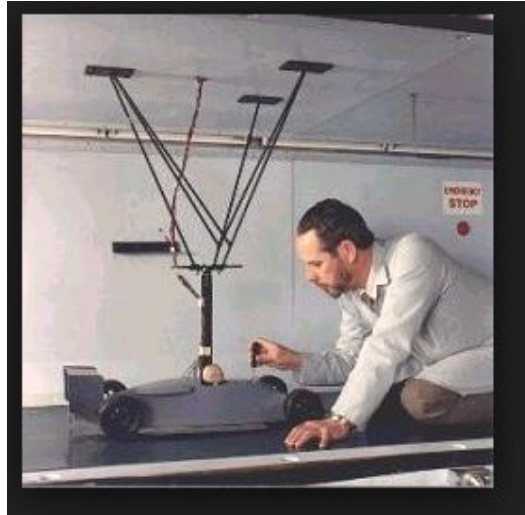


## UFT – Regular expression, recovery and reporting



“Take the first step in faith. You don't have to see the whole staircase, just take the first step.”

- Martin Luther King, Jr. quotes

[illegible]


### Step-3 Verify the result

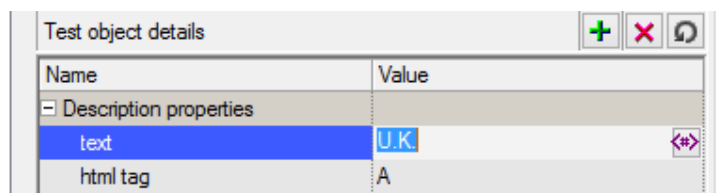
 *Navigate to "UFTRegExp" in UFT | View | Last Run results | ALT + V + X*

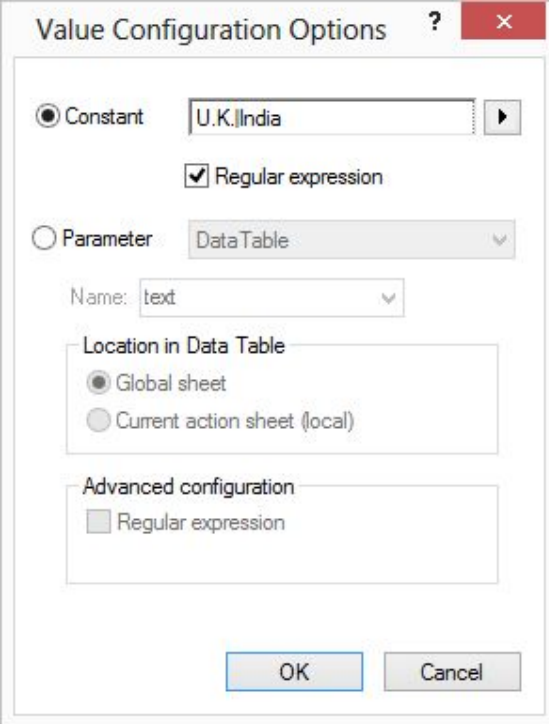


### Step-4 Fix the issue and rerun

This problem is appearing because there is no U.K. link present on the new.google.co.in page. To rectify this error, we need to make the object property dynamic to accommodate the change. If we can provide the link text property to accept either "U.K" or "India" then this script will work.

 *Navigate to "UFTRegExp" in UFT | Resource | Object Repository | U.K. Link | Click on text property | Click on <#> | Check Regular Expression box | No | provide text "U.K.|India" in the Constant field | OK | Close | Save | Run*





The dialog box is titled "Value Configuration Options" with a question mark icon and a close button. It contains two main sections: "Constant" and "Parameter".

- Constant:**
  - Selected with a radio button.
  - Text input field: "U.K.|India".
  - Checkmark for "Regular expression".
- Parameter:**
  - Not selected with a radio button.
  - Dropdown menu: "Data Table".
  - Text input field: "Name: text".
  - Section "Location in Data Table":
    - Selected with a radio button: "Global sheet".
    - Not selected with a radio button: "Current action sheet (local)".
  - Section "Advanced configuration":
    - Not selected with a checkbox: "Regular expression".

At the bottom are "OK" and "Cancel" buttons.

### Concept:

In STEP1, we are creating UK news specific test. In Step2, we are modifying the datatable to accommodate more than one country. In STEP3, we run the test that eventually fails for .co.in domain. This test design to find .co.uk but during run-time it meet with .co.in and there was no UK news section. In STEP4, we fix the above issue by “regularize” text property of UK link object to accommodate UK or India (by using or “|” special character).

A regular expression is a pattern string that can accommodate many complex search phrase. Regular expression use special characters, such as period ( . I.e. any one character), asterisk (\* - 0 or more character), caret (^ - Starting with), and brackets ([ ] – one of them) to make the pattern.

Regular expressions are used to identify objects, checkpoint and text strings with dynamic values. Suppose, we navigate on home page after login into site with page title Welcome <username>. In this case Object page title property will change for each user. We should regularize the <username> text with pattern to accommodate any word characters (\w).

Following are the most common pattern used in regular expression.

Symbol	Function
[xyz]	Match any one character enclosed in the character set. "[a-e]" matches "b" in "basketball".
[^xyz]	Match any one character not enclosed in the character set. "[^a-e]" matches "s" in "basketball".
.	Match any character except \n.
\w	Match any word character. Equivalent to [a-zA-Z_0-9].
\W	Match any non-word character. Equivalent to [^a-zA-Z_0-9].
\d	Match any digit. Equivalent to [0-9].
\D	Match any non-digit. Equivalent to [^0-9].
\s	Match any space character.
^	Only match the beginning of a string. "^A" matches first "A" in "An A+ for Anita."
\$	Only match the ending of a string. "t\$" matches the last "t" in "A cat in the hat"
{x}	Match exactly x occurrences of a regular expression. "\d{5}" matches 5 digits.
?	Match zero or one occurrences. Equivalent to {0,1}. "a?s?b" matches "ab" or "a b".
*	Match zero or more occurrences. Equivalent to {0,}.
+	Match one or more occurrences. Equivalent to {1,}.
()	Grouping a clause to create a clause. May be nested. "(ab)?(c)" matches "abc" or "c".
	Alternation combines clauses into one regular expression and then matches any of the individual clauses.

Let's explore the common scenario where string changes from design time to run time.

Date	Time	Email	Welcome message	Model Numebr
Order number	ticket number	Mobile number	Phone number	Vouchernumber
Address	post code	machine name	user name	Year
domain name	country name	Prefix	suffix	host name
file name	version number	result number	trading number	virtaul server
companyname	employee name	employer name	site name	IP addresses
transaction id	title	links	object property	color name
Modify date	Access date	Street name	coordinated	user ID

This is an example to let you think that we are surrounded by the dynamic strings that eventually we need to process during our automation tasks. Regular expression is not VBScript phenomena but it is generic concept applicable in nearly all popular languages including JAVA, UNIX and SQL etc. Luckily the meaning of special character also do not vary much from one language to other and you can spot the pattern of regular expression in most of the time. UFT support regular expression building by using GUI.

Let's explore some valid pattern

#### 1. valid e-mail address

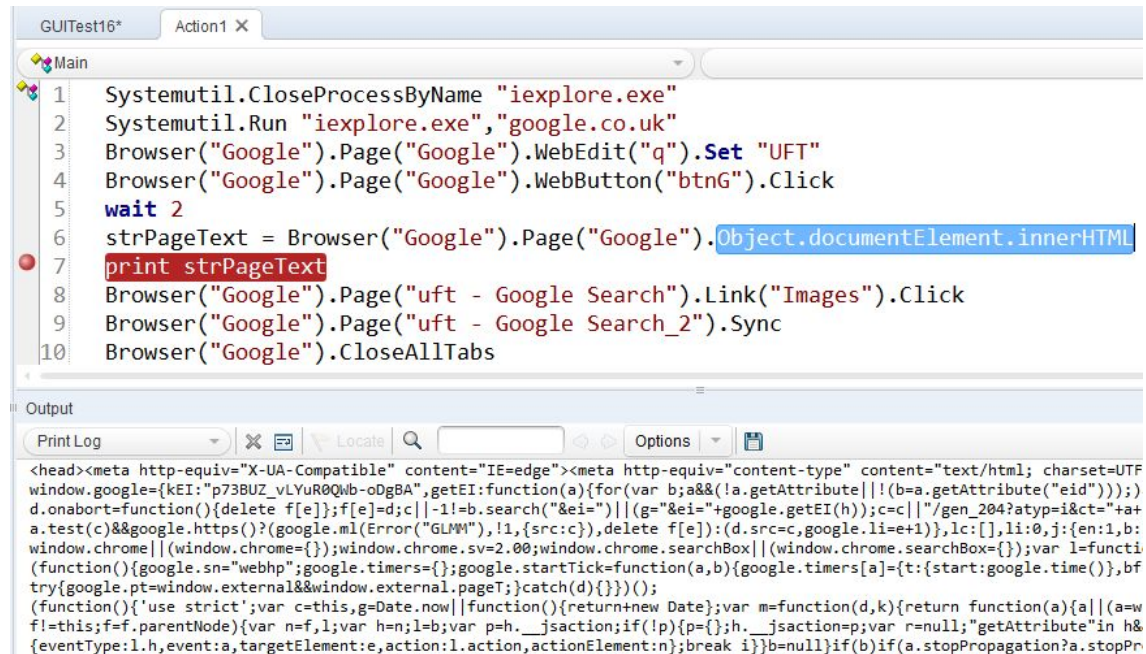
```
a. ^([\w-]+\.)*[\w-]+@([\w-]+\.)+[a-z]{2,4}$
```

