**Introduction to HP UFT**

|  |
| --- |
| **Version History** |
| **a)** **Astra QuickTest**  (1.0 to 5.0) – 2000 |
| **b)** **QTP** 5.6 -2002  6.5 -2003  8.0 - 2004  8.2-2005  9.0-2006  9.2 - 2007 --------Mercury Interactive)  9.5 -2008 ---------HP---------  10.0-2009  11.0-2010 |
| **c) UFT**  11.5 -2012  12.02 -2014  12.50 -2015  12.54 -2016  13.00 –never released    14.0 – 2017 |
| ---------Miicro focus------ |

|  |
| --- |
| **UFT Product Information** |
| Unified Functional Testing = QTP + Service Tools |
| UFT is a Functional and Regression Test Tool, supports Desktop and Web Applications |
| UFT developed in .NET technology |
| UFT is an object based Test tool  Software objects in Desktop / Windows based applications  Window, Dialog box, Edit box, Button, Radio button, Check box, Drop down box, List box, Combo box etc...  Software objects in Web based applications  Button, Radio Button, Edit box, List box etc...  Browser, Page, Link, Image etc..... |
| UFT has an integrated MS Access Database engine to support Database operations. |
| UFT has vbscript engine to apply programming logic to Tests  Work with flat files  Work with Excel files  Work with Database files |
| UFT is I-tier Application |

|  |
| --- |
| **UFT Supporting Environments** |
| **a) OS Environment**  MS Windows   Case Study :  Client side Market: Windows: 85% UNIX: 10%  server side Market: Windows: 45% UNIX: 45%  Selenium Windows, UNIX, Mac etc... |
| **b) Application Environment** > GUI based > Windows / Desktop > Web based  Selenium -Web applications only |
| **c) Scripting / Programming** > UFT VBScript  Selenium - Java/C#/Perl/Python/Ruby/PHP |
|  |
|  |

|  |
| --- |
| **UFT Add in (Environment compatible files)** |
| **a) Internal add ins**  1) Standard Windows (Built Add in) – By default loads so does not show in add-in window to choose.      2) Activex     3) Visual Basic     4) Web |
| **b) External Add ins**  5) Java  6) .NET for windows forms  7) .NET for web forms  8) WPF(windows presentation foundation)  9) SAP(to develop **E**nterprise **R**esource **P**lanning applications) for GUI  10) SAP(to develop **E**nterprise **R**esource **P**lanning applications) for web  11) Peoplesoft(to develop **E**nterprise **R**esource **P**lanning applications)  12) Siebel  13) Power Builder  14) Delphi  15) Web services  16) TE (Terminal Emulator for Mainframes-for ERP applications )  17) Oracle(not for Oracle DB but forms and Other oracle technologies)  18) Smalltalk(to support Ops concepts)  19) Stingray ()  20) Silver Light (MS technology for video streaming - outdated) |
| How to select appropriate Add ins for AUT: ------------------------------------------------- 3 web applications, developed using Java technology  1st App(front end controls are html , java used for internal architecture) - Web  2nd App (front end controls are html and ActiveX, java used for internal architecture) – Web , Activex  3rd App (front end controls are html, ActiveX and Java) – Web, Activex, Java  --------------------------------------------- Based on UI design. |
|  |

|  |
| --- |
| **UFT License** |
| a) Seat / Node Locked – fixed to machine, good when same machine used by multiple users. |
| b) Concurrent / Float - fixed to number of users access at a time , good when nearly each user have their own machine to use, so install on 50 machine , but if only 10 users use UFT at a time . Then only 10 licences are sufficient. |

|  |
| --- |
| **UFT IDE (Integrated development Environment)** |
| * UFT support object identification, vb does not * Eclipse does not support object identification, it’s Selenium webdriver’s , Eclipse only helps to write code in Java. * That’s the reason you can run Selenium webdriver test cases without Eclipse , but you cannot run UFT test cases without UFT(or it’s run time engine). |
| 3 types of features support  i) Direct features ii) Integrated tools from HP iii) Integrated tools from others |
| i) Direct features  Ex: Recording, Checkpoints, Output values, Data Table, Object Spy, Step Generator, Object Repository etc... |
| ii) Integrated tools from HP  Ex: Password encoder, Test Batch Runner, Test Result deletion tool etc... |
| iii) Integrated tools from others  Ex: MS Access Database for Database operations VBScript engine for applying programming logic MS Script Debugger for Debugging Tests |

|  |
| --- |
| **Limitations of UFT** |
| a) It doesn't support UNIX/Linux/Solaris operating environments |
| ~~b) It supports IE only for creating test tests~~ |
| c) Supports VBScript only for enhancing tests |
| d) Challenges in Object Identification |

