|  |
| --- |
| UFT: User Guide |

Contents

[General Information 3](#_Toc3219205)

[Function Description 3](#_Toc3219206)

[Login UFT() 3](#_Toc3219207)

[LaunchDiagBox() 4](#_Toc3219208)

[Authentification(Action, User, Password, Brand) 4](#_Toc3219209)

[BrandSelect(Brand) 5](#_Toc3219210)

[ModelSelect(DetectionType) 5](#_Toc3219211)

[LaunchApplication(Operation) 6](#_Toc3219212)

[SelectTab(Name) 7](#_Toc3219213)

[SelectECU(Family, SubFamily) 8](#_Toc3219214)

[SelectMenu(Name) 8](#_Toc3219215)

[SelectSideMenu(Name) 8](#_Toc3219216)

[CaptureScreen(Report) 8](#_Toc3219217)

[SeeTG () 9](#_Toc3219218)

[Impression () 9](#_Toc3219219)

[TestIdent(ParamName, Format, DataType) () 9](#_Toc3219220)

[TestDTC() 10](#_Toc3219221)

[EFFDTC () 10](#_Toc3219222)

[TestMP(ParamName, DataType, Format, Unit, Help) 10](#_Toc3219223)

[Delay(Time) 11](#_Toc3219224)

# General Information

HPE Unified Functional Testing is automated testing software designed for testing various software applications and environments. It performs functional and regression testing through a user interface such as a native GUI or web interface It works by identifying the objects in the application user interface or a web page and performing desired operations (such as mouse clicks or keyboard events); it can also capture object properties like name or handler ID. HPE Unified Functional Testing uses a VBScript scripting language to specify the test procedure and to manipulate the objects and controls of the application under test. To perform more sophisticated actions, users may need to manipulate the underlying VBScript.

# Function Description

The functions that are developed in UFT for automating various actions are written using Visual Basic Script language, and works properly only if the project has the correct repository file. In the case of a different repository is used, other than the standard one, it is possible that the functions will not work properly and the user might receive some run-time errors.

Each function call can require none, one or several run-time parameters. In the case when the function requires more than one parameter, the order of the parameters are important. If the order is not respected, then the script result will not be valid. Also, all parameters must be defined, as VBScript language does not allow the definition of optional parameters.

Note that in the library is set the path of the folder, which must contain the “Log title”, the folder with all the screenshots and all the three archives (APP\_log.zip, AWRoot\_trace, AWRoot\_log). If it cannot be set the path of the folder because already exist a folder with the same name, then the found folder will be renamed and will be added our folder.

The name of this folder will be “Report” and will be located in "C:\UFT\Reports\” path. In this folder there will be all the reports for a different test. Because there are many tests, “Report” folder must be differentiated. Therefore, the folder name will be composed of “Report” and the time at which it is created.

Example:

**Set** objFSO = **CreateObject**("Scripting.FileSystemObject")  
**Set** fso = **CreateObject**("Scripting.FileSystemObject")  
  
sTimeStamp = **Replace**(**Replace**(**Replace**(**Now**, " ", "\_"), "/", ""), ":", "")  
path = "C:\UFT\Reports\Report" + sTimeStamp  
**Err**.**clear**  
**On** **error** **resume** **next**  
    objFSO.CreateFolder(path)      
**If** **err** **Then**  
    objFSO.MoveFolder path, path + RandomNumber(1000,9999)  
     objFSO.CreateFolder(path)  
     objFSO.CreateFolder(path + "\images\")  
     **Set** f = fso.OpenTextFile(path + "\LogFile.txt", 2, **True**)  
**Else**  
    objFSO.CreateFolder(path)   
    objFSO.CreateFolder(path + "\images\")  
    **Set** f = fso.OpenTextFile(path + "\LogFile.txt", 2, **True**)  
**End** **If**  
path\_images = path + "\images\"

## Import an Excel Sheet(a single file) to DataTable(UFT)

A DataTable is similar with Excel. There are two types of Datatables:

- Action/Local DataTable - Each action has its own private datatable, also known as local datatable, which can be accessed across actions.

 - Global DataTable - Each test has one global data sheet that is accessible across actions.

Very important! The Excel file must be closed when the user run the application.