2. Date:  
(0[1-9]|1[012])[-./](0[1-9]|1[12][0-9]|3[01])[-./](19|20)[0-9]{2}
3. HTML tag  
<([A-Z][A-Z0-9]\*)\b[^>]\*>(.\*?)</\1>
4. IP Address  
\b(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.\{3\}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b
5. Credit- Card  
^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13}|3(?:0[0-5]|[68][0-9])[0-9]{11}|(?:2131|1800|35\d{3})\d{11})\$
6. Match a blank line  
"^\s\*\$"

### 3.0 Regular Expression in Script:

In previous section we have seen that object property can regularize by providing the desired pattern. Sometime it is necessary to handle the script with use of regular expression

Step-1 Record test to get the source code of Google page



```

GUI Test16*   Action1 X
Main
1  Systemutil.CloseProcessByName "iexplore.exe"
2  Systemutil.Run "iexplore.exe", "google.co.uk"
3  Browser("Google").Page("Google").WebEdit("q").Set "UFT"
4  Browser("Google").Page("Google").WebButton("btnG").Click
5  wait 2
6  strPageText = Browser("Google").Page("Google").Object.documentElement.innerHTML
7  print strPageText
8  Browser("Google").Page("uft - Google Search").Link("Images").Click
9  Browser("Google").Page("uft - Google Search_2").Sync
10 Browser("Google").CloseAllTabs
  
```

Output

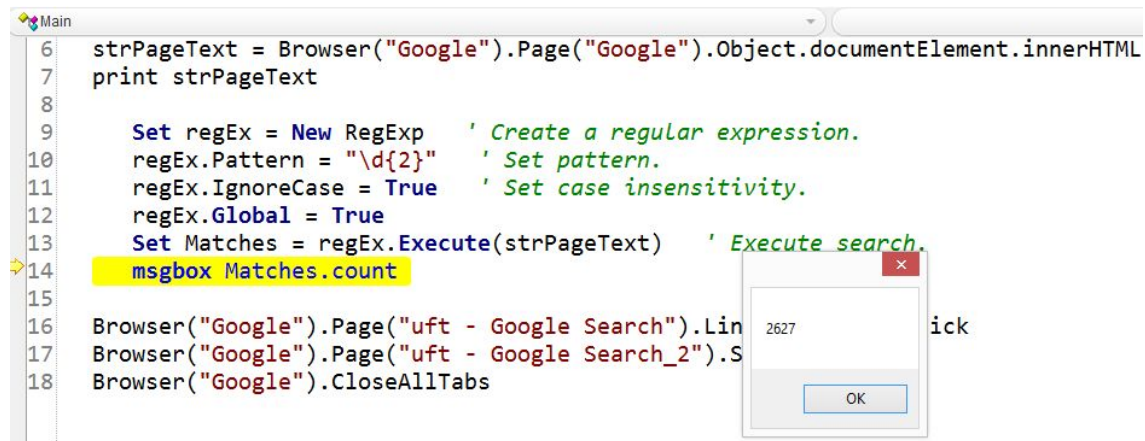
```

<head><meta http-equiv="X-UA-Compatible" content="IE=edge"><meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script>window.google={kEI:"p738UZ_vLYuR0QWb-oDgBA",getEI:function(a){for(var b;a&&(!a.getAttribute||!(b=a.getAttribute("eid")));)
d.onabort=function(){delete f[e];f[e]=d;c| -1!=b.search("&ei=")|| (g="&ei="+google.getEI(h));c=c||"/gen_204?atyp=i&ct="+a+
a.test(c)&&google.https()?(google.ml(Error("GLMM"),!1,{src:c}),delete f[e]):(d.src=c,google.li=e+1),lc:[],li:0,j:{en:1,b:
window.chrome||(window.chrome={});window.chrome.sv=2.00;window.chrome.searchBox||(window.chrome.searchBox={});var l=functi
(function){google.sn="webhp";google.timers={};google.startTick=function(a,b){google.timers[a]={t:start:google.time()},bf
try{google.pt=window.external&&window.external.pageT}catch(d){}}();
(function){'use strict';var c=this,g=Date.now||function(){return new Date};var m=function(d,k){return function(a){a|| (a=w
f!=this?f=f.parentNode){var n=f,l;var h=n;l=b;var p=h.__jsaction;if(!p){p={};h.__jsaction=p;var r=null;"getAttribute"in h&
{eventType:l.h,event:a,targetElement:e,action:l.action,actionElement:n;break i}}b=null;if(b)if(a.stopPropagation?a.stopPr
  
```

Step -2 Search the source code with regular expression

2.1 Find the two digit number on google source page





2.2 Modify the line `regEx.Pattern = "\d{2}"` to `regEx.Pattern = "\w"` and run the code. It will search alphanumeric word.

2.3 Modify the line `regEx.Pattern = "\w"` to `regEx.Pattern = "UFT"` and run the code. It will search UFT string in text

2.4 Modify the line `regEx.Pattern = "\w"` to `regEx.Pattern = "UFT|class"` and run the code. It will search either of string.

2.5 Modify the line `regEx.Pattern = "UFT|class"` to `regEx.Pattern = "\\` and run the code

### Concept

In STEP1, we get the browser source code of the Google page by using HTML document object model (DOM). In same manner we can get source code of any page. SO, if we want to find any string on the page then this way we can search text from the whole source code of the page. In STEP3, we are creating a regular expression object and setting the patter `{\d{2}}` i.e. two-digit number. We also setting other properties of regular expression e.g. "Ignorecase" and "Global". Global indicates if a pattern should match all occurrences (true) in an entire search string or just the first one (false).

In STEP2.2, we change the pattern from two-digit to any word (`\w`). In STEP2.3, we are searching "UFT" without any special character so it will only match "UFT". In STEP2.4, we are searching either "UFT" or "class" string.

In STEP2.5, we need to search "`\`" but it is itself a special character. If we need to search the special character then it is necessary to escape it with "`\`" so suppose if we are searching `*` in the text while `*` itself is wild character consider in pattern. To take the literal value (i.e. `*` consider as `*` instead of occurrence of 0 or more character) of special character we need to escape it with "`\`".

So, if we need to search a digit then use "`\d`" pattern and if we need to search `\d` in string then use "`\\d`" pattern.

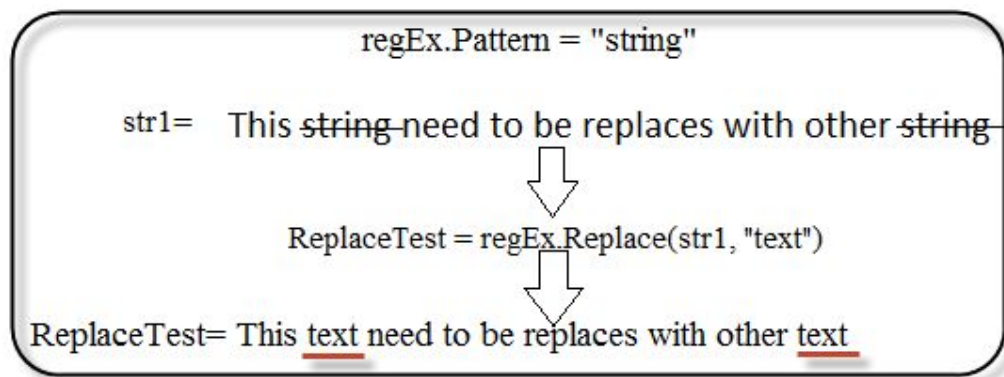
**Let's explore RegEx Object, pattern and Matches**

We can handle, manipulate, test and replace the text in string based on the pattern. VBScript provide RegExp object for regular expression. First we need to define the Object and then associate property values e.g. pattern, Ignorecases etc with the object.

