UFT Evaluation

Contents

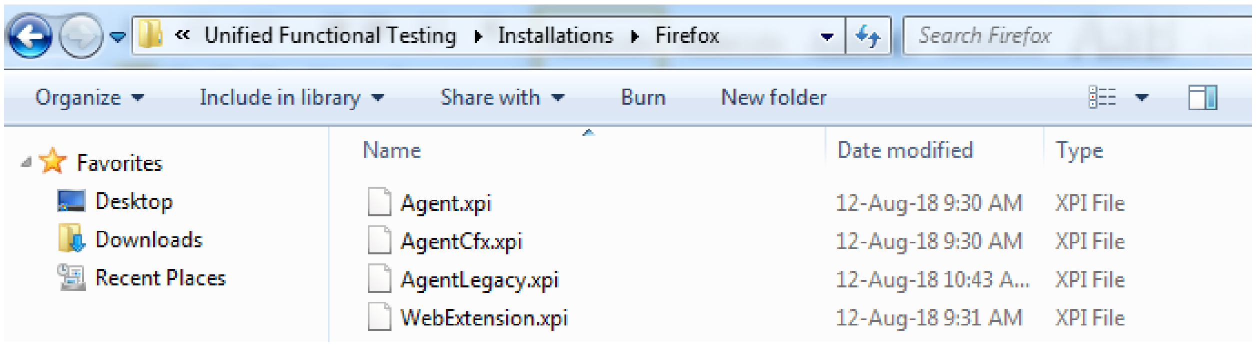
Environment

- DiagBox version: 09.34
- UFT version: 14.50 trial version
- Mozilla Firefox version: 52.2.0 portable (as provided by DiagBox installation)
- OS platform: Microsoft Windows 7

UFT plugin installation

For UFT framework to be able to pilot the DiagBox application interface, a plugin must be installed in order to permit the automation of actions.

Because DiagBox is built using the Mozilla Firefox browser, the appropriate plugin must be installed. This plugin can be found in the installation folder of the UFT framework (in our case *C:\Program Files (x86)\Micro Focus\Unified Functional Testing\Installations\Firefox*):

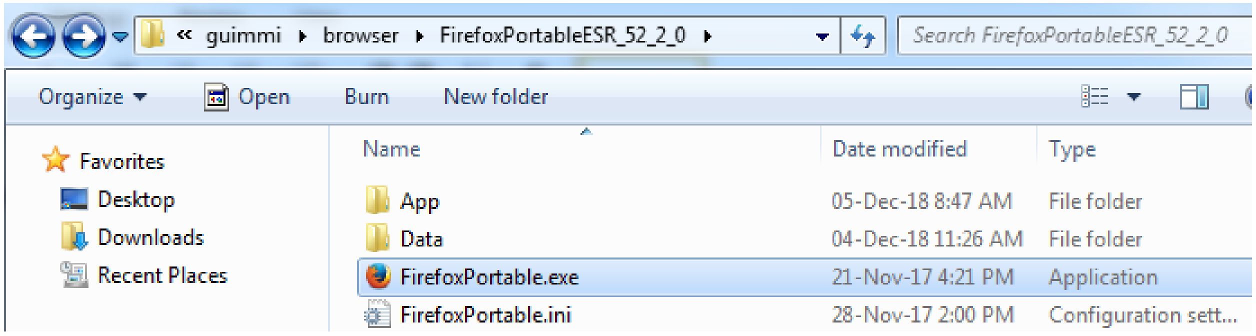


Name	Date modified	Type
Agent.xpi	12-Aug-18 9:30 AM	XPI File
AgentCfx.xpi	12-Aug-18 9:30 AM	XPI File
AgentLegacy.xpi	12-Aug-18 10:43 A...	XPI File
WebExtension.xpi	12-Aug-18 9:31 AM	XPI File

In this installation folder, multiple add-on can be found. The one needed for our case is the first xpi file called *Agent.xpi*.

