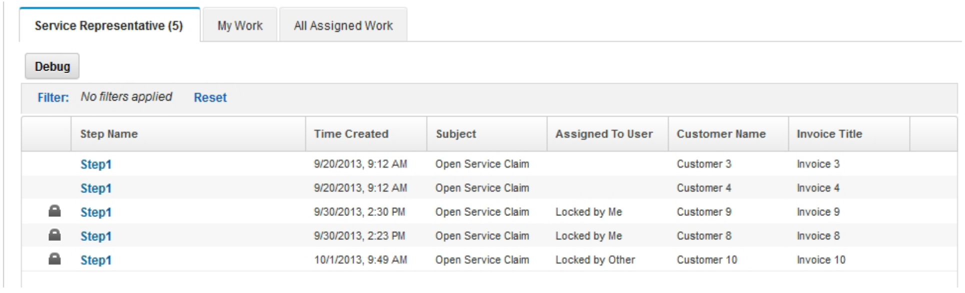
Pre-Requisites

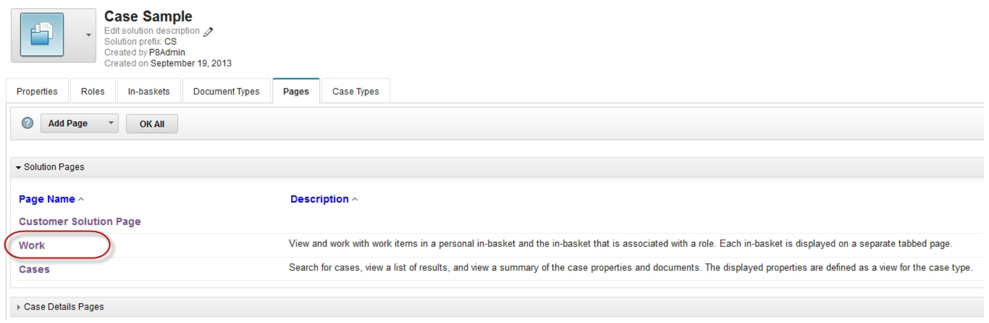
Open Case Builder.
Create a new solution / Edit an existing solution.
Create at least 2 Roles.
Create a Case Type / Task / etc.
Create some test cases / work items.
Lock at least one work item by the current user.
Lock at least one work item by another user.
Leave at least one work item unlocked.
Upon completion your Inbasket should look similar to this…



	Step Name	Time Created	Subject	Assigned To User	Customer Name	Invoice Title
	Step1	9/20/2013, 9:12 AM	Open Service Claim		Customer 3	Invoice 3
	Step1	9/20/2013, 9:12 AM	Open Service Claim		Customer 4	Invoice 4
🔒	Step1	9/30/2013, 2:30 PM	Open Service Claim	Locked by Me	Customer 9	Invoice 9
🔒	Step1	9/30/2013, 2:23 PM	Open Service Claim	Locked by Me	Customer 8	Invoice 8
🔒	Step1	10/1/2013, 9:49 AM	Open Service Claim	Locked by Other	Customer 10	Invoice 10

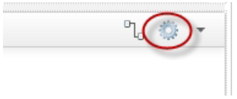
Create the menu event action

Open Case Builder.
Edit your solution.
Click on the "Pages" tab.
Expand the "Solution Pages" group.
Open the "Work" page.

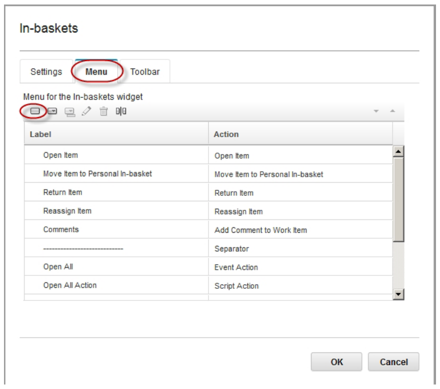


Page Name	Description
Customer Solution Page	
Work	View and work with work items in a personal in-basket and the in-basket that is associated with a role. Each in-basket is displayed on a separate tabbed page.
Cases	Search for cases, view a list of results, and view a summary of the case properties and documents. The displayed properties are defined as a view for the case type.

On the In-baskets widget, click the "Edit Settings" icon.