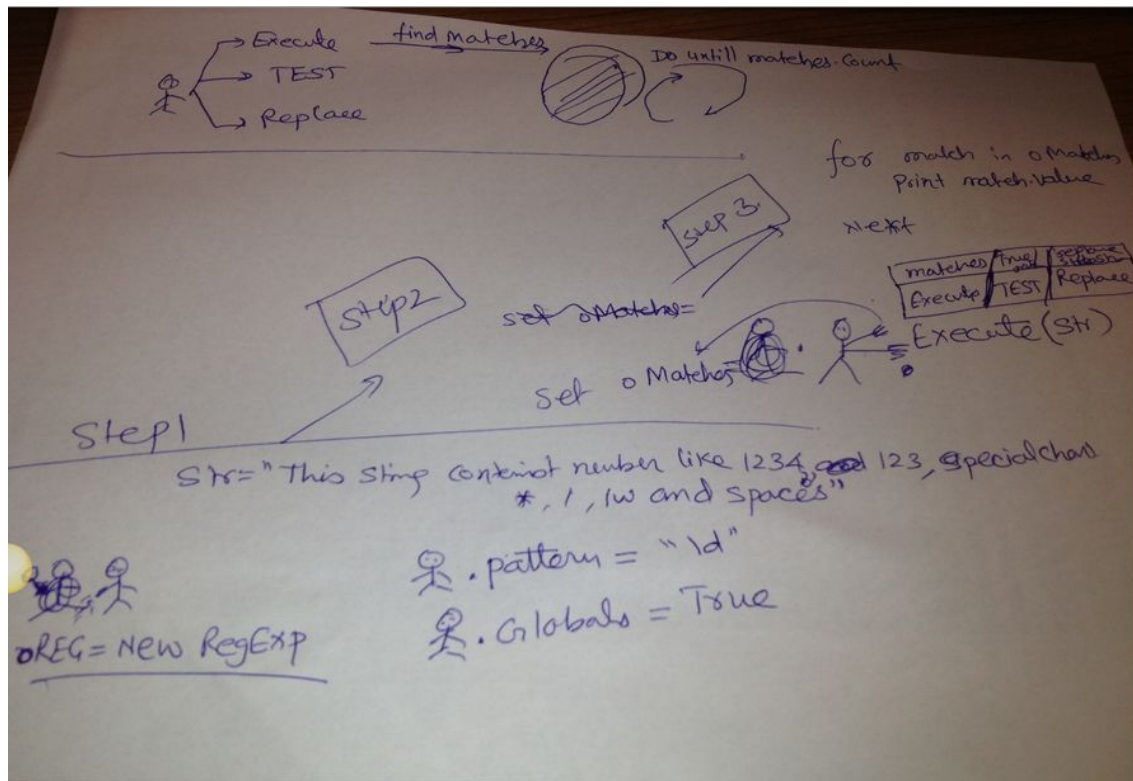
```
Set regEx = New RegExp  
regEx.Pattern = "\d" 'To search a digit  
regEx.IgnoreCase = True
```

Now this regEx object can call any of these method

1. regEx.Execute(str) - This method return the matches in the str based on the pattern associated with the regEx object. Matches is also an object that can have multiple properties.
2. regEx.Test(str) - If pattern exist in str then it will return true otherwise false.
3. regEx.Replace(str, replStr) - It will replace str to replStr wherever it will find the pattern.







### Matches by Execute method

patrn = "\d"

strng = "This 1 string need to search by given pattern , It can have special characters like \, \w, \d and \*"

```
Dim regEx, Match, Matches ' Create variable.
Set regEx = New RegEx ' Create a regular expression.
regEx.Pattern = patrn ' Set pattern.
regEx.IgnoreCase = True ' Set case insensitivity.
regEx.Global = True ' Set global applicability.

Set Matches = regEx.Execute(strng) ' Execute search.

For Each Match in Matches ' Iterate Matches collection.
    MatchStr = MatchStr & Match.FirstIndex & ". Match Value is '"
    MatchStr = MatchStr & Match.Value & "'." & vbCrLf
Next
```

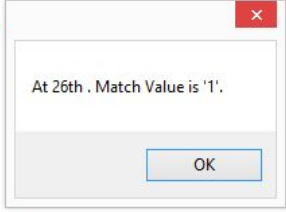
```

patrn = "\d"
strng = "We are finding digit like 1 int this string"
Dim regEx, Match, Matches ' Create variable.
Set regEx = New RegExp ' Create a regular expression.
regEx.Pattern = patrn ' Set pattern.
regEx.IgnoreCase = True ' Set case insensitivity.
regEx.Global = True ' Set global applicability.

Set Matches = regEx.Execute(strng) ' Execute search.

For Each Match in Matches ' Iterate Matches collection.
    MatchStr = "At " & MatchStr & Match.FirstIndex & "th " & ". Match Value is '"
    MatchStr = MatchStr & Match.Value & "'." & vbCRLF
MsgBox MatchStr
Next

```



## A Note on Match Object

When we execute regular expression with any pattern, it returns match object. Match object supports FirstIndex Property, Length Property and Value Property. There may be more than one matches by the specified pattern. A SubMatches collection contains individual submatch strings from the matches.

```

Set Matches = oRegExp.Execute(inpStr)
Matches.SubMatches(0) ' Returns first string of the match collection.

```

## A recap on Backslash Character

A backslash (\) has special meaning in regular expression. There are several special characters for regular expression. But what if those characters themselves become a part of text. Backslash forces the VBScript engine to consider those characters with their face value i.e. \. will be treated as . while. will be considered as a wild character.

The escape sequence like \n, \t etc. have special meaning but backslash can force it to be considered as a literal instead of with special meaning. So the special characters, such as the letters n, t, w, or d, should be used with “\” as per the requirement. Few examples are given below:

```

\w is a special character that matches any word character including underscore
\\ matches the literal character \
\( matches the literal character (
QuickTestPro\.co\.uk

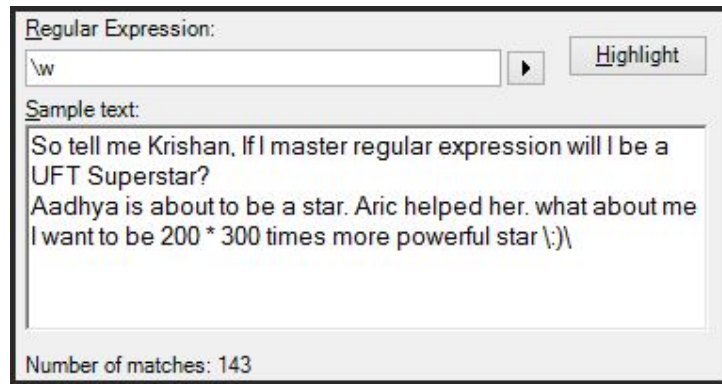
```

## Regular Expression Evaluator

UFT provides Regular Expression Evaluator to create and test a regular expression to determine whether it suits your needs.



[Navigate to UFT \(No need to create test\) | Tools | Regular Expression Evaluator | Provide your sample text | pattern “\w” | Highlight | try following patterns](#)



Common Patterns							
\*	?	\?	\s	Krishan	\d	\d{2}	[0-9]0*
.*uper*	^Aadhya	)	\)	star?\$	star\?\$	\W	\D

<revise delete either image or this >

Common Patterns							
\*	?	\?	\s	Krishan	\d	\d{2}	[0-9]0*
.*uper*	^Aadhya	)	\)	star?\$	star\?\$	\W	\D

## Smart Regular Expression List

UFT provides smart regular expression wizard to assist us to develop pattern comfortably. This wizard contains most used regular patterns.

STEP1: Record a test



*Go to UFT | File | New | Test | Create new GUI Test as "SmartRegEx" | Create | Record | Select "Record and run test on any browser" | OK | Press F5 key | Click on IE | "google.co.uk" | Search "Krishan" | Close | Stop | Resources | Object Repository | Go to object "q" | Select <> | Check Regular Expression | Click ▶*

## Recover Scenario

Automation runs without manual intervention and execution bound to face the challenges of unexpected errors, events, error and sometime application crash. These error can leave test to fail and even hinder other test to execute normally afterwards. Mr. UFT rescue us from such undesirable but unexpected events by using recovery scenario. UFT enables us create recovery scenarios where we define the trigger criteria (e.g. unexpected pop-up). If error occurs during run-time then UFT invoke recovery scenarios to handle that error. So now we have to understand recover scenario manager and the relation of recovery scenario and unexpected events.