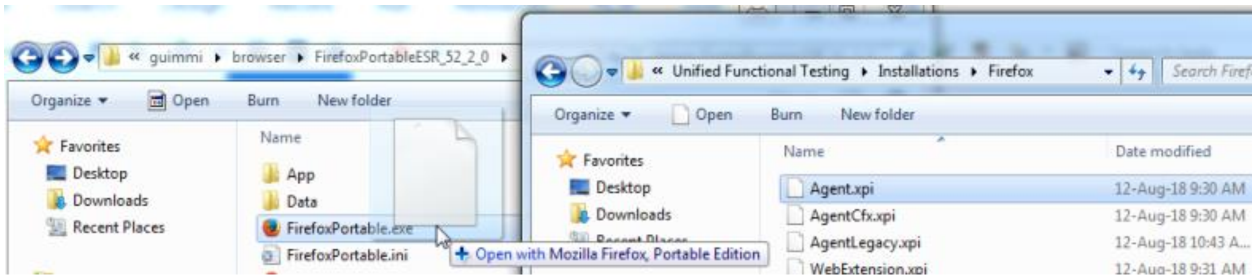
Installation procedure

1. Locate the *Agent.xpi* file in your UFT installation path (as shown before)
2. In the *C:\AWRoot\bin\guimmi\browser\FirefoxPortableESR_52_2_0* path, locate the Firefox executable:


« guimmi ▸ browser ▸ FirefoxPortableESR_52_2_0 ▸
Search FirefoxPortableESR_52_2_0
Organize
Open
Burn
New folder
Favorites
Desktop
Downloads
Recent Places

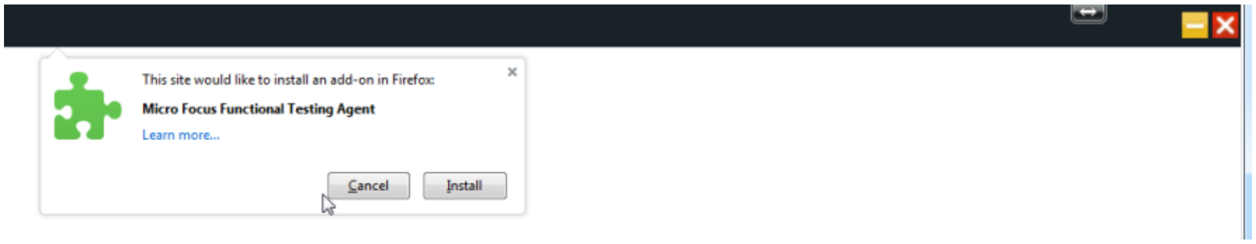
Name	Date modified	Type
App	05-Dec-18 8:47 AM	File folder
Data	04-Dec-18 11:26 AM	File folder
FirefoxPortable.exe	21-Nov-17 4:21 PM	Application
FirefoxPortable.ini	28-Nov-17 2:00 PM	Configuration sett...

3. Using the drag-and-drop method, drag the *Agent.xpi* file over the *FirefoxPortable.exe*:


« guimmi ▸ browser ▸ FirefoxPortableESR_52_2_0 ▸
Organize ▾ Open Burn New folder
Favorites
Desktop
Downloads
Recent Places
Name
App
Data
FirefoxPortable.exe
FirefoxPortable.ini
+ Open with Mozilla Firefox, Portable Edition
« Unified Functional Testing ▸ Installations ▸ Firefox
Search Firef
Organize ▾ Open Burn New folder
Favorites
Desktop
Downloads
Name
Date modified

Name	Date modified
Agent.xpi	12-Aug-18 9:30 AM
AgentCfx.xpi	12-Aug-18 9:30 AM
AgentLegacy.xpi	12-Aug-18 10:43 A...
WebExtension.xpi	12-Aug-18 9:31 AM