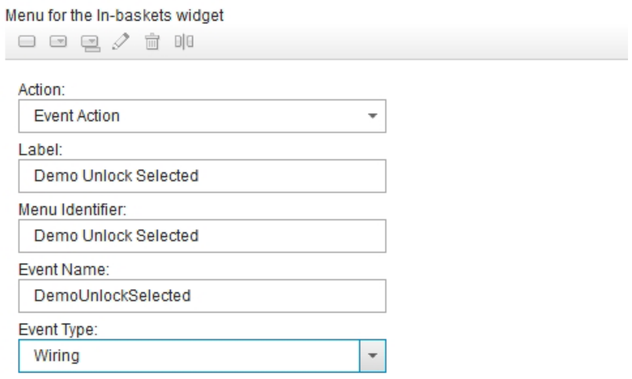


Click the "Menu" tab.
Click the "Add Menu Item" toolbar button.



Label	Action
Open Item	Open Item
Move Item to Personal In-basket	Move Item to Personal In-basket
Return Item	Return Item
Reassign Item	Reassign Item
Comments	Add Comment to Work Item
------------------------------	Separator
Open All	Event Action
Open All Action	Script Action

Create a new menu item similar to the following:



Action: Event Action

Label: Demo Unlock Selected

Menu Identifier: Demo Unlock Selected

Event Name: DemoUnlockSelected

Event Type: Wiring

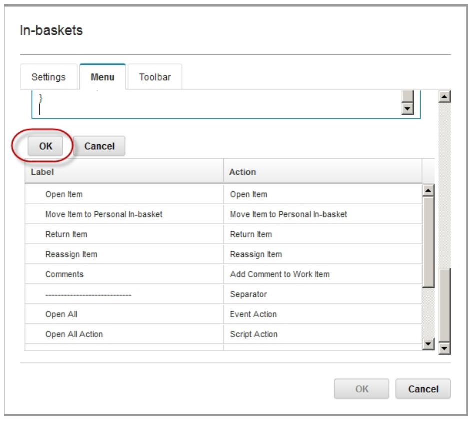
In the “Show this event action:” box, paste the following code:
NOTE: You need to add the name of one of your Roles to line 20 of the script (the line beginning “`var myAuthorisedRoles =`”)
NOTE: The script checks the current users’ role membership against a hardcoded list of authorised Roles in the script. This concept can be extended to query LDAP membership, or to call a bespoke “Authorisation” REST API.

```
try {
    var isAuthorised = false;

    /*userid variables*/
    var myUserid = "";
    var userid = ecm.model.desktop.userId;
    var displayName = ecm.model.desktop.userDisplayName;

    /*REST API variables*/
    var myCP = this.getActionContext("Solution")[0].connectionPoint;
    var myRoles;
    var serverBase;
    var feedURL;
    var deferred;
    var myAppSpace = this.getActionContext("Role")[0].parent.id;
    var xhrArgs;

    /*List of authorised roles*/
    /*YOU NEED TO ADD THE NAME OF ONE OF YOUR SOLUTION ROLES INTO THIS LIST*/
    var myAuthorisedRoles = "|All|Administrator|Exceptions|Service Representative|";

    /*Get current username*/
       if (userid != null)
       {
              myUserid = userid;
       }
       else
       {
              if (displayName != null)
              {
                     myUserid = displayName;
              }
       }

       /*See if the current username was picked up*/
       if (myUserid != "")
       {
        /*Get CASE REST API url*/
        serverBase = window.location.protocol + "\/\/" + window.location.host;
        feedURL = serverBase + "/CaseManager/P8BPMREST/p8/bpm/v1/appspaces/" + myAppSpace +
"/myroles?cp=" + myCP;

        /*Get my Roles*/
        xhrArgs = {
        url: feedURL,
        handleAs: "json",
        sync: true,
        headers: { "Content-Type": "application/json"},
            load: function(data)
            {
               myRoles = data;
            } ,
            error: function(error)
            {
                alert ("Error Encountered..." + error);
            }
        };
        deferred = dojo.xhrGet(xhrArgs);

        /*Iterate through roles and see if authorised*/
        for (myRole in myRoles) {
            if (myAuthorisedRoles.indexOf("|"+myRole+"|") != -1)
            {
                isAuthorised = true;
                break;
            }
        }

        }
    /*Return whether the current user is authorised or not*/
    return isAuthorised;
}
catch (Error) {
        alert ("Source Module: " + this.arguments.label + " Script
Adaptor\r\n\r\n"+Error.name+" - "+Error.description+"\r\n"+Error.message);
    /*An error was thrown. Always return false*/
    return false;
}
```