The syntax to import an Excel sheet is:

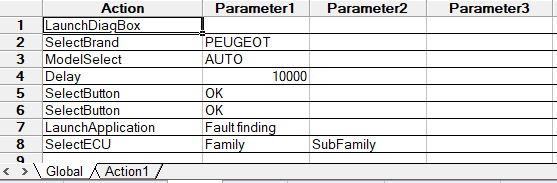
DataTable.ImportSheet Path\_FileName, SheetName, DataTable\_Sheet

Ex: DataTable.ImportSheet "C:\UFT\TestImportExcelSheet.xlsx", "Sheet1", "Action1"

'Import "Sheet1" from Excel TestImportExcelSheet.xlsx  to local Datatable "Action1"

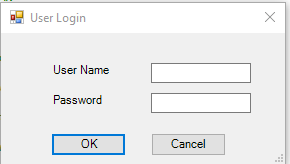
Example for accessing items:

First of all, it is important to set the current row, specified by RowNumber (ex.: 1) and after this select the element using DataTable(“Action”, Global”).



## Login UFT()

This function will generate a pop-up “User Login”. The pop-up is not from DiagBox and the user has to complete it with username and password.



Depending of what button the user will select, will happen as follow:

* **OK** - The data introduced by user will be saved in a structure. If Authentication function is called without parameters username and password, the data from structure will be used**.**
* **Cancel** - No data is saved.

## LaunchDiagBox()

Launches the DiagBox application and manages the update request pop-up. In the case of pop-up updating, the application will click on the cancel button.

## Authentification(Action, User, Password, Brand)

This function permit the user authentication and can be run in any part of the script execution. It requires several parameter, as defined in this order:

* **Action**: it represent the user action and can have one of the three following values (other values are not accepted, otherwise the function will generate an error):
  + OK (for validating of user-data), User and Password are obligatory according to two possibilities:
    - The User and Password parameters are left blank in the function, and in this case the parameters are recovered thanks to the data present in UFT (brick B0000).
    - The User and Password parameters are not left blank in the function and in this case the parameters recovered in the B0000 are ignored in favor of the new parameters entered. Ignored parameters are not stored.
    - In case of “User” and “Password” parameters are empty, then authentication will be done by using the password and the user from the Login Uft.
    - Cancel (for not validating of user-data), doesn’t need the parameters: User, Password, Brand.
* **User**: it represents the user code
* **Password**: it represents the password associated to the given user code
* **Brand**: it represent the brand the user is diagnose and can have one of the following values (other values are not accepted):
  + CITROËN
  + DS
  + FENGSHEN
  + PEUGEOT

Special use-cases:

1. When the given action is either *Cancel* or *Delete*, the user will not have to fill the other data
2. In some screens, when prompting the user authentication pop-up, the brand is already filled in; in this case, the user will not have to specify this parameter

Errors:

1. In case of Authentication Error pop-up, an error message will be added to the report, which will inform that the user or password is wrong.
2. The Authentication Error pop-up can also appear if the brand is incorrect. Also, an error message will be added to the report, which will inform that the brand in wrong.
3. Another popup appears when the action is not for validation, cancellation and deletion. An error message will be added to the report with this issue.

## BrandSelect(Brand)

Using this function, the user can select the brand (car-maker) for which the test will be executed.

It has only one parameter that must be defined which represents the brand name, and that can have one of the following values:

* CITROËN
* DS AUTOMOBILES
* PEUGEOT
* FENGSHEN

Error:

1. The program execution will be stop if the brand is wrong and does not have the values: CITROËN, DS AUTOMOBILES, PEUGEOT or FENGSHEN. An error message will be added to the report and will tell to the user that the brand is wrong.

## ModelSelect(DetectionType)

This function will select a specific car model, depending on the previously selected brand. The vehicle selection will be done depending on the value of the given input parameter, as it follows:

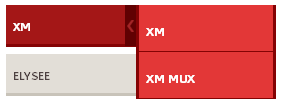
* **AUTO**: the DiagBox tool will try and read the VIN number of the vehicle connected to this tool
* **The VIN number**: the DiagBox tool will input the VIN number (alphanumeric string formed from 17 characters) corresponding to the vehicle connected to the tool
* **Model name**: represents the name of the vehicle that the user wish to connect to

Special use-case:

1. In the case when the user wants to access a specific vehicle using the third method (specifying the car model name), if the mentioned vehicle model has more versions, the vehicle and the version will be separated by the “;” character.

Example:

For the case below:



If the user wants to select *XM MUX* vehicle, then the function will be called with the following parameter: “XM;XM MUX”.