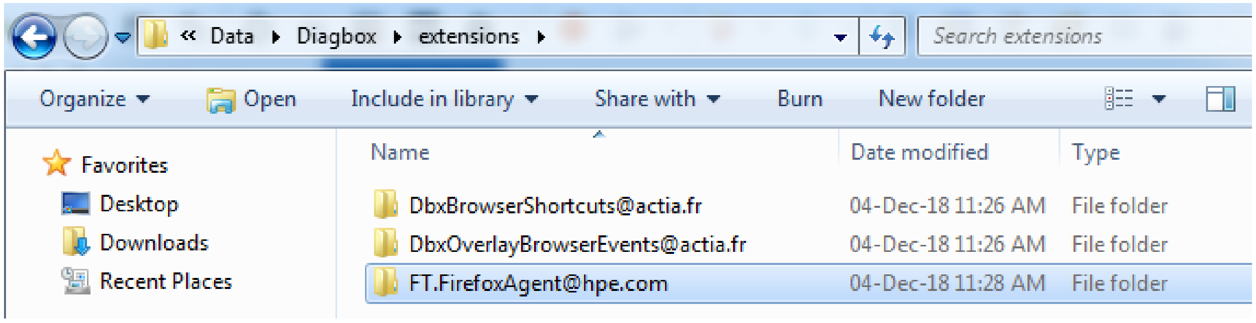
4. The Mozilla Firefox will start and demand the add-on installation – click on the install button:



This site would like to install an add-on in Firefox:

Micro Focus Functional Testing Agent

5. After the installation, in the default Firefox profile named *Diagbox*, the add-on should be present (path: *C:\AWRoot\bin\guimmi\browser\FirefoxPortableESR_52_2_0\Data\Diagbox\extensions*):

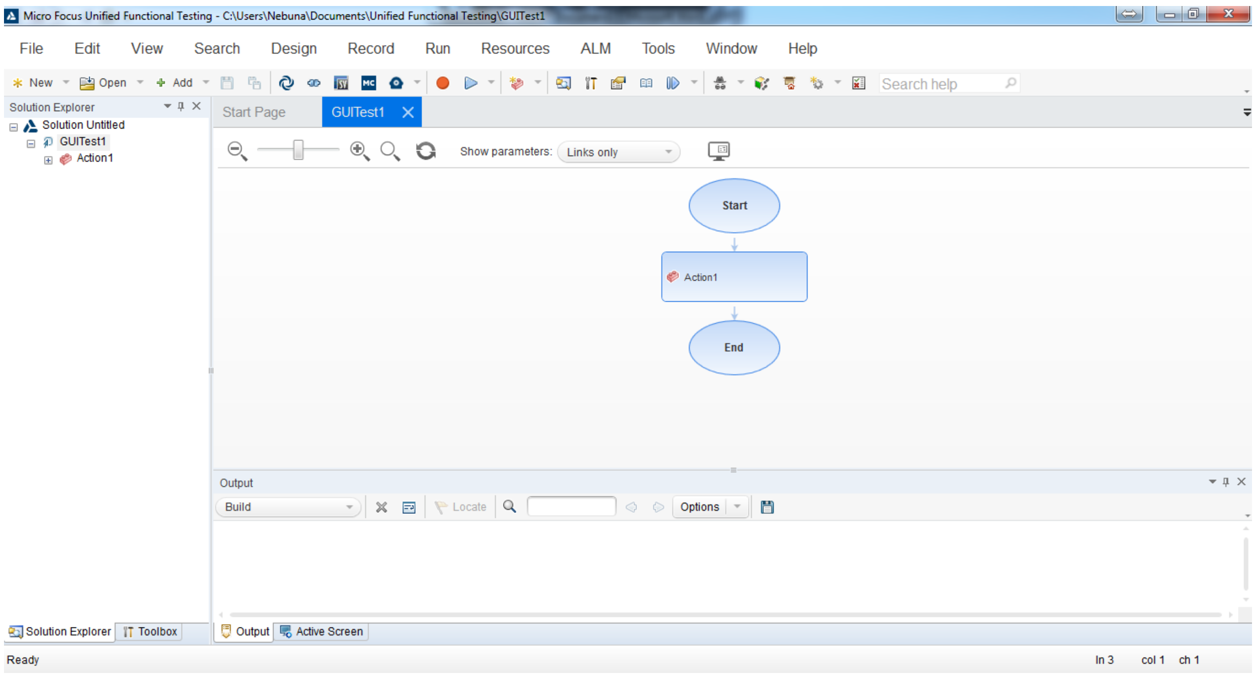

« Data ▸ Diagbox ▸ extensions ▸
Search extensions
Organize ▾ Open Include in library ▾ Share with ▾ Burn New folder
Favorites
Desktop
Downloads
Recent Places