In the “Enable this event action:” box, paste the following code:
NOTE: The script returns true if all of the selected work items are locked and false otherwise. This is one example of how you might apply “enable” logic using this new feature of ICM 5.2

```
try {
    /*Init return variable*/
    var allLocked = true;

    /*Get the work items that were selected*/
    var selectedWorkItems = this.getActionContext("WorkItemReference");

    /*Variable declarations*/
    var currentWorkItem;
    var i;

    /*Was an array of work items returned*/
    if (dojo.isArray(selectedWorkItems))
    {
        /*Loop through the array*/
        for (i=0; i<selectedWorkItems.length; i++)
        {
            /*Get a reference to the current work item*/
            currentWorkItem = selectedWorkItems[i];
            /*Check whether work item is locked*/
            if (currentWorkItem.lockedUser == "")
            {
                /*If not locked, set variable to false and break out of the loop*/
                allLocked = false;
                break;
            }
        }
    }
    else
    {
        /*Get a reference to the current work item*/
        currentWorkItem = selectedWorkItems;
        /*Check whether work item is locked*/
        if (currentWorkItem.lockedUser == "")
        {
            /*If not locked, set variable to false*/
            allLocked = false;
        }
    }
    /*Return whether all of the selected work items were locked or not*/
    return allLocked;
}
catch (Error) {
    alert ("Source Module: " + this.arguments.label + " Action Adaptor\r\n\r\n"+Error.name+" -
"+Error.description+"\r\n"+Error.message);
    /*An error was thrown. Always return false*/
    return false;
}
```

Click “OK” to add the menu item.


In-baskets

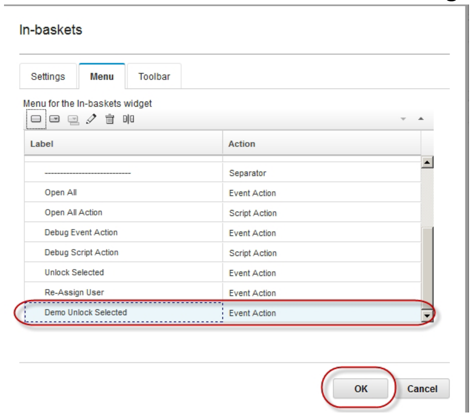
Settings | Menu | Toolbar

OK | Cancel

Label	Action
Open Item	Open Item
Move Item to Personal In-basket	Move Item to Personal In-basket
Return Item	Return Item
Reassign Item	Reassign Item
Comments	Add Comment to Work Item
------------------------------	Separator
Open All	Event Action
Open All Action	Script Action

OK | Cancel

The new menu item will be visible in the “Menu for the In-baskets widget” list.
Click “OK” to close the In-baskets setting dialog.


In-baskets

Settings | Menu | Toolbar

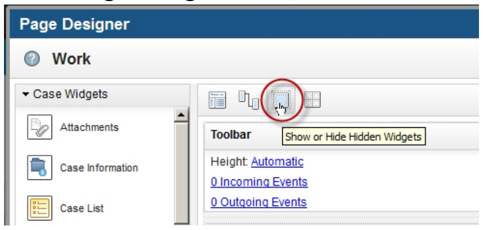
Menu for the In-baskets widget

Label	Action
------------------------------	Separator
Open All	Event Action
Open All Action	Script Action
Debug Event Action	Event Action
Debug Script Action	Script Action
Unlock Selected	Event Action
Re-Assign User	Event Action
Demo Unlock Selected	Event Action