Errors:

The test will stop and will add errors in Report when:

- the button AUTO is not pushed

- the name of the vehicle given as parameter is not correct

WiFiButton(Name)

This function will select << WiFi >> button and depending on the parameter *Name,* it will do as follow:

* **CANCEL**: the button **Cancel** of the pop-up opened by clicking << WiFi >> button will be pushed
* **RESTART**: the button **Restart** of the pop-up opened by clicking << WiFi >> button will be pushed

If the pop-up "AUTHENTICATION" appears the button "CANCEL" of it will be pushed.

Errors:

The test will stop and will add errors in Report when:

- the given parameter is NOT "CANCEL" or "RESTART"

- the buttons "CANCEL" or "RESTART" aren't pushed.

## LaunchApplication(Operation)

This function will select one of the operations possible in DiagBox, depending on the given input parameter, as follows:

* **Delivery**: it will select the “Delivery vehicle to customer” operation
* **Repair**: it will access the “Repair” function
* **Fault**: it will access the “Fault Finding” section
* **Maintenance**: it will access the “Maintenance” operation

After launching “Fault Finding” section, there will be the possibility of getting the Authentication pop-up. In this case it will be pressed the “Cancel” button from the pop-up, to skip the authentication.

Error:

1. The program execution will be stop if the operation is wrong and does not have the values: Delivery, Repair, Fault Finding or Maintenance. An error message will be added to the report and will tell to the user that the operation is wrong.

SelectButton(Action)

This function will click on various buttons located in the interface, depending on the action received as a parameter.

The action will be one of the following actions:

* **OK**
* **BACK**
* **CANCEL**
* **RESET** : reinitialize in the authentication pop-up
* **JDD**
* **REFRESH**

Sometimes the interface may contain more buttons with the same action. In object repository they will have different names.

Example:

If we in the interface contains one button "OK" and in repository it is named "OK\_BTN\_2", the function will try to push on "OK\_BTN" first time and if it doesn't exist, "OK\_BTN\_2" will be pushed.

Errors:

The test will stop and will add errors in Report when:

- the button given as parameter will not be pushed.

## SelectTab(Name)

This function will select one of the available tabs in DiagBox interface, depending on the tab name:

* **HOME**: will select the “Home” tab
* **DOCUMENTATION**: will select the “Documentation” tab
* **EXPERT**: will select the “Expert” tab
* **MEASUREMENTS**: will select the “Measurements” tab
* **REPORTS**: will select the “Reports” tab

Error:

1. The program execution will be stop if the Name is wrong and does not have the values: Home, Documentation, Expert, Measurements or Reports. An error message will be added to the report and will tell to the user that the tab name is not recognized.

## SelectECU(Family, SubFamily)

This function will select one ECU, depending on the given parameter *Family*.

Errors:

The test will stop and will add errors in Report when:

- the parameter SubFamily has as value "Communication error" or "Not present"

- in the text is not find the string Family & " " & SubFamily

## SelectMenu(Name)

This function will select the menu, given as parameter.

* **Name**: represent a list of actions for a specific computer

Error:

1. The program execution will be stop if the Name is wrong and does not have the values from the list. An error message will be added to the report and will tell to the user that the name is wrong.

## SelectSideMenu(Name)

Using this function, the user can select from SideMenu the menu, given as parameter.

Errors:

The test will stop and will add errors in Report when:

- the SideMenu is not clicked

- the value of parameter Name is not in list of SideMenu

## CaptureScreen(Report)

Using this function, the user will be able to do a screenshot of the DiagBox interface. Depending on the value of the input parameter (*True* or *False*), the screenshot will be present in the test result report.

## SeeTG ()

This function selects the button “View the Global test in Standard mode”.

cid:image001.png@01D4C48A.32E97A80

Error:

1. The program execution will be stopped if the button doesn’t exist in the page and an error message will be added in the report.

## Impression ()

This function selects the button “Generate the print report”.

cid:image002.png@01D4C48A.32E97A80

Error:

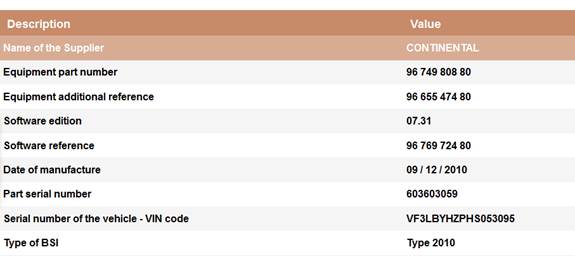
1. The program execution will be stopped if the button doesn’t exist in the page and an error message will be added in the report.