Name	Date modified	Type
DbxBrowserShortcuts@actia.fr	04-Dec-18 11:26 AM	File folder
DbxOverlayBrowserEvents@actia.fr	04-Dec-18 11:26 AM	File folder
FT.FirefoxAgent@hpe.com	04-Dec-18 11:28 AM	File folder

UFT utilization

After the add-on has been installed, UFT is able to recognize the DiagBox graphical user interface elements. In order to do this, UFT must be started with administrator rights (right-click on the icon and select “Run as administrator” option).

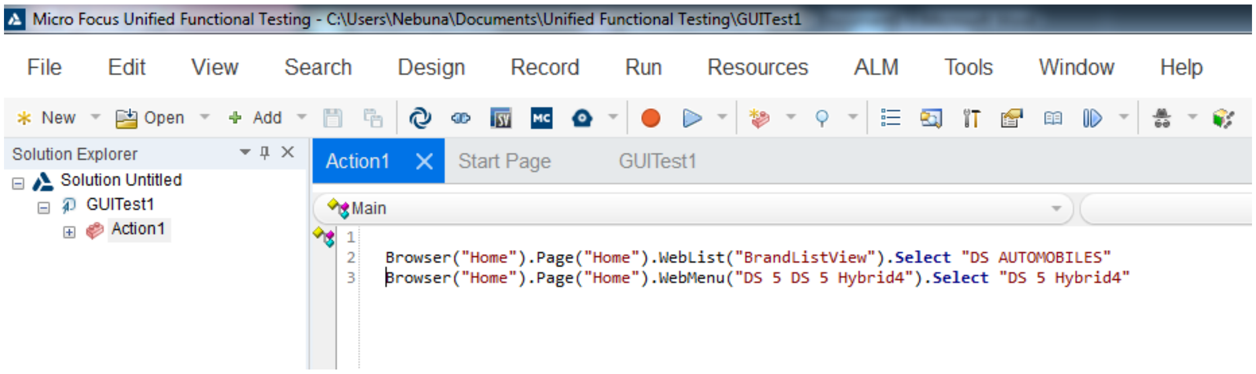
At the welcome screen, check that the web checkbox is checked (as DIagBox is seen as a web application) and create a new project:



The test can be seen as a schematic, where all the action that can and must be done have to be placed between the **Start** and **End** phases. In this case, there is an action named **Action1** that can store some instruction regarding the test to be performed. Using the record and spy functions present in the framework, several actions can be automatized.

Button click

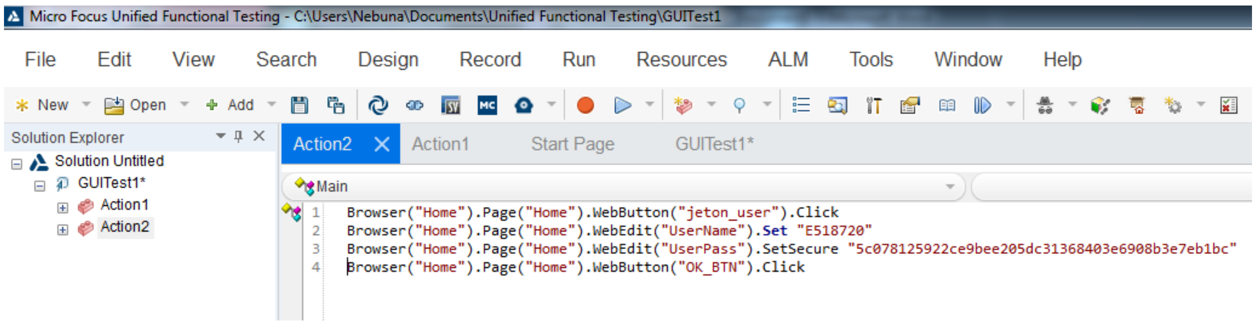
Having DiagBox previously launched, we can record simple actions as clicking on GUI buttons. The code generated below represents the selection of a car manufacturer and of a vehicle:



```
Browser("Home").Page("Home").WebList("BrandListView").Select "DS AUTOMOBILES"
Browser("Home").Page("Home").WebMenu("DS 5 DS 5 Hybrid4").Select "DS 5 Hybrid4"
```