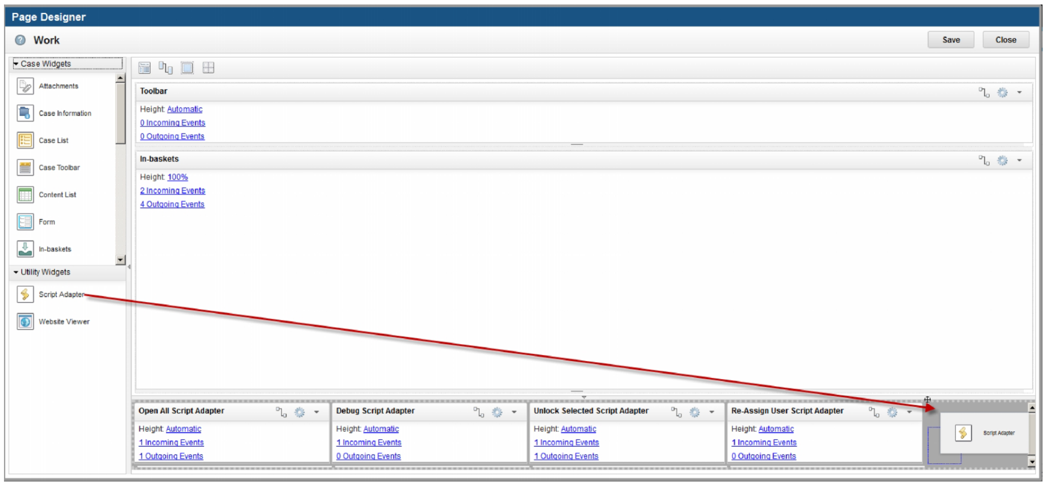
OK | Cancel

Create the Script Adaptor

In the Page Designer area, click the “Show or Hide Hidden Widgets” toolbar button.



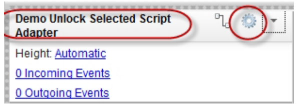
In the Widgets palette, expand the “Utility Widgets” group.
Drag the “Script Adaptor” widget into the Hidden widget area.



Height: Automatic
0 Incoming Events
0 Outgoing Events

Height: 100%
2 Incoming Events
4 Outgoing Events

Rename the Script Adaptor to something descriptive.
Click on the "Edit Settings" button.



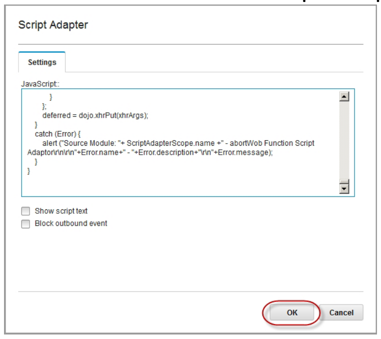
Demo Unlock Selected Script Adapter
Height: Automatic
0 Incoming Events
0 Outgoing Events

In the "Javascript:" box, paste the following code:
NOTE: The BPM REST API is used in this script and not the new model API. There are a couple of goods reasons for this:

- The model API does not currently provide a method to unlock a work item that has been locked by another user.
- The Inbaskets widget does not respond to a broadcast "icm.Refresh" event. If you want to refresh the Inbasket after the script has completed, then using the dojo.xhrGet and dojo.xhrPut commands with the "sync: true" setting allows you to synchronously unlock the work items. This enables you to wire an outbound event from the Script Adaptor to the Inbaskets → Refresh event.

```
try {

    var ScriptAdapterScope = this;

    /*Get the work items that were selected*/
    var selectedWorkItems = payload.WorkItemReference;

    /*Variable declarations*/
    var currentWorkItem;
    var i;

    /*userid variables*/
    var myUserid = "";
    var userid = ecm.model.desktop.userId;
    var displayName = ecm.model.desktop.userDisplayName;

    /*REST API variables*/
    var myCP = payload.Solution.connectionPoint;
    var xhrArgs;
    var myPayload;
    var myETag;
    var deferred;
    var serverBase;
    var feedURL;

    /*Get current username*/
    if (userid != null) {
        myUserid = userid;
    }
    else {
        if (displayName != null) {
            myUserid = displayName;
        }
    }
    /*Was an array of work items returned*/
    if (dojo.isArray(selectedWorkItems)) {
        /*Loop through the array*/
        for (i = 0; i < selectedWorkItems.length; i++) {
            /*Get a reference to the current work item*/
            currentWorkItem = selectedWorkItems[i];

            /*Check whether work item is locked*/
            if (currentWorkItem.lockedUser != "") {
                /*Check whether work item is not locked by the current user*/
                if (currentWorkItem.lockedUser.toUpperCase() != myUserid.toUpperCase()) {
                    /*Locked by someone else, so need to override the lock*/
                    /*Call the overrideLock function*/
                    overrideLock(currentWorkItem.queueName, currentWorkItem.id);
                }
                /*Abort the current action on the wob*/
                /*Call the abortWob function*/
                abortWob(currentWorkItem.queueName, currentWorkItem.id);
            }
        }
    }
    else {
        /*Get a reference to the current work item*/
        currentWorkItem = selectedWorkItems;

        /*Check whether work item is locked*/
        if (currentWorkItem.lockedUser != "") {
            /*Check whether work item is not locked by the current user*/
            if (currentWorkItem.lockedUser.toUpperCase() != myUserid.toUpperCase()) {
                /*Locked by someone else, so need to override the lock*/
                /*Call the overrideLock function*/
                overrideLock(currentWorkItem.queueName, currentWorkItem.id);
            }
            /*Abort the current action on the wob*/
            /*Call the abortWob function*/
            abortWob(currentWorkItem.queueName, currentWorkItem.id);
        }
    }
}
catch (Error) {
    alert("Source Module: " + this.name + " Script Adaptor\r\n\r\n" + Error.name + " - " +
Error.description + "\r\n" + Error.message);
}

function overrideLock(currentQueueName, currentWobId) {
    try {
        /*Get CASE REST API url*/
        serverBase = window.location.protocol + "\/\/" + window.location.host;
        feedURL = serverBase + "/CaseManager/P8BPMREST/p8/bpm/v1/queues/" + currentQueueName +
"/stepelements/" + currentWobId + "?cp=" + myCP;

        /*Get Step Element*/
        xhrArgs = {
            url: feedURL,
            handleAs: "json",
            sync: true,
            headers: { "Content-Type": "application/json" },
            load: function (data) {
                myPayload = data;
            },
            error: function (error) {
                alert("Error Encountered..." + error);
            }
        };
        deferred = dojo.xhrGet(xhrArgs);
        /*Get ETag from request*/
        myETag = deferred.ioArgs.xhr.getResponseHeader("ETag");

        /*Need to override the existing lock*/
        /*Get CASE REST API url*/
        feedURL = serverBase + "/CaseManager/P8BPMREST/p8/bpm/v1/queues/" + currentQueueName +
"/stepelements/" + currentWobId + "?cp=" + myCP + "&action=overrideLock";

        /*You must pass in the ETag from the previous REST call into an "If-Match" header
tag*/
        xhrArgs = {
            url: feedURL,
            handleAs: "json",
            sync: true,
            headers: { "Content-Type": "application/json", "If-Match": myETag },
            load: function (data) {
                myPayload = data;
            },
            error: function (error) {
                alert("Error Encountered..." + error);
            }
        };
        deferred = dojo.xhrPut(xhrArgs);

    }
    catch (Error) {
        alert("Source Module: " + ScriptAdapterScope.name + " - overrideLock Function Script 
Adaptor\r\n\r\n" + Error.name + " - " + Error.description + "\r\n" + Error.message);
    }
}


function abortWob(currentQueueName, currentWobId) {
    try {
        /*Get CASE REST API url*/
        serverBase = window.location.protocol + "\/\/" + window.location.host;
        feedURL = serverBase + "/CaseManager/P8BPMREST/p8/bpm/v1/queues/" + currentQueueName + 
"/stepelements/" + currentWobId + "?cp=" + myCP;

        /*Get Step Element*/
        xhrArgs = {
            url: feedURL,
            handleAs: "json",
            sync: true,
            headers: { "Content-Type": "application/json" },
            load: function (data) {
                myPayload = data;
            },
            error: function (error) {
                alert("Error Encountered..." + error);
            }
        };
        deferred = dojo.xhrGet(xhrArgs);
        /*Get ETag from request*/
        myETag = deferred.ioArgs.xhr.getResponseHeader("ETag");

        /*Unlock the WOB by issuing abort action*/
        /*Get CASE REST API url*/
        serverBase = window.location.protocol + "\/\/" + window.location.host;
        feedURL = serverBase + "/CaseManager/P8BPMREST/p8/bpm/v1/queues/" + currentQueueName + 
"/stepelements/" + currentWobId + "?cp=" + myCP + "&action=abort";

        /*You must pass in the ETag from the previous REST call into an "If-Match" header 
tag*/
        xhrArgs = {
            url: feedURL,
            handleAs: "json",
            sync: true,
            headers: { "Content-Type": "application/json", "If-Match": myETag },
            load: function (data) {
                myPayload = data;
            },
            error: function (error) {
                alert("Error Encountered..." + error);
            }
        };
        deferred = dojo.xhrPut(xhrArgs);
    }
    catch (Error) {
        alert("Source Module: " + ScriptAdapterScope.name + " - abortWob Function Script 
Adaptor\r\n\r\n" + Error.name + " - " + Error.description + "\r\n" + Error.message);
    }
}
```