## UFT Recovery Scenario Exercise

STEP1: Let's create a dynamic pop-up with help of Wscript



👉 Open notepad | copy the following lines | save at "C:\Test\popup.vbs" | double click on "popup.vbs"

```
1 Set popUPWScript = CreateObject("WScript.shell")
2 popUPWScript.popup "popup come anytime","120","Error-app stuck"
```

```
Set popUPWScript = CreateObject("WScript.shell")
popUPWScript.popup "popup come anytime", "120", "Error-app stuck"
```

## STEP2: Create a Test to search “KrishanShukla”




Go to UFT | File | New | Test | Create new GUI Test as "RecoveryPopUP" | Create | Record | Select "Record and run test on any browser" | OK | Press F5 key | Click on IE | "google.co.uk" in URL | Search with "KrishanShukla" | Enter | Click on Image Link | Close browser | Stop | Save | Modify Test by giving 60 second wait after Step entering "KrishanShukla" in script

STEP3: Create a recovery scenario by recovery manager:



 [Navigate to "Recovery Popup" test in UFT | Resources | Recovery Scenario Manager | Click New Scenario](#)  [Next | pop-up window radio button | Next | either follow method 1 or method 2...](#)  
/

### Method-1

Double click on popup.vbs at "C:\Test\popup.vbs | Go to UFT and Click on  | Click on pop-up |

## Method-2

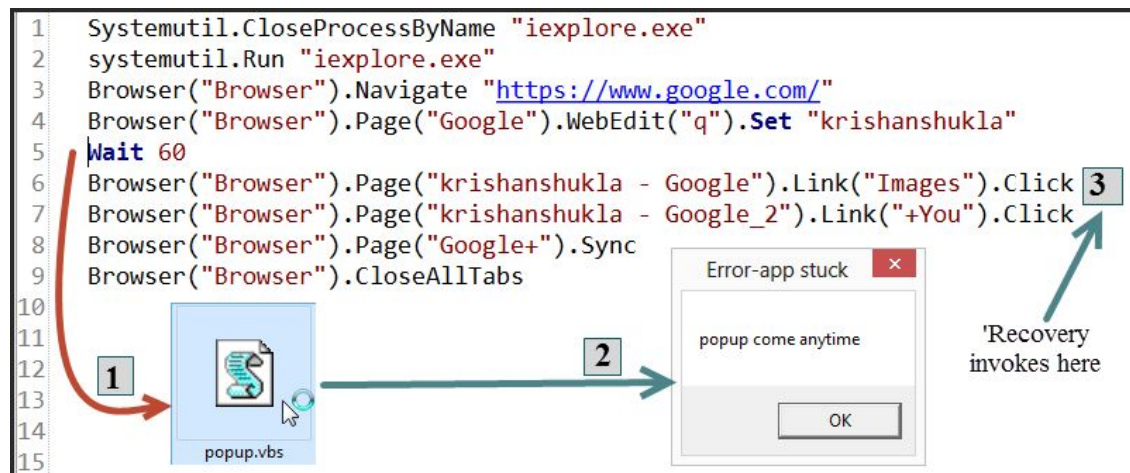
Check window title checkbox | provide "Error-app stuck" | Check Window text contains | provide "OKpopup come anytime" in the box |

...Next | Next | Select radio button "Keyword and mouse button" | Select radio button "Click on Default button/ Press the Enter Key" | Next | Uncheck "Add another recovery operation" | Next | Select "Proceed to next step" | Next | provide "Error\_App\_popup" in Scenario name | Next | Check both of check box "Add scenario to current test" and "Add scenario to default test settings" | Finish | close | Yes | Save scenario "RecoverySc" | Save | run

STEP4: Invoke the recoveryScenario



*Navigate to "RecoveryPopUP" test in UFT | Run | During execution, close the IE browser at step "KrishanShukla" and Go to "C:\Test\popup.vbs" and double click on "popup.vbs" | Stop test when test reach at line 7 | View | Last Run Results | ALT + V + X | wait for test to run completely*



```
1 Systemutil.CloseProcessByName "iexplore.exe"
2 systemutil.Run "iexplore.exe"
3 Browser("Browser").Navigate "https://www.google.com/"
4 Browser("Browser").Page("Google").WebEdit("q").Set "krishanshukla"
5 Wait 60
6 Browser("Browser").Page("krishanshukla - Google").Link("Images").Click
7 Browser("Browser").Page("krishanshukla - Google_2").Link("+You").Click
8 Browser("Browser").Page("Google+").Sync
9 Browser("Browser").CloseAllTabs
```

The diagram illustrates the execution flow of a test scenario. It includes a code editor with VBS code, a file icon for popup.vbs, and an error dialog box. Arrows and numbers 1, 2, and 3 indicate the sequence of actions:

- 1**: A red arrow points from line 5 of the code to the popup.vbs file icon.
- 2**: A green arrow points from the popup.vbs file icon to the "Error-app stuck" dialog box.
- 3**: A green arrow points from the "Images" link in the code to the "Recovery invokes here" text.

The "Error-app stuck" dialog box contains the text "popup come anytime" and an "OK" button.


Step Name: Error-app

Step Done

Object	Details
Error-app	<p><b>Scenario:</b> Error-app  <b>Defined in:</b> C:\UFT\Scenario\RecoverySc.qrs  <b>Description:</b>  <b>Post-recovery operation:</b> Proceed to next step.</p> <p><b>Activated by trigger:</b>  <b>Type:</b> Pop-up window  <b>Contains the caption:</b> Error-app stuck  <b>Contains the text:</b> popup come anytime</p> <p><b>The current test step details:</b>  <b>Object:</b> Link("Images")  <b>Method:</b> Click  <b>Arguments:</b> EMPTY  <b>Result:</b> Cannot identify the object</p>

## Concept

In STEP1, we have created a dynamic pop-up with help of Wscript. This pop-up will vanish in specified time. Remember this pop-up is not an error but automater logic. This example is just to show you that if any error comes (e.g. pop-up) then how we can handle them. In STEP2, we created a Google search test.

In STEP3, we created recovery scenario. Here we are handling pop-up by two method 1. Identify that object by  or 2. Define the object (pop-up) property. We used "Keyword and mouse button" to click on default pop-up option if it appears. We associate this recovery scenario definition to the current test and default test settings. Defining the recovery scenario will not invoke it. The recovery scenario will only run when it encounter the problem.

In STEP4, we have invoked the recovery by inducing the error during run-time. We started our UFT test and in middle of run invoke pop-up. We can find the recovery detail in UFT report.

Recovery scenarios are intended for the event which are uncertain or face synchronization issues. Recovery scenario slow down the performance. If there are known issues with application then it is always best to deal it in test itself by using if-else or Optional steps (OptionalStep\* – It also slow down the performance). Few common example where we can employ recovery scenario