**UFT Test Process**

|  |
| --- |
| **SDLC Manual Phases** |
| Test Planning  Task  Requirement gathering.  Artifact  Requirement document  Test Plan document |
| Test Design  Task  Scenario identification  Test case identification  Test data creation  Artifact  Test Case |
| Test Execution  Task  Test case execution  Defect logging  Defect follow-up  Artifact  Test Execution results  Defect Report |
| Test Closure  Task  Closure preparation  Artifact  Closure document |

|  |
| --- |
| **UFT Test Process Phases** |
| Planning Phase |
| Generating Basic Tests |
| Enhancing Tests |
| Running and Debugging (Optional) Tests |
| Analysing Test Results |
| Reporting Defects |

|  |
| --- |
| **Planning Phase** |
| * Get Environment Details from Development team (UI design technology and Database) and select appropriate Add in for our AUT      UI design - for selecting Add ins      Database - to create Database Connection string |
| * UFT Deployment / Installation |
| * Analysing the AUT in terms Object Identification |
| * Select Tests for Automation      Tests that can be executed on Every build (Sanity Tests)      Tests that can be executed on Every Modified build (Regression Tests)     Tests that can be executed using multiple sets of Test Data(Data Driven Tests) |
| * Tool settings Configuration and Globalize     Object Identification Configuration     Test settings Configuration     Tool options Configuration etc. |
| * Automation Framework Implementation (Optional) |

|  |
| --- |
| **Generating Basic Tests** |
| a) Object Repository based Test Design      1) Recording     2) Keyword driven methodology |
| b) Descriptive Programming / Programmatic descriptions      1) Static programming     2) Dynamic Programming |
| c) Hybrid (OR and DP) |
| **Prerequisites for Generating Tests**  **i) Test Scenario: (For Recoding / Keyword driven)**  Ex:  a) Login Functionality in Flight Reservation Application  b) Login Functionality in Gmail Application  **ii) Steps / Navigation : (For Recoding / Keyword driven)**  a) Login Functionality in Flight Reservation Application                  1) Launch the Application                  2) Select the Login Dialog box                  3) Enter Agent Name                  4) Enter Password                  5) Click OK Button  b) Login Functionality in Gmail Application                  1) Launch the Browser and Navigate to Gmail home page (www.gmail.com)                  2) Enter Email                  3) Enter Password                  4) Click Sign in Button  **iii) Verification points: (For Recoding / Keyword driven)**  a) Login Functionality in Flight Reservation Application  Check the existence of Flight Reservation window, if exists then pass otherwise fail.  b) Login Functionality in Gmail Application  Check the existence of Email page, if exists then pass otherwise fail.  **iv) Error Handling: (For Recoding / Keyword driven)**  a) Login Functionality in Flight Reservation Application  Close the Error Popup and Cancel the Login Dialog.  b) Login Functionality in Gmail Application  Read Error message and close the Browser window.  **v) Input Data: (For Recoding / Keyword driven)**  a) Login Functionality in Flight Reservation Application  Agent Name=  Password=  b) Login Functionality in Gmail Application  Email=  Password=  **vi) Add Comments: (For Recoding / Keyword driven)**  Note: These are common for all types of test design.  vii) Test Objects information  viii) Methods/Operations Information  **vii) Test Object information: (For Keyword driven)**  **viii) Method/Operations on Test-Object information: (For Keyword driven)** |

|  |
| --- |
| **Prerequisites for Generating Tests** |
| a) Object Repository based Test Design      1) Recording     2) Keyword driven methodology |
| b) Descriptive Programming / Programmatic descriptions      1) Static programming     2) Dynamic Programming |
| c) Hybrid (OR and DP) |

|  |
| --- |
| **Enhancing Tests** |
| * Try to use VBScript features instead of Tool features:   a) VBScript features are faster in execution  b) VBScript features won't corrupt  c) Script features have no limitations  d) All VBScript features can be used internally as well as externally |
| * Inserting Checkpoints (9 + 3) (UFT) |
| * Inserting Output values (UFT) |
| * Inserting Transaction Points (UFT) (Start and End) / Timer Function (VBScript) – Performance testing feature. |
| * Inserting Flow control statements (VBScript)   a) Conditional statements (alternative of Checkpoint, also use for error handling)             i) If statement             ii) Select (Switch in Java) Case statement   b) Loop statements(for parameterization)             i) For...Next             ii) Do While / Until...Loop             iii) While...Wend             iv) For Each...Next (For Enhanced loop) |
| * Generating steps using Step Generator (UFT)   Using Step generator we can generate script for  Recordable as well as non-recordable steps (operation on disabled button etc.) and Irregular operations. |
| * Synchronization (UFT, VBS) – matching the speed of UFT an AUT, synchronization default 20 sec. |
| * Parameterization (UFT, VBS) – replace hard code values with variable |
| * Using Environment variables for centralized maintenance of Variables for multiple test cases. Tests portability |
| * Adding comments |
| * Calling Actions (UFT), Functions (VBScript) |
| * Using Regular Expressions         To handle dynamic objects         For search operations |
| * Using Automation Objects (VBScript) |
| * Error Handling (VBScript, UFT) |
| * **Using Automation Objects (VBScript)**  a) File System Object It is used to work with Drives, Folders and files.  i) What is Computer File system?     It is a feature of Operating system used to work with Drives, Folders and files.  ii) Examples for File system Operations Create a Folder Copy a Folder Delete a Folder Create a Text file Write data Read data Compare data Search operations Delete a text file etc...  iii) How end user performs File system Operations?     End user performs file system operations manually with the help of Input devices     If it is command line OS (Ex: DOS, UNIX) with help of OS Commands  iv) How to perform automatic File system Operations using VbScript?  Using File system object   v) Syntax for Creating Automation Object: Set Variable = CreateObject("Class Value") Set -VBScript statement CreateObject - Built in function of VBScript  vi) Create File System Object Set Variable = CreateObject("Scripting.FileSystemObject")  Note: It is only for flat files, but we can create and delete other types files also. File system object treats them as flat files only.  vii) Excel Application Object It is used to perform operations on Excel Application  Create Excel file Row count Read data  Write data Compare data Rename sheets Add sheets Move sheets etc...  viii) Create Excel Application Object  Set Variable = CreateObject("Excel.Application")  ix) Word Application object It is used to perform operations on word application  x) Create Word Application Object  Set Variable = CreateObject("Word.Application")  xi) Database Objects  1) Database Connection object It is used to connect to a Database (Any Database)  Note: Connection string varies from one database to another.  Create Database Connection object:      Set Variable = CreateObject("Adodb.Connection")  Note: Adodb means = ActiveX Data Object Data Base  2) Database Recordset object  It is used to perform operations on Database records (Tables)  Create Database Recordset object:      Set Variable = CreateObject("Adodb.Recordset")  xii) Dictionary object  It is used to define key, value pairs  Dim a(3) a(0) = 10 a(1) = 20  Create Dictionary Object     Set Variable = CreateObject("Scripting.Dictionary") |
|  |