Click “OK” to close the Script Adaptor settings dialog

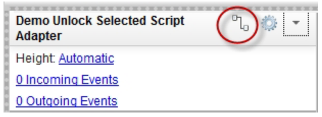


JavaScript::

```
            }
        };
        deferred = dojo.xhrPut(xhrArgs);
    }
    catch (Error) {
        alert ("Source Module: "+ ScriptAdapterScope.name +" - abortWob Function Script 
Adaptor\r\n\r\n"+Error.name+" - "+Error.description+"\r\n"+Error.message);
    }
}
```

☐ Show script text
☐ Block outbound event

In the Script Adaptor, click on the "Edit Wiring" button

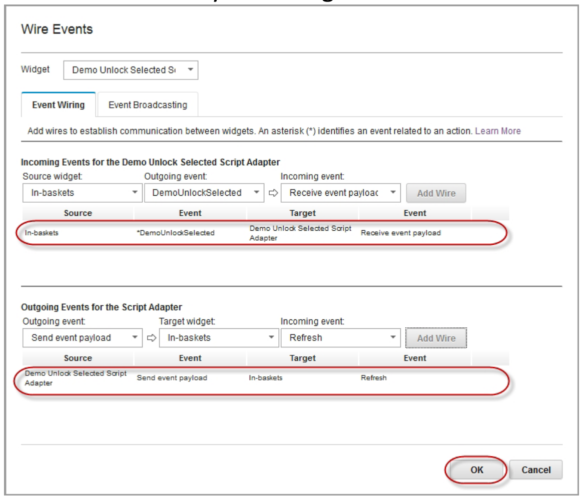

Demo Unlock Selected Script Adapter

Height: Automatic

0 Incoming Events

0 Outgoing Events

Wire the "DemoUnlockedSelected" Inbaskets event as an Inbound event into your Script Adaptor
Wire the Inbaskets "Refresh" event as an Outbound event from your Script Adaptor.
Click "OK" to save your changes.


Wire Events

Widget: Demo Unlock Selected S

Event Wiring | Event Broadcasting

Add wires to establish communication between widgets. An asterisk (*) identifies an event related to an action. Learn More

Incoming Events for the Demo Unlock Selected Script Adapter

Source widget: In-baskets — Outgoing event: DemoUnlockSelected — Incoming event: Receive event payload — Add Wire

Source	Event	Target	Event
In-baskets	*DemoUnlockSelected	Demo Unlock Selected Script Adapter	Receive event payload

Outgoing Events for the Script Adapter

Outgoing event: Send event payload — Target widget: In-baskets — Incoming event: Refresh — Add Wire

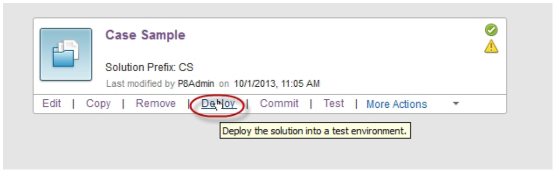
Source	Event	Target	Event
Demo Unlock Selected Script Adapter	Send event payload	In-baskets	Refresh

OK | Cancel

In the Page Designer, Click "Save" and then "Close".

In your Solution click "Save and Close".

In the Case Builder home page, "Deploy" your solution.


Case Sample

Solution Prefix: CS

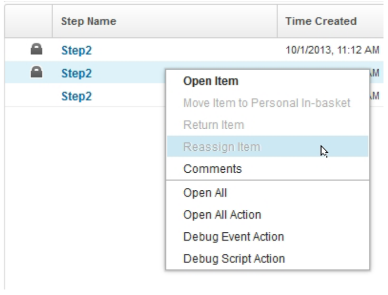
Last modified by P8Admin on 10/1/2013, 11:05 AM

Edit | Copy | Remove | Deploy | Commit | Test | More Actions

Deploy the solution into a test environment.

Test the Script

Assign your username only to a Role that is not in the authorised roles list. The "Demo Unlock Selected" menu item should not be visible.



Step Name	Time Created
Step2	10/1/2013, 11:12 AM
Step2	
Step2	