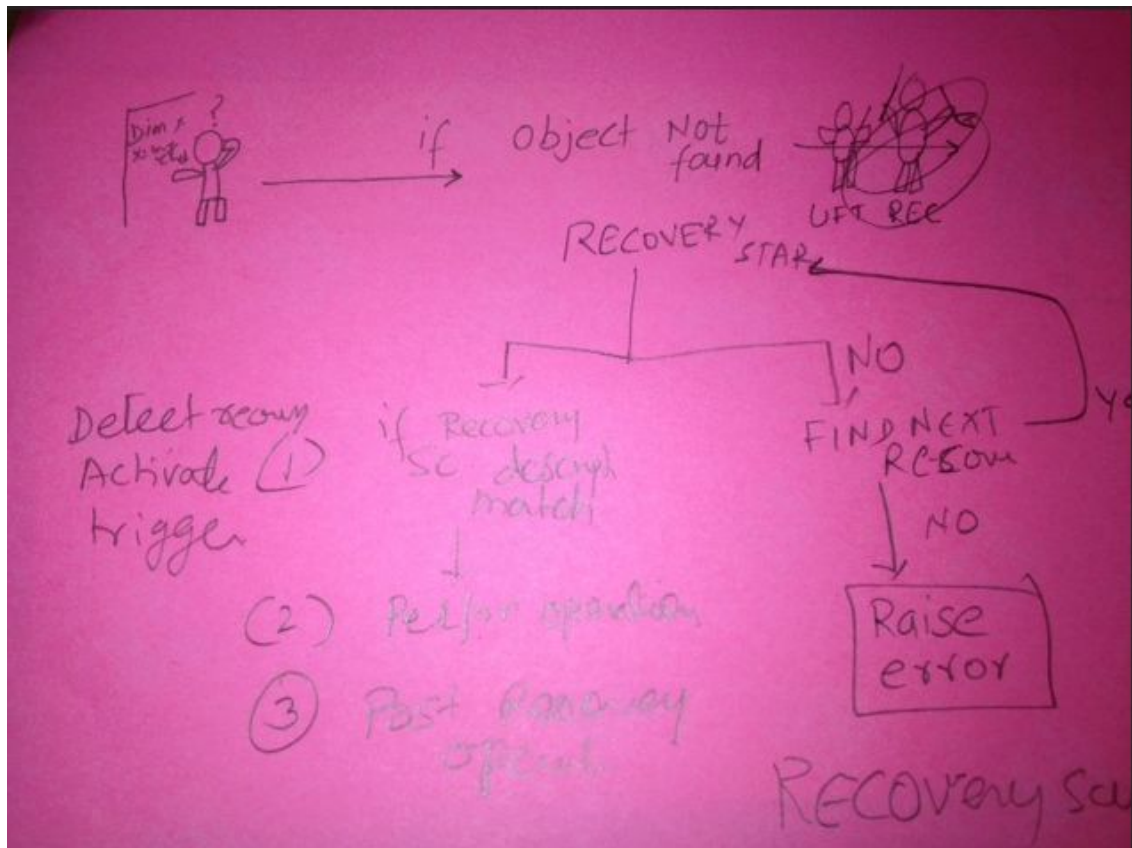
1. Security or domain pop-up errors
2. IE crash error
3. Printer error
4. Third party Interface error
5. System Error
6. Process interruption error and so on...

\*ConfuSense: Optional step will try to run the step but will not return error if it fail to do so.

OptionalStep.Browser("QuickTestPro").Page ("Training").WebEdit("UFT").Set "Yes"



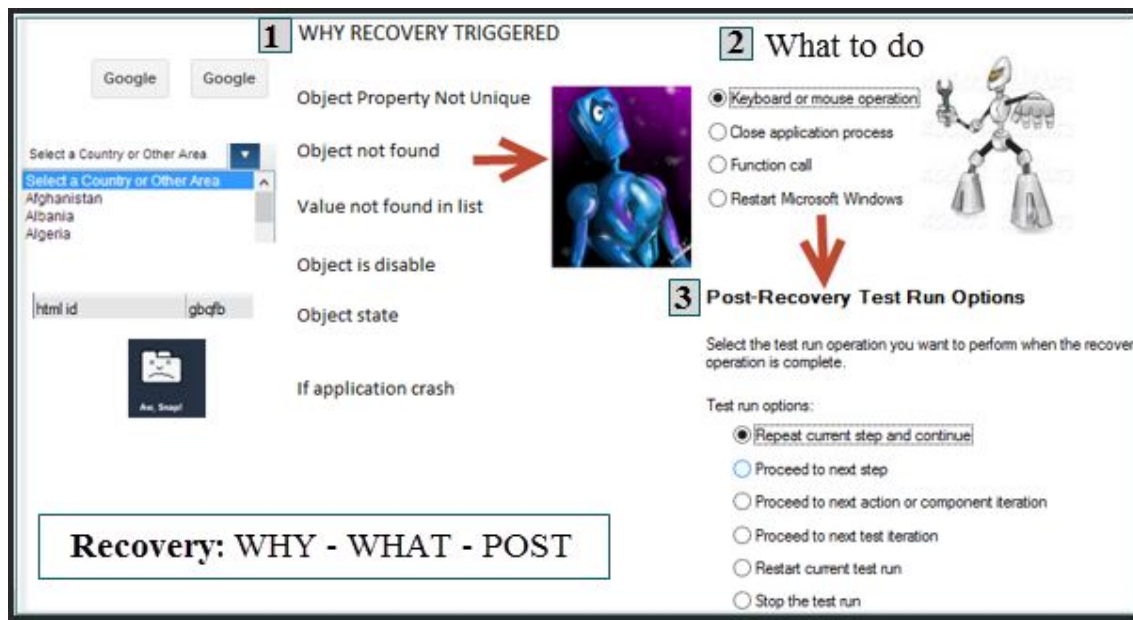
The following work-flow explains the recovery invocation process.



Recovery scenario process:

Recovery scenario consist three main steps

1. Why it happened - **Trigger Event**
2. What to do about it - **Recovery Operations**
3. Post recovery actions - **Post- Recovery Test Run Option**



### Trigger Event

There are major four categories of trigger event. 1. Unexpected pop-up 2. Object state 3. Test run error 4. Application crash. There are various types of pop-up including security pop-up, IT support patch over network, java script pop-up, additional authentication pop-up and browser specific pop-ups. We can register these pop-up to perform the recovery operations. Object state is also important way to find the error. Suppose we need to click on button but what if it is disable or we need to select web list but it has no item.

Test run errors such as "object not found", "Item not in list" can create lot of trouble during run-time. Do not confuse with object state and test run error. Object state specifically deal with particular object property while test run error considered for whole application e.g. "object not found" or "object is disable" is applicable for application-wide. Application (process name) crash is fourth and last recovery triggered event.

### Recovery Operations

This steps tells the action we should take during error encounter. There are four ways to handle any error 1. Keyboard or mouse operation (e.g. press enter) 2. Close application Process 3. Function Call 4. Restart windows (Yes, It works sometime☺).

Keyboard operation are pressing enter, escape or click object with specific label ("Yes" or "NO"). We can also provide the combination of keys (e.g. CTRL + ALT + DEL). Closing application process and restarting windows are straightforward. We can close any process (not only application but any process). Function call is the most versatile way to handle the error. We can create function and write our logic to react on certain errors. Let's explore the function call recovery.

**Problem Set:** Suppose we have country list (WebList). It works fine but if we are unable to find a country (e.g. "stralia") in that then we should invoke recovery function to select other country (e.g. "Germany").

## Exercise: Understanding Functional call recovery

STEP1: Create our own Webpage with country weblist



Navigate to "C:\Test" | Create text file | write the following code | Rename the file "Country.html" | Run the "Country.html" by double click on it

```
<HTML>
<HEAD>
  <TITLE> Select Country</TITLE>
</HEAD><script language=vbscript>
  <!--
  Sub showchange ()
    ' alert (document.form1.CountryList.value)
    MsgBox document.form1.CountryList.value
  End Sub
  -->
</script>
<BODY>
<form id="form1" name="form1">
<select id=CountryList size=2 onchange='showchange()'>
  <option value=1>Australia</option>
  <option value=2>USA</option>
  <option value=2>Germany</option>
  <option value=2>UK</option>
</select>
</form>
</BODY>
</HTML>
```

STEP2: Automate the weblist



Navigate to UFT | New | Test | FunctionRecoverTest | Record | Click on "Internet Explorer" | navigate to "C:\Test\country.html" | Click on the "Allow blocked content" if it appears | Select "Australia" | OK | Select USA | Ok | Close the browser | Stop | make "Allow blocked content" step as OptionalStep

```

Systemutil.Run "iexplore.exe"

'Both will work
'Browser("Browser").Navigate "C:\Test\Country.html"
Browser("Browser").Navigate "file://C:/Test/Country.html"

'Optional step - execute if present during run-time
OptionalStep.Browser("Browser").WinObject("Notification")._
.WinButton("Allow blocked content").Click
Browser("Browser").Page("Select Country").WebList("select").Select "Australia"
Browser("Browser").Dialog("VBScript").WinButton("OK").Click
Browser("Browser").Page("Select Country").WebList("select").Select "USA"
Browser("Browser").Dialog("VBScript").WinButton("OK").Click
Browser("Browser").Page("Select Country").Sync
Browser("Browser").CloseAllTabs

```


STEP3: Induce the error by changing the object property



*Open "C:\Test\countrylist.html" in notepad | Modify "Australia" to "stralia" | Save | Close |  
Navigate to "FunctionRecoverTest" in UFT | Run*

STEP4: Manage Recovery scenario



*Create a CountryRecovery.vbs in "C:\Test\Recovery" (create Recovery folder in Test folder if not present) | Navigate to "FunctionRecoverTest" in UFT | Resources | Recovery Scenario Manager | Click  | Next | Test Run Error | Select "Item in list or menu not found" | Next | Next | Function call | Next | "Country\_weblist" in functional library | browser and select "CountryRecovery.vbs" | Write the logic in the function as given below | Next | Deselect Add another recovery | Next | Repeat current step and continue | Next | "CountryRecoverySc" scenario name | Next | Check "Add Scenario to current test" and "Add scenario to default test settings" | Finish | Close | "Yes" to save | Save as MyCountryRecoverySc.qrs | Run Test | View | Last Run Results | ALT + V + X | Recovery*

```

Function RecoveryFunction1(Object, Method, Arguments, retVal)
Wait(2)
'Object here is any weblist where item not found
Object.select "#2" 'Third item in weblist
End Function

```

### Concept: Understanding Functional call recovery

In STEP1, we have create our own webpage. VBScript invented for web application but eventually become popular among system administration tasks. Here we are creating html page with weblist under form1. HTML page considered as a document and we can access any element on the html page by traversing on the path of that object e.g. to access the value of weblist under form1 we can use following syntax.