|  |
| --- |
| **Error Handling** |
| Handling expected and unexpected errors |
| Expected error:      Whenever we use invalid input, then expected errors Note: We use Invalid input for negative testing |
| Unexpected Error:      Availability of Resource     Insufficient Resource     Resource Response |

|  |
| --- |
| **Running (Executing) and Debugging Tests** |
| * Debugging is optional but when it is required?      Test is not showing any errors and providing correct output - Not required     Test is showing errors -Optional     Test is not showing any errors and Not providing correct output -Yes * Using VBScript Debug commands and Breakpoints we can debug tests. |
| * Test Run * Single Test Run - Using Run Command * Batch Testing     Using "Test Batch Runner Tool"     Using AOM Script (QTP Application Object) AOM-Automation Object Model     Using Driver Script (UFT Test)     Using ALM/QC Tool |

|  |
| --- |
| **Analysing Test Results** |
| Test Result for every Test Iteration |
| Status of Test Result:      Pass (If expected = Actual)     Fail (If expected <> Actual)     Done (Test executed without errors)         (\* There are no verification in our test then done status)     Warning (Problems during Test Execution |
| Define Test Results  UFT Provides Test result for every test iteration if we use UFT Tool features like checkpoints, output values and transaction points etc...  If user uses own logic (VBScript features) the user need to define test result. |

|  |
| --- |
| **Reporting Defects** |
| Functional Test Automation        Defect Management ------------------------------------------------------------ UFT                    Manual (Excel sheet...) --------------------------------------------------------- UFT                    Bugzilla, Jira etc... -------------------------------------------------------------- UFT                    ALM (Application Lifecycle Management) |
|  |

**UFT Tool window Architecture**

|  |
| --- |
| **Add in Manager** |
| During launching UFT shows Add in Manager,  It shows all available Add ins in our company, Select appropriate add in for our Application,  Note: Do not select unnecessary Add ins, then Tool execution performance will be reduced. |
| Launch UFT Tool first then AUT,  Note: If we launch AUT first, then UFT may not recognize Application objects. |

|  |
| --- |
| **UFT Tool Editor** |
| It is an area where we Create/View/Edit a Test or Test Script. |
| Using UFT Tool editor we can delete one or more statements.  Note: UFT Test is a file, you can delete UFT test file from Hard disk. |
| It has 2 views  Editor view - Test in VBScript format  Keyword view - Test in GUI format  Views are different, but Test is same.  If we perform any modification in one view that automatically reflects in another view. |
| Switch from Editor to Keyword view:  View menu -> Keyword view  Switch from Keyword view to Editor:  View menu-> Editor |
| UFT Test / Test Script  Test - One or more Actions to perform a task or tasks  Action - Set of statements to perform a task or tasks  Statement / Step / Instruction - a minimal executable unit  Note: One test may have one or more Actions, if it is simple test use single Action for lengthy test use multiple Actions.  For re-usability we prefer multiple Actions. |
| Statement Vs Line  One statement may have one or more lines and vice versa.  Two statements in single line  Msgbox "Hello UFT" : Msgbox "Hello VBScript"  One Statement in Two Lines  Msgbox \_  "Hello UFT" |

|  |
| --- |
| **Active Screen** |
| Launch Active Screen  View menu-> Active Screen  It captures and holds screen shot for every user operation on AUT. |
| Advantage:  Using Active screen we can easily understand and Edit Tests |
| Disadvantage:  It occupies more memory, reduces Tool execution performance. |
| Configure Active Screen:  Tools menu-> Options -> GUI Testing ->Active Screen-> Increase/Decrease Capture Level->Ok  Note: This feature is related to Recording only. |

|  |
| --- |
| **Data Table** |
| It is an integrated spread sheet for Data Related operations |
| Using Data Table we can work with Flat files, Excel files and Database files. Or Using VBScript Automation Objects we can work with Flat files, Excel files and Database files. |
| Navigation: View menu-> Data |
| It has 2 types of sheets  a) Global sheet (For Entire Test)  b) Action Sheets / Local sheets (for specific Actions) |
| Data Table located in 2 areas  i) In UFT Main window (Design time Data Table)  ii) In UFT Result window (Run-time Data Table) |
| Usage of Data Table:  i) Enter Test data into Data Table and connect to the Test  ii) Import Test data from external files (Text/Excel) and connect to the Test  iii) Import test data from a Database and connect to the test  ---------  iv) Using Data Table methods and Programmatic statements  Note: We insert Programmatic statements in order to overcome some limitations of Tool features. |