## TestIdent(ParamName, Format, DataType) ()

This function is used to test a parameter from the IDENTIFICATION menu. The input parameters are:

* **ParamName**: the name of parameter that has to be verified
* **Format**: type of value (Integer, String or Date)
* **DataType**: the form of parameter

Example:



For parameter “Equipment part number”, DataType = “XX XXX XXX XX” and Format = “Integer”

Call of function: **TestIdent(”Equipement part number”, ”Integer”, ”XX XXX XXX XX”)**

Error:

1. The program execution will be stopped if the Parameter doesn’t exist in the page or the value of the ParamName has an incorrect format. An error message will be added in the report.

## TestDTC()

This function will read and print the present faults from the menu “FAULTS READING”.

## EFFDTC ()

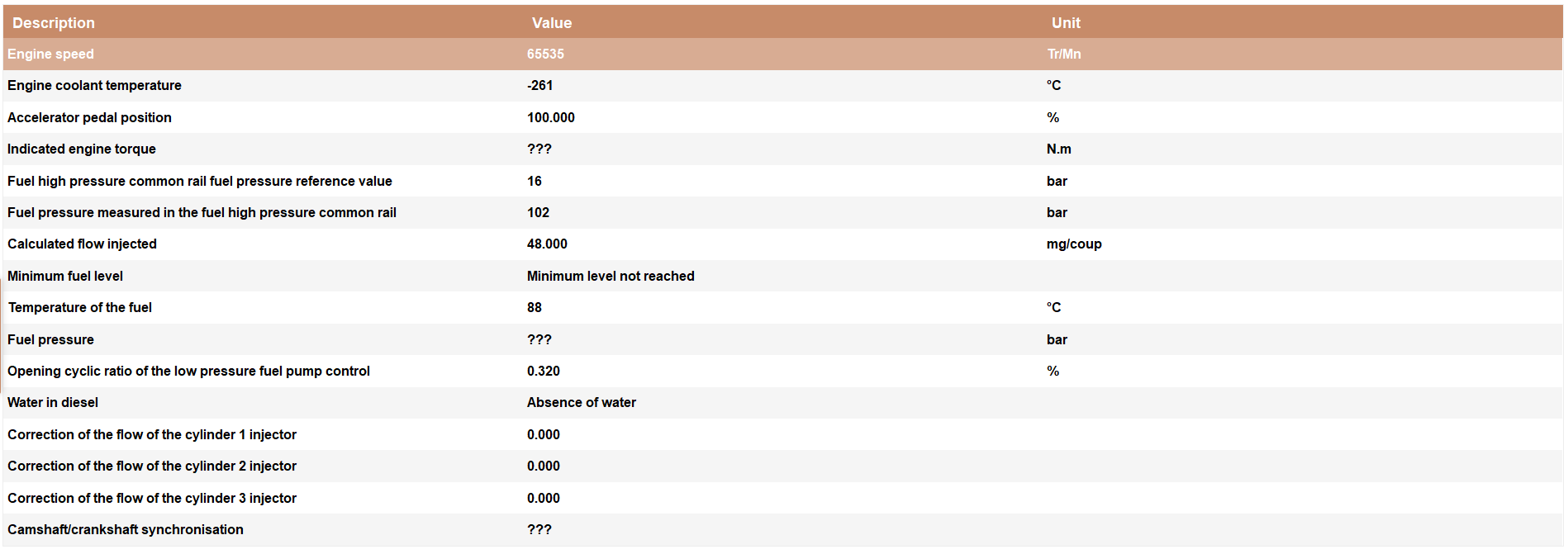
In this function if the delete button exists in the menu “FAULTS READING”, then all the faults found will be deleted.

## TestMP(ParamName, DataType, Format, Unit, Help)

This function is used to test a parameter in the “STANDARD PARAMETERS MEASUREMENTS” menu. The input parameters are:

* **ParamName**: the name of parameter that has to be verified
* **DataType**: the form of parameter
* **Format**: type of value (Integer, String or Date)
* **Unit**: measurement unit of the ParamName
* **Help**

Example:



Call of function: **TestMP (”Engine coolant temperature”, ”-XXX”, ”Integer”, ” °C”, ””)**

Error:

1. The program execution will be stopped if the Parameter doesn’t exist in the page, the value of the ParamName has an incorrect format or the unit is not right. An error message will be added in the report.

## Delay(Time)

This function will pause the execution of the code depending the time passed in parameter "Time". The last value shown is true until changed.