DOM 1.Treat HTML as Document Page 2.Form html tag 3.Object Id 4.Object Attribute

**1** **2** **3** **4**  
`document.form1.CountryList.value`

Document Object Model (DOM) is used to access webpages at run-time. HTML page can be treated as document. We can access the individual items from the scripting languages e.g. VBScript, JavaScript etc. by using DOM. Every web browser implement DOM differently so one code may not work on other browser.

In STEP2, We are automating our own application. If you notice the navigation path, we have provided the <file:///C:/Test/Country.html> but we can provide the ordinary path as well. This step may take little more time during replay due to security.

In STEP3, We induced an error. In practical world, application keeps change and object describing may differ from one version to other version. Here we are misconfigure the "Australia". Now our script will report error "Object not found" during run-time.

In STEP4, we have created our own logic to handle the situation when object not found in the weblist. Our recovery scenario stored in .qrs (Quicktest Recover Scenario) file. We can create one recovery scenario for an issue and add in multiple tests. So one known issue must define once and associate with many test with that susceptible error. Recovery scenario function call takes four parameter 1. Object 2. Method 3. Arguments and 4. retVal

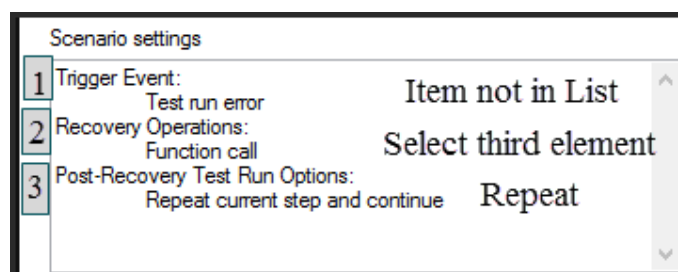
Object as Object: The object of the current step i.e. WebList in our example

Method as String: The method of the current step -

Arguments as Array: The actual method's arguments -

retVal - Result as Integer, The actual method's result.

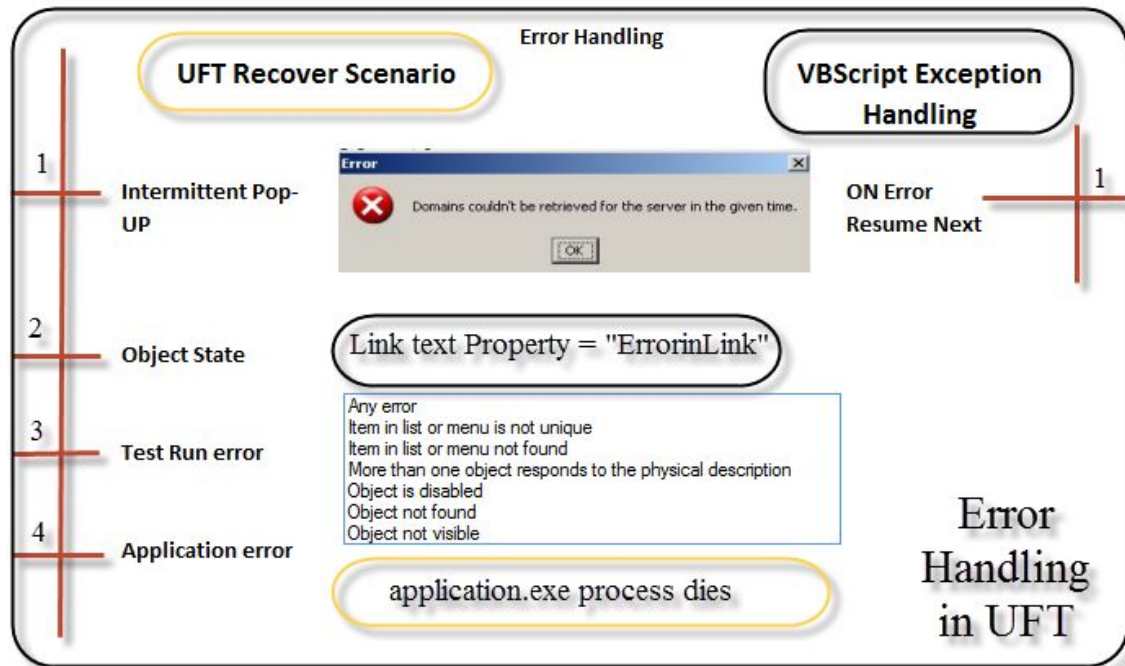
We can write a detail logic in the function call including capturing bitmap, closing processes etc. So, function call provide great flexibility to handle the error. The overall scenario comprises three steps as given below:



## VBScript Recovery Scenario:

UFT recovery scenario is very comprehensive and we should choose the appropriate recovery method based on the requirement for example pop-up should manage by pop-up trigger etc.

The following figure summarize the recovery scenario. VBScript also help to manage errors. VBScript have internal Error object ( so no need for object repository) that takes care the internal run errors.



## Error Object

When we run script, VBScript by default created intrinsic object "Err" i.e. we need not to create Err (or err) object but can directly use Err object during run-time. The default property of the Err object is Number i.e. **Err.Number** contains an integer. If script execution does not generate any error than Number remains zero otherwise it returns the number related to the error.

E.g. Err. **Number** will return 11 for "[Division by zero](#)" error.

## Exercise -



Go to UFT / File / New / Test / Create new GUI Test as "ErrorInTest" / Create as following / Save / Run

```
intFirstVal = InputBox("Enter first Int value , example 10")
intSecondVal = InputBox("Enter second Int value , example 20")
MsgBox intFirstVal / intSecondVal
```



Go to "ErrorInTest" in UFT / Run / provide 10 / provide 0 / Skip after getting error / Stop

```
intFirstVal = InputBox("Enter first Int value , example 10")
intSecondVal = InputBox("Enter second Int value , example 20")
```

```
MsgBox Err.Number
```



```
MsgBox intFirstVal / intSecondVal  
MsgBox Err.Number
```

## Concept of Exercise

In STEP1, we have create a simple program to calculate division of two input number. In STEP2, we induced error by “diving by zero”. It is important to note the Err.Number before and after the errors. There are detail error codes for err object and you can find it from the MSDN

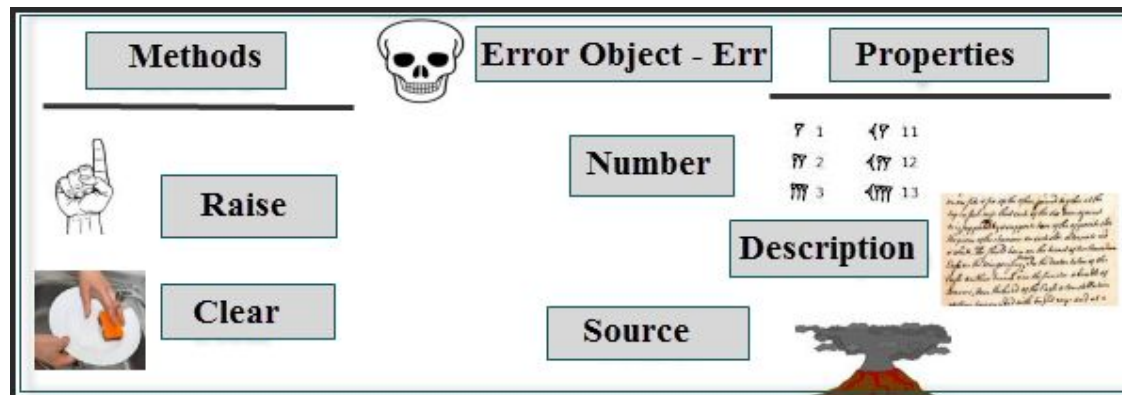
MSDN Reference for error code:

Run-Time Error - <http://msdn.microsoft.com/en-us/library/xe43cc8d.aspx>

Syntax Error - <http://msdn.microsoft.com/en-us/library/2z84dwk8.aspx>

Error object comprises many properties and methods to provide various information related to run-time errors.