|  |
| --- |
| **Errors** |
| It shows syntax errors automatically while saving the Test.  View -> menu |
| UFT Commands  In 3 ways  a) As Menu items  b) As Tool bar items  c) As shortcut keys. |
| Disadvantage:  It occupies more memory, reduces Tool execution performance. |
| Configure Active Screen:  Tools menu-> Options -> GUI Testing ->Active Screen-> Increase/Decrease Capture Level->Ok  Note: This feature is related to Recording only. |

|  |
| --- |
| **UFT Menus** |
| UFT Commands  In 3 ways  a) As Menu items  b) As Tool bar items  c) As shortcut keys. |
| 1 File Menu  Create New Test  Open Existing Test  Add Test (Using this command we can open multiple tests at a time in UFT Tool editor)  Save Test  Close Test  Settings  Associate Resource files  Associate Function library files  Associate Recovery Scenario files  Associate Environment Variables files  Export to Test to Zip  Import Test from Zip  Exit |
| 2 Edit Menu  Redo  Undo  Cut  Copy  Paste  Delete  Format  Code snippet (To generate VBScript conditional and Loop statements syntax) |
| 3) View menu  Data Table  Active Screen  Errors  Debug  Test Flow  Switch from Editor View to Keyword view and vice versa  Launch Result window |
| 4) Search menu:  Find  Find next  Replace  Incremental search  Goto |
| 5) Design Menu:  Create New Action  Call to existing Action  Copy Action  Insert checkpoints  Insert Output values  Insert Transaction points  Insert Synchronization point  Launch Step Generator  Function Definition Generator etc... |
| 6) Record Menu  Record (Normal Recording)  Stop  Analog Recording  Low Level Recording  Insight Recording  Record and Settings Etc... |
| 7) Run Menu  Run  Pause  Stop  Compile  Run modes  Step into  Step over  Step Out  Insert / Remove Breakpoints  etc... |
| 8) Resources menu  Launch Object Repository  Edit Local Objects  Create Shared Repository  Edit Shared objects  Associate shared repositories  Create New Recovery scenario  Edit existing Recovery scenario etc... |
| 9) ALM  ALM Connection  Checkout  Checkin  Version History  Baseline History etc... |
| 10) Tools Menu  Object Spy  Object Identification  Virtual Objects  Data Driver  Regular expression evaluator  View Options  Tool options  Etc... |
| 11) Window Menu  Next window  Previous window  Close all documents |
| 12) Help menu  UFT Tool features documentation  VBScript documentation  New features  Knowledge forums  Updates  Technical Support  Product info  License info  etc... |

**Record and Run Tests**

|  |
| --- |
| **What is Test Recording?** |
| It is process of generating steps for every user action AUT and storing objects information into Object Repository. |
| Steps for Recording:  Select Record:  Record menu-> Record  Or  Select Record command from the Tool bar  Or  User F6 shortcut key  -> Click OK  -> Navigate the Application  -> Stop Recording |
| After Recording  i) UFT generates steps / statements for all user actions on AUT  ii) UFT stores Objects information in to Object Repository. |
| Recording Modes  Normal Recording  Analog Recording Low Level Recording Insight Recording |

|  |
| --- |
| **Normal Recording Mode** |
| It records user mouse and keyboard operations based on objects Or It records user context sensitive operations based on objects |
| It is unable to record user continuous mouse operations. |
| It is the default mode of UFT  Note: Without selecting Normal recording mode we can't select other recording modes (Analog Recording, Low Level Recording, Insight Recording). |
|  |

|  |
| --- |
| **Analog Recording Mode** |
| It records user Actions on AUT based on x, y coordinates (Desktop or Application window).  It records all user actions in a track file |
| It can record normal operations also, but we can't use this mode for normal operations. |
| **Drawbacks:**  i) It occupies more memory than Normal recording then UFT execution performance will be reduced.  ii) It doesn’t generate steps for every user action on AUT, it records all user actions in a track file, so Test is not readable as well as editable. |
| **Requirement:**  In our test some actions are normal and one or two continuous mouse operations.  =====================================  Select Normal and Analog Record modes. |

|  |
| --- |
| **Low level Recording Mode** |
| It records some operations on Non-supported environments  It considers all objects in two categories only (Window, Win Object) |
| **Drawbacks:**  i) It occupies more memory than Normal recording then UFT execution performance will be reduced.  ii) It considers all objects in two categories only (Window, WinObject), so Test is not readable. |

|  |
| --- |
| **Insight Recording Mode** |
| It records some operations on Non-supported environments and capture object images also. |
| **Drawbacks:**  i) It occupies more memory than Normal recording then UFT execution performance will be reduced.  ii) Test is not readable. |

|  |
| --- |
| **Types of Objects for UFT** |
| i) Run-time objects  ii) Test Objects  iii) Utility Objects  iv) Automation Objects |

|  |
| --- |
| **Run-time objects** |
| The objects present in the Application. |
| Software objects have different states.  States of Software Objects:  Enabled,  Disabled,  Visible,  Hidden,  Focused |

|  |
| --- |
| **Test Objects** |
| Reference of Run-time object is Called Test Object |
| Test object names vary from one environment to another.  Ex:  Button in Standard windows environment -WinButton  Button in VB environment -VbButton  Button in Web environment -WebButton  ---------------------------  Browser - Browser  Page-Page  Link-Link |
| How to get Test object names:  Using Object Spy  Tools Menu-> Object Spy |
| How to get all test objects names in an environment  Using Object Identification  Tools menu-> Object Identification-> Select Environment |
| Three types of test/run time objects:  a) Constant Objects (Properties values are fixed)  b) Dynamic Object (Properties values change dynamically throughout the execution)  Using Regular expressions we can handle Dynamic objects.  c) Duplicate Objects (Two or more objects with same properties)  Using index property we can handle duplicate objects. |