Data introduction

Using the same record function, we can see how the user can automatize actions where some data input must be done:

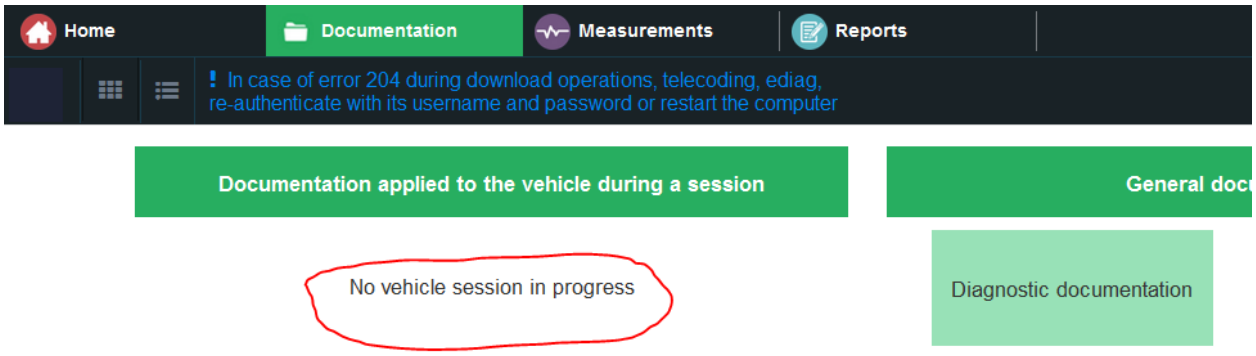


```
Browser("Home").Page("Home").WebButton("jeton_user").Click
Browser("Home").Page("Home").WebEdit("UserName").Set "E518720"
Browser("Home").Page("Home").WebEdit("UserPass").SetSecure "5c078125922ce9bee205dc31368403e6908b3e7eb1bc"
Browser("Home").Page("Home").WebButton("OK_BTN").Click
```

In the picture before, the execution of these functions represents the DiagBox action when a user must authenticate himself in order to have access to other functionalities.