Error Object has following methods and properties.



Let's do one exercise to understand all these method and properties.









## Exercise-

STEP1: Create a Test with Error object Methods


 [Go to UFT / Create the test as given below / Run](#)

```
On Error Resume Next  
Dim fvalue,svalue,calcValue  
fvalue = InputBox("Enter First value","User Input - 1",10)  
svalue = InputBox("Enter Second value","User Input - 2","0")  
msgbox err.Number  
msgbox err.source  
msgbox err.description  
calcValue = (fvalue / svalue)  
msgbox err.Number  
msgbox err.source
```

```
msgbox err.description
err.raise 5
msgbox err.Number
```

```
1 On Error Resume Next
2 Dim fvalue,svalue,calcValue  VBScript  creates internal Err Object 
3
4 fvalue = InputBox("Enter First value","User Input - 1",10)
5 svalue = InputBox("Enter Second value","User Input - 2","0")
6
7 msgbox err.Number  .Number =0 i.e. No Error
8
9 msgbox err.source  .Source i.e application that generated the error(Excel,VBS etc)
10
11 msgbox err.description
12
13 calcValue = (fvalue / svalue) "On Error" -  .Number set to error
14
15 msgbox err.Number  .Number =11
16
17 msgbox err.source
18
19 msgbox err.description
20
21 err.raise 5  .Raise will override .Number to USER preferred number
22
23 msgbox err.Number  .Number =5 now
```

Let's do one more exercise to understand all these method and properties usage.

 [Go to UFT | Create the test as given below | Run](#)

```

1 On Error Resume Next
2 Dim fvalue,svalue,calcValue,multiplyVal
3
4 arrVal=Array(5,10,15)
5 fvalue = InputBox("Enter First value","User Input - 1",10)
6 svalue = InputBox("Enter Second value","User Input - 2","0")
7
8 calcValue = (fvalue / svalue)           'First Error
9
10 If err.Number<>0 Then
11     MsgBox err.Number & " : Number error occured in calculation"
12 End If
13
14 err.clear                               'Clear will clear previous error
15
16 ' Out of index of array of three element
17 arrIndex = InputBox("Enter rray Index","User Input - 1",3)
18 multiplyVal = arrVal(arrIndex) * arrVal(arrIndex+1) 'Second Error
19
20 If err.Number<>0 Then
21     MsgBox err.Number & " : Number error occured in array"
22 End If

```

## Concept:

In STEP1, we have created a script to describe various error methods and properties. In STEP2, we emphasis on "On Error Resume Next"\* statement.

When VBScript program starts, Err object is like a blank page that contains no error with Err.Number equals to zero. When program encounter any error then new number assign to Err.Number object. Source property tells about the application from where error is originating. It returns usually class name or programmatic ID of the object that caused the error. So suppose if we process MS Excel that encounter any error (e.g. "Division by zero") then source set to excel. When generating an error from code, Source is your application's programmatic ID. Err.Description property consists of a description of the error. We can define our own description to alert the user.

Error object provides two methods - Raise and Clear. Raise method comprises five arguments where number is the only mandatory parameter.

Err.Raise (number, [source], [description], [helpfile], [helpcontext])

We can raise customize error numbers by using raise method. It will override the current number and provide opportunity for automater to code their own error codes for the known errors.

```

On Error Resume Next
MsgBox Err.Number 'Returns zero
l=0/0             'Induced error
MsgBox Err.Number 'System Generated error
Err.Raise 15 ' Override and raise an overflow error.

```

```

MsgBox ("Error #" & CStr(Err.Number) & " " & Err.Description)
Err.Clear ' Clear the error.
MsgBox Err.Number 'Reset error number

```

Clear method is very important method to reset the err.number. Clear method automatic calls when the following statement execute during run-time.

```

On Error Resume Next
Exit Sub
Exit Function

```

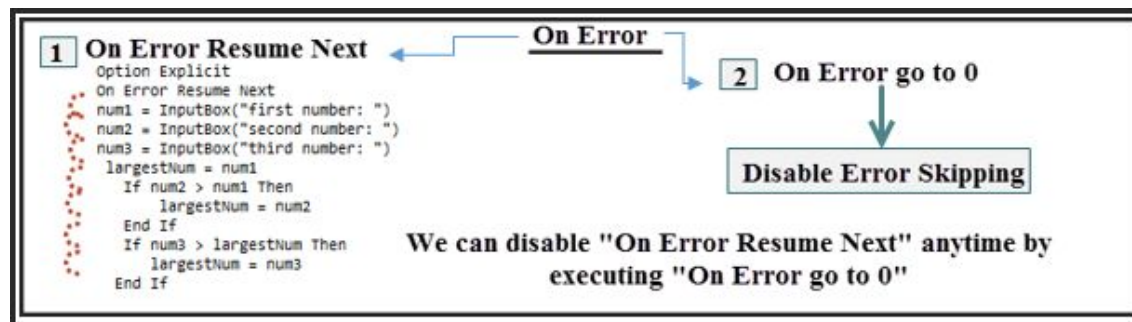


**\*ConfuSense** – VBScript is not case-sensitive language so upper or lower cases are valid statement.

## On Error Statements:

If we get the error than UFT halt execution and prompt for the error. To continue with the execution we should use “On Error Resume Next” statement. This statement will skip the step mired with error (as many time as it will find during script) and keep execution till end without any error pop-up.

The Err object's properties are reset to zero or zero-length strings ("" ) after an On Error Resume Next statement. The Clear method also be used to explicitly reset Err object.



## UFT Report

UFT produce report after test run. UFT creates new folder (default “res”) for each run to save test results. By default, result viewer opens automatically at the end of the test run. We can change this setting by deselecting “View results when run session ends” at Tools | Options | General tab | Run Sessions. We can also view results by opening the results.xml file during run-time. This xml prove helpful because we can analyse it at run time itself e.g. if we want to see the result of specific iteration before run continue for next iteration we can pause the run by using message box and verify the xml result file. The following statement will provide the result path.

```

MsgBox "Open the following file:" & Reporter.ReportPath

```

N.B: To view the partial results in the Run Results Viewer, you need to open the results file from another computer.

Run Results Viewer is a standalone application. It enables to share report with all stakeholder who do not installed UFT. The Run Results Viewer is installed and can be opened from Start > All Programs > HP Software > HP Run Results Viewer > Run Results Viewer.

## Exercise – Understanding UFT Report

STEP1: Create a Google search test

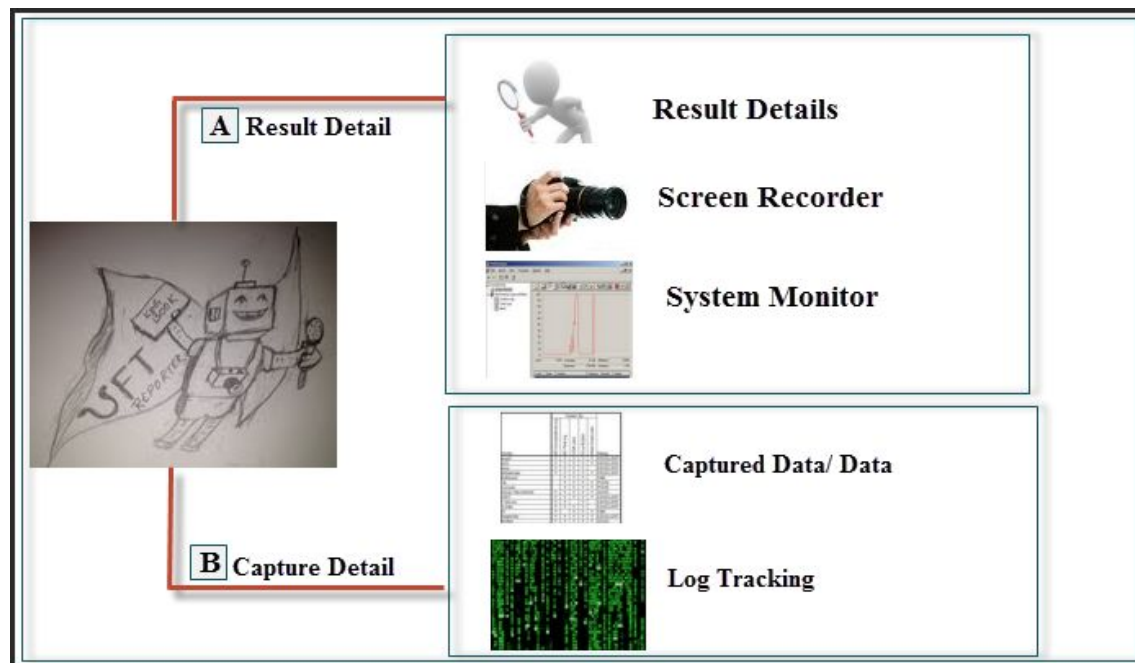
 Go to UFT | File | New | Test | Create new GUI Test as “Reports\_in\_UFT” | Create | Record | Select “Record and run test on any browser” | OK | Press F5 key | Click on IE | “google.co.uk” in URL | Search with “Krishan Wikipedia” | Enter | Click on Image Link | Close browser | Stop | Save | Modify Test by giving 5 second wait after Step entering “UFT” in script