|  |
| --- |
| **Utility Objects** |
| They are UFT reserved objects used for Testing and Result reporting.  Ex:  SystemUtil  RepositoriesCollection  Environment  Services  Reporter  etc... |
| SystemUtil is Utility object  InvokeApplication is Utility statement  Launch Software Applications  a) Windows based Application  Syntax:  SystemUtil.Run "Path of the Application"  Ex:  SystemUtil.Run "C:\Program Files\HP\Unified Functional Testing\samples\flight\app\flight4a.exe"  Or  InvokeApplication "C:\Program Files\HP\Unified Functional Testing\samples\flight\app\flight4a.exe"  b) web based Application  Syntax:  SystemUtil.Run "Path of the Browser", "url"  Ex:  SystemUtil.Run "C:\Program Files\Internet Explorer\iexplore.exe", "www.icicibank.com" |

|  |
| --- |
| **Automation Objects** |
| Automation Objects are used defined objects used to work with drives, Folders, files and Databases. |
| a) File System Object  It is used to work with Drives, Folders and falt files.  Class value -("Scripting.FileSystemObject") |
| b) Dictionary Object  It is used to define Key, value pairs  Class value - ("Scripting.Dictionary") |
| b) Excel Application Object  It is used to perform operations on Excel Application.  Class value ("Excel.Application) |
| c) Word Application Object  It is used to perform operations on Word Application.  Class value ("Word.Application) |
| d) Database Connection object  It is used to connect to a Database  Class value ("Adodb.Connection") |
| e) Database Recordset object  It is used to perform operations Database Tables(Records)  Class value ("Adodb.Recordset") |

|  |
| --- |
| **Object Repository** |
| It is a storage place to store Test objects information. |
| Two types of Object Repository:  a) Local Object Repository (Internal file)  b) Shared Object Repository (External file, .tsr) |

|  |
| --- |
| **Local Object Repository** |
| UFT creates a Local Repository for every Action during Recording. Local Repository files will be saved along with the Test. |
| User (Tester) Can edit(Add, Rename, Delete) Local objects. |

|  |
| --- |
| **Shared Object Repository** |
| User creates shared object repositories either by Adding objects or by Exporting local object. |
| User(Tester) has to create maintain these files |
| 1. **Create Shared OR:**  Open the Dialog box / Window / Web page (AUT)  Select Resources menu -> Object Repository Manager-> Object menu-> Add objects -> Show the Dialog box / Window / Web page (AUT)-> Select the Filter->Click OK   Save the Repository with .tsr extension. |
| 1. **By Exporting Local Objects:**  Launch Local object Repository->File menu-> Export Local Objects->Enter File Name->Create |
| **Types of Filter while adding object**   i) Selected Object Only (UFT will store selected object only, no child objects)  ii) Default objects (All objects which are having functionality)  iii) All Objects (All objects including static objects)  iv) Selected Object types (Selected class of Objects only, but user can select one or more classes) |

|  |
| --- |
| **Operations on Object Repository** |
| **a) Add Objects (Local, Shared)**  **i) Add Objects to Local** Resources menu-> Object Repository->Object-> Add Objects to local-> Show the object->OK  **ii) Add Objects to Shared:** Resources menu-> Object Repository manager->File->Open->Browse path of the file -> File menu->Enable Editing ->Object-> Add Objects->Show the object->OK-Save->Close OR manager. |
| **b) Rename Objects(Local, Shared)**  **i) Rename Local Objects:** Resources -> Object Repository->Select Object & Right click-> Rename->Close Object Repository  **ii) Rename Shared objects:** Resources -> Object Repository Manager->File->Open->Browse path of the file -> File menu->Enable Editing ->Select Object & Right click-> Rename->save->Close OR manager |
| **c) Delete Objects (Local, Shared)**  **i) Delete Local Objects:** Resources -> Object Repository->Select Object & Right click-> Delete->Confirm Deletion-> Close Object Repository  **ii) Delete Shared objects:** Resources -> Object Repository Manager->File->Open->Browse path of the file -> File menu->Enable Editing ->Select Object & Right click-> Delete->Confirm Deletion->save->Close OR manager |
| **d) Export Local Objects**  Resources -> Object Repository->File->Export Local objects->Browse path to store->Enter file name ->Create->Close Object Repository |
| **e) Merge Repositories**  Resources -> Object Repository Manager->Tools->Object Repository merge tool->Browse path of the primary file, Secondary file -> OK->Close->Save the Repository-Close Object Repository merge tool->Close Object Repository Manager |
| **f) Associate Shared Object Repositories**  Test  Action 1    Repository 1 and Repository 2 Action 2    Repository 2 and Repository 3  Navigation- Resources->Associate Repositories-> Click Add sign->Browse path of the Repository-> select Action-> Associate  **Why we need to Associate shared Repositories?**  In order to create and execute Tests manually (Without Recording).  -> If it is Recording UFT creates Local repositories and it uses local object Information while test execution. -> If user wants to create tests manually then create shared object Repositories and Associate. |
| **g) Load Repositories during directly in the Test Script**  **Syntax:** RepositoriesCollection.Add "Path of the shared object Repository file"  **Ex:** RepositoriesCollection.Add "C:\Users\GVenkat\Desktop\Repository1.tsr" |
| **h) Export Test Objects to XML / Import Test Objects from XML**  Advantage:  We can edit shared objects without UFT Tool.  **Export Test Objects to XML:**  Resources -> Object Repository Manager->File->Open->Browser path of the OR file ->File->Export Test Objects to XML->Enter file name->Save  **Import Test Objects from XML:**  Resources -> Object Repository Manager->File->Import Test Objects from XML->Browse path of the XML file |
| **i) Map objects in between OR and AUT:**  How to map: Using View options in Object Repository  i) Highlight in Application (from OR to AUT) ii) Locate in Repository (From AUT to OR) |
| **j) Define New Test Objects**  Using this feature we can create Tests even though Application is not ready. It is not recommendable feature.  How? By getting objects information from development team.  Steps:  i) Get objects information from development team.  ii) Resources->Object->Define New Test Object->Select Environment ->Select Class of Object->Enter Name |