Text recognition

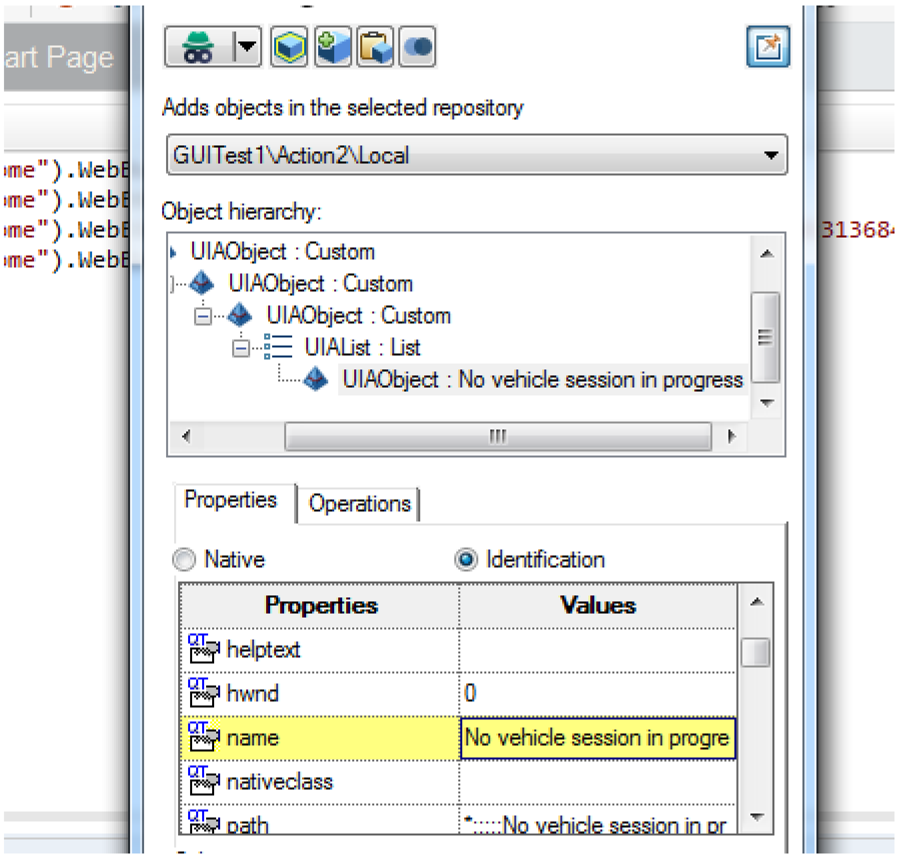
In some cases, it is necessary to be able to extract some data that is shown on the screen and check it against a predetermined value (e.g. check if the value of a parameter is correct or not). In this case, the text recognition function already embedded in UFT is usefull. In the given screen, the highlighted screen area is only a text area, without other functions:



! In case of error 204 during download operations, telecoding, ediag, re-authenticate with its username and password or restart the computer

No vehicle session in progress

This area is found by UFT using the spy function. After the inspection, this area will be mapped in the repository, and several attributes will be saved.


Adds objects in the selected repository

GUITest1\Action2\Local

Object hierarchy:

```
UIAObject : Custom
  UIAObject : Custom
    UIAObject : Custom
      UIAList : List
        UIAObject : No vehicle session in progress
```

Properties | Operations

Native [✓ Identification]

Properties	Values
helptext	
hwnd	0
name	No vehicle session in progre
nativeclass	
path	*:::::No vehicle session in pr

Using the name attribute, we can check the value of it and compare it with a predetermined value.

Known issues and limitations

1. If you are working on a computer where the UAC (User Account Control) option is set to ON, UFT does not support testing on Mozilla Firefox browsers that were installed (or upgraded to a new version) after you installed UFT.

Workaround: After installing Mozilla Firefox on the environment described above, log in as an administrator and open UFT. This enables UFT to install files that are required for Mozilla Firefox support.

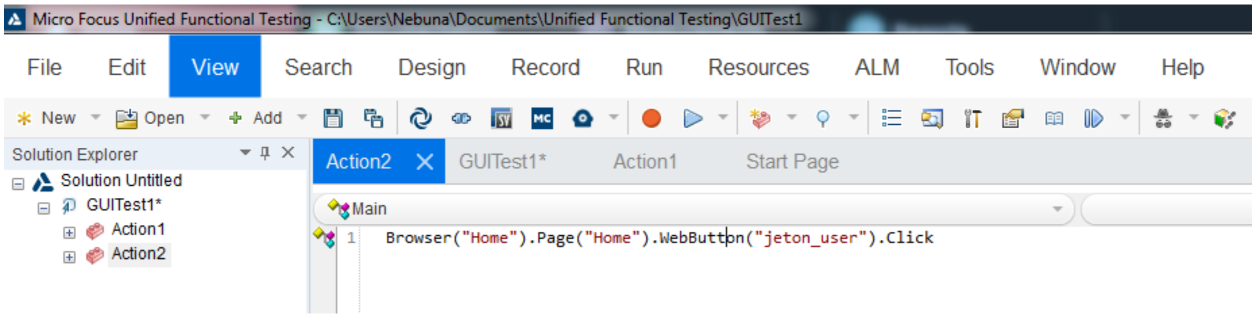
2. UFT does not support accessing browser dialog boxes (such as alert, confirmation, or prompt) directly in Mozilla Firefox.