STEP2: Run Test and verify report

 Go to “Report\_UFT” test in UFT | Run | View | Last Run Results | ALT + V + X |

### Concept:

In STEP1, we have created a Google search test. In STEP2, we navigated to the UFT results. UFT Result viewer captures the result detail, System performance, screen recorder and many more information.



We can export report in to various file format. We can navigate to Run Result Viewer | File | Export to File To verify the exported file format.

UFT Report Section	Supported File Format for Result Export
Step details	HTML (*.htm, *.html) (default) , PDF and DOC
Data Table	Excel (*.xls)
Log Tracking	XML (*.xml)
Screen Recorder	FlashBack (*.fbr)- Can view in HP Micro Recorder
System Monitor	Text (*.csv, *.txt) (default) , Excel, XML , HTML Note: Only the system monitor data is exported, not the graph

Result detail pane displays the details for an iteration, action and step. It contains the crucial information about environment, time and product version. It also shows current and previous run iteration so automater can compare both of the records. We can find the individual test steps detail as well in this pane. Recover information, smart identification and all other steps information also goes in result detail pane.

Screen recorder enables to view a movie of a run session. We can see partial or full movie. Screen capture happens only on the primary monitor so if your system is attached with multiple screens then make sure application run on primary screen. The movie can be exported in .fbr files and can view by HP Micro Recorder.

System monitor enables us to view the system counters in a line graph. System Counters List contains the disk, memory, CPU and other performance indicators related counters.

Capture Data display images of application for the highlighted step, API or Service test information, bitmap checkpoint image, a file content checkpoint comparison or other data.

Log tracking pane displays a log messages that UFT received from application during the test-run. Log writes the following severity level TRACE, DEBUG, INFO, WARN, ERROR and FATAL. We can enable logging in test by navigating to File | Setting | Log Tracking.

## Reporter Object

We can manage report by using Reporter object.

Exercise to understand all the Methods:



*Navigate to "Reports\_in\_UFT" test in UFT | Modify as given below | Save | Run | View | Last Run Results | ALT + V + X |*



```

1 SystemUtil.CloseProcessByName "iexplore.exe"
2 SystemUtil.Run "iexplore.exe","google.co.uk"
3 ' You cannot set path , but only get it.
4 msgbox "My Report path is : " & Reporter.ReportPath
5
6 If Browser("Google").Page("Google").WebEdit("q").Exist Then
7 Reporter.ReportEvent micPass, "Google Search", "Google Search button found."
8 Else
9 Reporter.ReportEvent micFail, "Google Search", "Google Search button Not found."
10 End If
11
12 If Reporter.RunStatus =micFail Then
13     ExitAction
14 End If
15
16 Browser("Google").Page("Google").WebEdit("q").Set "UFT"
17 Browser("Google").Page("Google").WebEdit("q").Submit
18 Reporter.ReportNote "Going to click on Image link"
19 Browser("Google").Page("UFT - Google Search").Link("Images").Click
20
21 CurrentMode = Reporter.Filter
22 If CurrentMode =rfDisableAll Then
23     Reporter.Filter = rfEnableAll
24 Else
25 Reporter.Filter = rfEnableErrorsOnly
26 End If
27 Browser("Google").Page("uft - Google Search_2").Sync
28 Browser("Google").CloseAllTabs

```

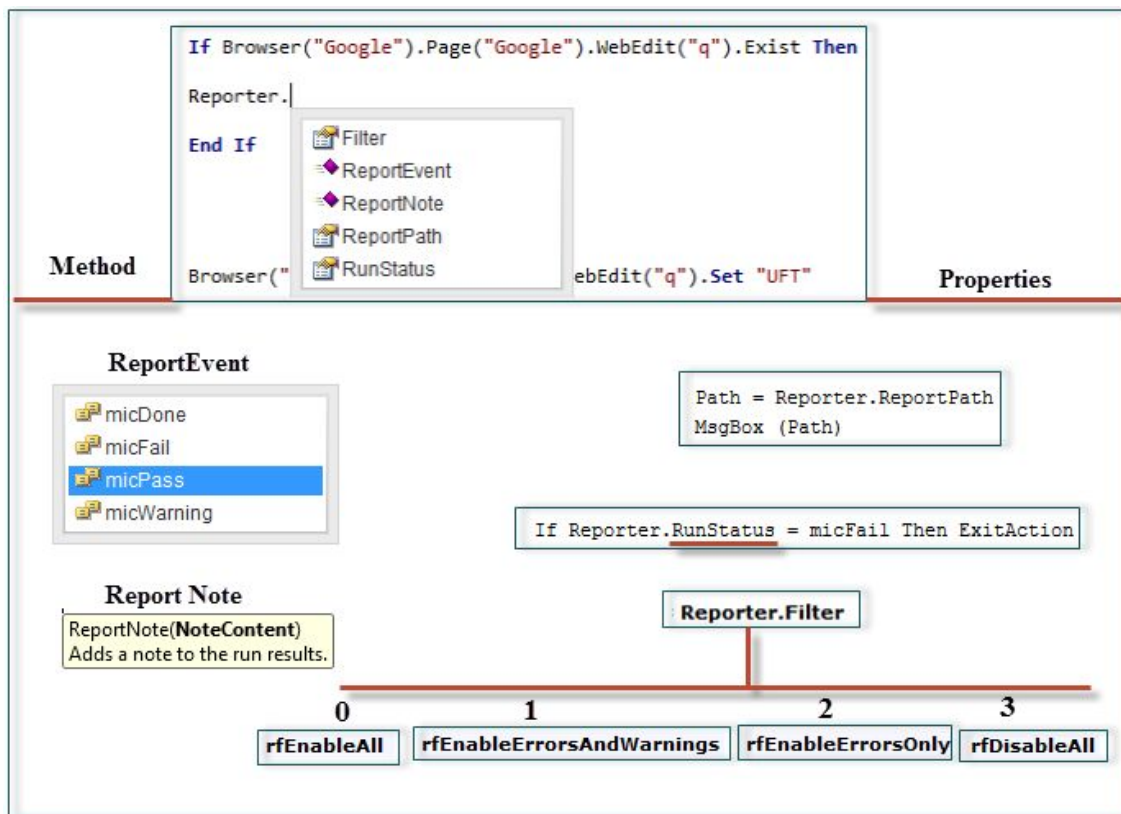


*Navigate to “Reports\_in\_UFT” test in UFT | Run | Close the browser just after it opens to induce the error | Keep wait for script to conclude | View | Last Run Results | ALT + V + X*

## Concept:

Reporter object supports following methods and properties:

Method	Description	Property	Description
ReportEvent	Report event to result viewer	Filter	Filter result based on status
ReportNote	Note to result viewer	ReportPath	Return the report path
		RunStatus	Returns run-status



ReportEvent reports an event to the run results. UFT reports event for all steps in script but we can also write this statement whenever we need to report any event in report. This is the specification of ReportEvent:

Reporter.ReportEvent EventStatus, ReportStepName, Details, [ImageFilePath]

Where EventStatus - micPass, micFail, micDone and micWarning

ReporterNote sends note to report and it can be seen in Executive Summary Notes section of the Result Details pane.

We can filter the result by setting the filter property to rfEnableAll (Default), rfEnableErrorsAndWarnings, rfEnableErrorsOnly and rfDisableAll. ResultPath is read only property and cannot be used to set the path of result. Reporter.RunStatus returns the status of the test.