|  |
| --- |
| **Advantages/Disadvantage of Recording** |
| **Advantages of Recording**  a) Easy to generate Tests and It is fast  b) Scripting knowledge is optional  c) It is used to analyse the AUT in terms of object Identification  d) It is recommended for short term projects  e) It is recommended for dynamically changing UI designs |
| **Disadvantages of Recording**  a) No centralized maintenance of Test Objects  b) Low in performance  c) Less or no concentration on complex functionality testing  d) Testers may not have command on Tests so locating errors is difficult  e) All types of test object statements can't be recorded  ex: Irregular operations and Operations on disabled objects. |

|  |
| --- |
| **Types of statements in UFT Test** |
| We use different types of statements in our tests.  i) Declaration Statements  ii) Utility statements  iii) Test Object Statements  iv) Conditional statements  v) Loop Statements  vi) Automation object statements  vii) Checkpoint statements  viii) Output value statement  ix) VBScript Statements  x) Function Calls, Action Calls |
| **i) Declaration Statements**  a) Variables  Ex:  Dim a, b(3), c(), d(4, 5)  b) Constants  Const city="London", num=100, x = #10/10/2010# |
| **ii) Utility statements**  Ex:  SystemUtil.Run "C:\Program Files\HP\Unified FunctionalTesting\samples\flight\app\flight4a.exe"  SystemUtil.Run "C:\Program Files\Internet Explorer\iexplore.exe", "www.gmail.com"  RepositoriesCollection.Add "D:\Login.tsr" |
| **iii) Test Object Statements**  Syntax:  TestObject("ObjectName").Method /Operation  Or  TestObject("ObjectName") … .Child TestObject("Objectname").Method /Operation  Ex:  Dialog("Login").Activate  Browser("Gmail").Close  Dialog("Login").WinButton("Cancel").Click  Browser("Google").Page("Google").Sync  Window("Flight Reservation").Dialog("Open Order").WinButton("Cancel").Click  Browser("Gmail").Page("Gmail").Link("Create an account").Click |
| **iv) Automation object statements**  Ex:  Set objFso = CreateObject("Scripting.FileSystemObject")  objFso.CreateFolder "C:\Users\G C Reddy\Desktop\UFT" |
| **v) Conditional statements**  Ex:  Dim a, b  a = 100  b = 50  If a > b Then  Msgbox "A is a Big Number"  Else  Msgbox "B is a Big Number"  End If  Note: We use conditional statements to insert verification points and for Error handling. |
| **vi) Loop Statements**  Ex:  For i = 1 To 5 Step 1                  Msgbox i & " Hello UFT"  Next |
| **vii) Function Calls, Action Calls**  Ex:  Call Login("abcd", "mercury")  RunAction "Login [GUITest4]", oneIteration |
| **viii) VBScript Statements**  ex:  Dim  With  set  For  Option Explicit  On Error Resume Next etc...  Ex:  Set x = Dialog("Login")  x.Activate  x.WinEdit("Agent Name:").Set "abcd"  x.WinEdit("Password:").SetSecure "54ebda121556ea2ddf1d0c77a6ddd6839ca748ee" x.WinButton("OK").Click |
| **ix) Checkpoint statements**  Window("Flight Reservation").WinEdit("Tickets:").Check CheckPoint("Tickets:") |
| **x) Output value statement**  Window("Flight Reservation").WinEdit("Name:").Output CheckPoint("Name:") |

|  |
| --- |
| **What is Step Generator?** |
| It is a Library of functions and Utility objects used to generate recordable and Non-recordable steps. |
| Navigation:  Design menu->Step Generator  Or Shortcut key F7 |

|  |
| --- |
| **Keyword Driven Methodology** |
| Generating Tests manually (Without Recording) using Keywords(Objects, Methods, Functions etc...) |
| Steps:  **i) Create Shared Object Repositories**  **ii) Associate Shared Object Repositories** Or Load Repositories during directly in the Test Script  **iii) Generate Steps / Statements**      a) Using Editor View      Or     b) Using Keyword view (Easy to generate Test Object statements)     Or     c) Using Step Generator     Or     d) By drag and drop objects from OR to UFT Editor(Test Object Statements only) |