Workaround: Use the HandleDialog() or GetDialogText() methods.

3. Objects in a BootStrap library that use the **<a>** element with both the *btn* and *disabled* classes cannot be recognized by UFT. These classes disable pointer-events, which are required for UFT object recognition.
4. Elements that aren’t in the object repository can’t be accessed, and, therefore, automatized. Even if it is known how can a specific object be accessed, the user will receive a value if he will try to access that object

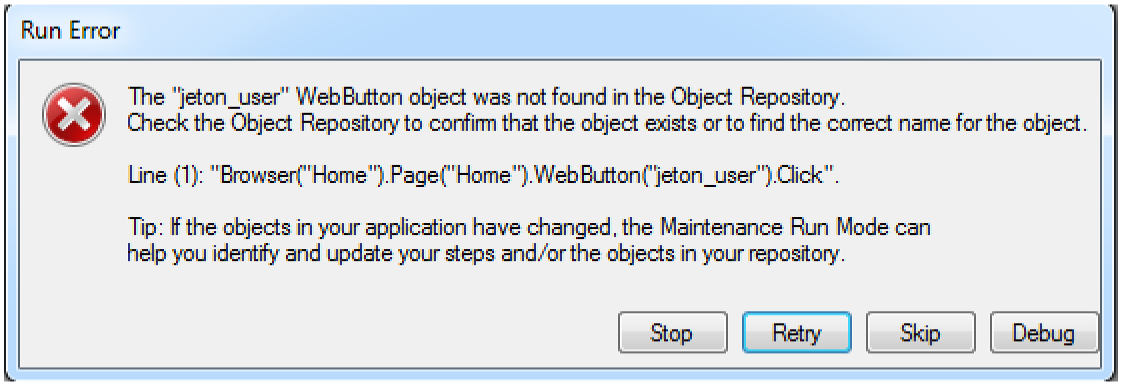
Example:

In the following program sequence will launch the user-authentication window:


Micro Focus Unified Functional Testing - C:\Users\Nebuna\Documents\Unified Functional Testing\GUITest1
File Edit View Search Design Record Run Resources ALM Tools Window Help
New Open Add
Solution Explorer
Solution Untitled
GUITest1*
Action1
Action2
Action2 GUITest1* Action1 Start Page
Main

```
Browser("Home").Page("Home").WebButton("jeton_user").Click
```

After this step, the button *jeton_user* will be mapped in the object repository and will be accessible to the project scope. However, if the same code line will be used in another UFT project where this specific button will not be present in the repository, at the run-time, the following error message will be displayed:



```
The "jeton_user" WebButton object was not found in the Object Repository.
Check the Object Repository to confirm that the object exists or to find the correct name for the object.

Line (1): "Browser("Home").Page("Home").WebButton("jeton_user").Click".

Tip: If the objects in your application have changed, the Maintenance Run Mode can
help you identify and update your steps and/or the objects in your repository.
```

Workaround 1: at the beginning of the project, all GUI elements will be mapped into the object repository and this repository will be shared by all the future projects.

Problem for workaround 1: if there will be a tool evolution in the future that will introduce new graphical elements that will be needed in the tests, the object repository will have to be updated each time. The end-user will have to learn the details of UFT framework.

Workaround 2: there is a possibility to access the elements that are not present in the object repository by using the descriptive programming method. Basically, the content of each page will be broken into pieces and each element will have to be checked. Depending on the attributes of each element, the element will be identified and accessed.

Problem for workaround 2: the conception time will be longer because for each element various checks will be performed. Also, there is the risk that the elements cannot be properly identified based on their attributes (several elements can have the same attribute values). However, this solution will require just basic knowledge from the end-user part.

Conclusion

Even if the totality of screens and actions have not been examined in this period, based on the tests executed so far and on type of actions that are needed to be automatized during DiagBox tests, we can assume that UFT can be successfully used in the automatization toolchain. However, taking into account the particularities of the tested application (especially the fact that there will be an continuous Diagbox evolution) and the necessary workaround (the use of descriptive programming method), for the conception phase there will have to be allocated more time.