|  |
| --- |
| **Advantages of Keyword Driven Methodology** |
|  |

|  |
| --- |
| **Descriptive Programming Or Programmatic Descriptions** |
| Enter / Provide objects information directly into scripts is called Descriptive Programming.  No need to have Object Repositories separately, we provide objects info directly in the Test/Test script/library. |
| **Types of Descriptive Programming**  a) Static Programming  Enter/Provide objects information (Properties and values) directly into statements  or steps is called Descriptive Programming.  b) Dynamic Programming  Creating description objects and generating steps / statements using description objects. |
| Test object statements vary from OR based test to DP |
| **a) Static Programming**  Syntax:  TestObject("Property Name:=Property value").Operation  Or  TestObject( "Property1 Name:=Property1 value","Property2 Name:=Property2 value" ).Operation  Or  TestObject("Property Name:=Property value")…… . ChildObject("Property Name:=Property value").Operation  Ex:  Browser("CreationTime:=0").Close  Dialog("text:=Login").Activate  Dialog("text:=Login", "width:=320", "height:=200").Activate  Dialog("text:=Login").WinButton("text:=Cancel").Click  Browser("CreationTime:=0").Page("title:=Gmail").Link("text:=Create an account").Click  Hybrid:  TestObject(“ObjectName from OR”)…… . ChildObject("Property Name:=Property value").Operation  Ex:  Dialog("Login").WinButton("text:=Cancel").Click  Browser("Gmail").Page("Gmail").Link("text:=Create an account").Click  **Centralized maintenance of Test Objects** Use Constants to replace literal values and place constants in a library file. If any modifications are there, open the Library file, perform changes and save.   * Syntax: Const constantName = Value TestObject(constantName1 , constantName2).Operation * Ex: Const Login = "text:=Login"   Const Agent ="attached text:=Agent Name:",  Const Password ="attached text:=Password:" Const Ok1="text:=OK", Ok2="height:=23", Ok3="width:=60"  Systemutil.Run "C:\Program Files\HP\Unified Functional Testing\samples\flight\app\flight4a.exe" Dialog(Login).Activate  Dialog(Login).WinEdit(Agent).Set "abcd" Dialog(Login).WinEdit(Password).SetSecure"54edcfd7c14cf9a3e6f38219556c9040102f5860" Dialog(Login).WinButton(Ok1, Ok2, Ok3).Click   * Create Library file Launch Notepad(Editor)   Type or Paste the code  Save as .vbs file Associate Library file(We can associate one or more library files)   * Association two levels a) Test Level File menu -> Settings -> Resources ->Click Add symbol -> Browse path of the file -> OK * b) Tool Level File menu -> Settings -> Resources ->Click Add symbol -> Browse path of the file ->**Set As default**-> OK |
| **a) Dynamic Programming**  Step 1: Create Description Objects: (Set Variable = Description.Create)  Step 2: Enter Properties information : (DescriptionObject("Property1Name").Value = Property1 value)  Step 3: Generate steps using description objects: (TestObject(DescriptionObject).Method or Operation)  Syntax:  Set Variable = Description.Create  DescriptionObject("Property1Name").Value = Property1 value  DescriptionObject("Property2Name").Value = Property2 value  TestObject(DescriptionObject).Method or Operation  Ex:  Set Login = Description.Create  Login("text").Value = "Login"  Login("enabled").Value = True  Dialog(Login).Activate  **Centralized maintenance of Test Objects** Put Object information in library file   * Create Description Objects: (Set Variable = Description.Create)   Syntax:  Set Variable = Description.Create    Ex:  Set Login = Description.Create   * Enter Properties information : (DescriptionObject("Property1Name").Value = Property1 value)   Syntax:  DescriptionObject("Property1Name").Value = Property1 value  DescriptionObject("Property2Name").Value = Property2 value  Ex:  Login("text").Value = "Login"  Login("enabled").Value = True   * Create Library file save object information and associate library |
| **Test Execution Process in case of OR based Test:**  During test execution UFT reads statements from UFT tool editor one by one  and gets objects information from Object Repository  based on that information performs actions on AUT.  **Test Execution Process in case of DP:**  During test execution UFT reads statements from UFT tool editor one by one  and objects information also available in the Statements  based on that information performs actions directly on AUT. |
| **How to select appropriate Properties information for recognizing objects?**  If the development team follows UI design standards then use UFT Object Identification Configuration (\*UFT Object Identification configuration Based on UI design standards)  If development team not follows UI design standards for some objects then select one or two properties and perform dry run, if they are suitable then continue otherwise change the properties and perform dry runs. |
| **Handle Duplicate Objects**   * Latest Browser CreationTime - N-1 (10 Browsers opened on desktop, 9) Oldest Browser creation time - 0 Only one Browser opened on Desktop Creation time - 0 * Two links with same name Use index property  Link("text:=News", "index:=1").Click |

|  |
| --- |
| **Advantages of Descriptive Programming** |
|  |

|  |
| --- |
| **Test Methods / Operations** |
| Method Name: **Activate**  Description: It activates a Dialog box or window  Syntax:  Object.Activate  Example:  Dialog("Login").Activate  Window("Flight Reservation").Activate |
| Method Name: **Click**  Description: It clicks an object(Buttons, Links)  Syntax:  Object.Click  Example:  Dialog("Login").WinButton("Cancel").Click  Browser("Google").Page("Google").Link("Gmail").Click |
| Method Name: **Set**  Description:  a) Enter a value into Edit box  Or  b) Check/Uncheck a Check box  Or  c) Select a Radio Button  Syntax:  a) Object.Set "Value/Parameter"  b) Object.Set "ON/OFF"  Example:  Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "ON"  Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "OFF"  Window("Flight Reservation").Dialog("Open Order").WinEdit("Edit").Set 10  Window("Flight Reservation").WinRadioButton("First").Set |
| Method Name: **SetSecure**  Description: It enters encoded value into Password objects  Syntax:  Object.SetSecure "encoded value"  Example:  Browser("Gmail").Page("Gmail").WebEdit("Passwd").SetSecure "54ee7ee879e31f23e955e813a9101ba2843c7f00"  Dialog("Login").WinEdit("Password:").SetSecure "54ee7ef45df3be4e8b2d20eed1cd085c21bc59a46bcb" |
| Method Name: **Select**  Description: It selects an item from a combo box or list box  Syntax:  Object.Select "item"  Or  Object.Select (index)  Example:  Window("Flight Reservation").WinComboBox("Fly From:").Select (0)  Window("Flight Reservation").WinComboBox("Fly To:").Select "London"  Window("Flight Reservation").Dialog("Flights Table").WinList("From").Select "20262   DEN   10:12 AM   LON   05:23 PM   AA     $112.20" |
| Method Name: **Close**  Description: It closes an object(Dialog box, window, Browser window)  Syntax:  Object.Close  Example:  Window("Flight Reservation").Dialog("Open Order").Close  Window("Flight Reservation").Close  Browser("Google").Close |
| Method Name: **Sync**(Only for Web)  Description: It waits for the Browser to complete its current navigation  Syntax:  Object.Sync  Example:  Browser("Gmail").Page("Gmail").Sync |
| Method Name: **Naviagte**(Only for Web)  Description: It opens a specified url in the browser window  Syntax:  Object.Navigate "URL"  Example:  Browser("Gmail").Navigate "www.icicibank.com" |
| Method Name: **GetItemsCount**  Description: It returns items count items from a combo box or list box  Syntax:  Variable = Object.GetItemsCount  Example:  Items\_Count = Window("Flight Reservation").WinComboBox("Fly From:").GetItemsCount  Msgbox Items\_Count |
| Method Name: **GetContent**  Description: It returns content from a combo or list box  Syntax:  Variable = Object.**GetContent**  Example:  Content = Window("Flight Reservation").WinComboBox("Fly From:").GetContent  Msgbox Content |
| Method Name: **GetVisibleText**  Description: It returns text value prom an object  Syntax:  Variable = Object.GetVisibleText  Example:  Price = Window("Flight Reservation").WinEdit("Price:").GetVisibleText()  Button = Window("Flight Reservation").WinButton("Update Order").GetVisibleText()  Msgbox Price  Msgbox Button |
| Method Name: **GetRoProperty**  Description: It returns run time object property value  Syntax:  Variable = Object.GetRoProperty("PropertyName")  Example:  Price = Window("Flight Reservation").WinEdit("Price:").GetROProperty("text")  Msgbox Price  x = Window("Flight Reservation").WinEdit("Price:").GetROProperty("enabled")  Msgbox x  y = Window("Flight Reservation").WinEdit("Price:").GetROProperty("width")  Msgbox y |
| Method Name: **CaptureBitmap**  Description: It captures screen shot during test execution and stores in a specified location.  Syntax:  Object.CaptureBitmap "Path"  Example:  Dialog("Login").CaptureBitmap ("C:\Users\G C Reddy\Desktop\Login.bmp")  ------------------------  Note:  Whenever we want produce screen shots for Not reproducible defects. |

|  |
| --- |
| **Transaction Points** |
| Using Inserting Transaction Points we can measure Test Transaction.  We can measure entire Test transaction Time and Part of the Test Transaction Time.  We can insert multiple transaction points in a Test. |
| Transaction points export the value to UFT Result window only.  We can't display the value locally and it can't be exported to external files. |
| **Services Utility object**  Syntax:  Services.StartTransaction "TransactionName"  Statements  -----------  -----------  -----------  Services.EndTransaction "TransactionName"  Ex:  Services.StartTransaction "Orders"  …  Services.StartTransaction "Login"  …  …  Services.EndTransaction "Login"  …  ...  Services.EndTransaction "Orders" |

|  |
| --- |
| **Timer Function // Alternative of Transaction Points** |
| Timer Function (VBScript Built in Function) using this we can measure Test transaction Time.  It returns time in seconds that have elapsed since 12:00 AM (Midnight) based on local system. |
| Transaction points export the value to UFT Result window only.  We can't display the value locally and it can't be exported to external files. |
| Syntax:  Variable1 = Timer  -----------  -----------  -----------  Variable2 = Timer  Msgbox (Variable2-Variable1)  Ex:  Start\_Transaction = Timer  End\_Transaction = Timer  TransactionTime = End\_Transaction - Start\_Transaction  Msgbox TransactionTime  Reporter.ReportEvent 2, "Res1", "Transaction Time is: "& TransactionTime |

|  |
| --- |
| **Reporter -Utility object** |
| UFT provides Test Result for every Test iteration if we use UFT tool features like Checkpoints, Output values, Transaction points etc...  If we use our own logic then we need to define them in Test Result. |
| Syntax:  Reporter.ReportEvent <Result Status Code>, "<Result step name>", "<User Details>” |
| Result Status Code                  Either micPass or 0 for Pass status                  Either micFail or 1 for Fail Status                  Either micDone or 2 for Done status                  Either micWarning 3 for Warning Status |
| Ex:  Start\_Transaction = Timer  …  End\_Transaction = Timer  TransactionTime = End\_Transaction - Start\_Transaction  Msgbox TransactionTime  Reporter.ReportEvent 2, "Res1", "Transaction Time is: "& TransactionTime |