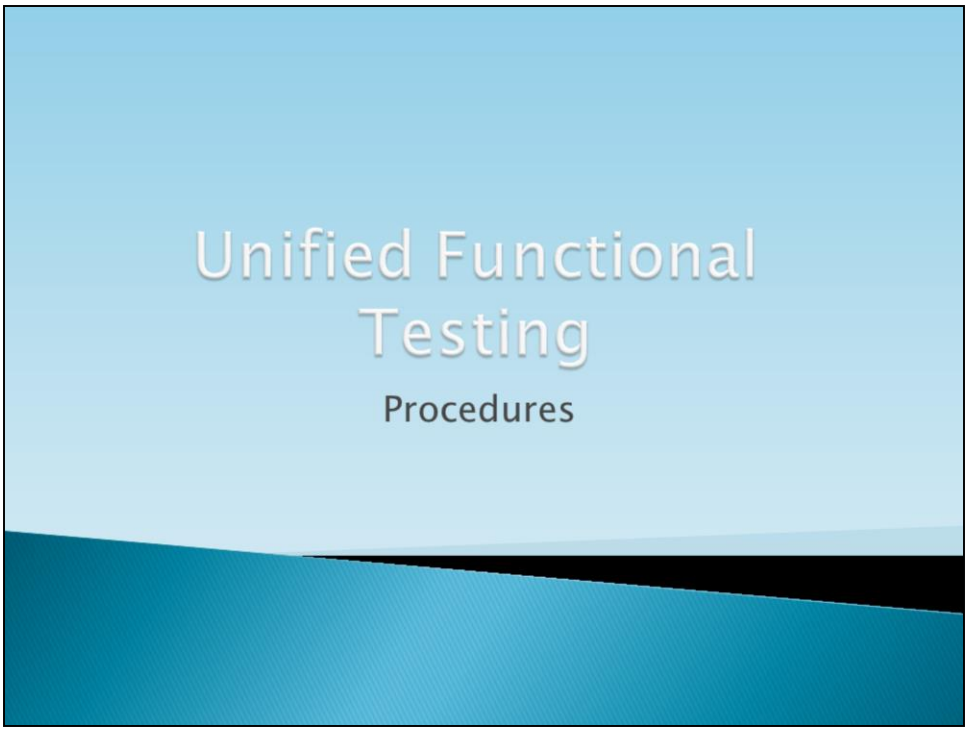

Unified Functional Testing
Procedures

Lesson Objectives

By the end of this Lesson you will be able to:

- Identify the advantages of creating a procedure in a test.
- Create new subroutines and functions.
- Register a procedure with an object class.
- Build and associate a function library.

Topics

1. What is a procedure
2. Functions
3. Subs
4. Register function with test object
5. Create a function library
6. Associate a function library to a test

Procedure – Definition

- Procedure – a series of statements grouped together to perform a specific task.
- Creating procedures in VBScript can:
  - Simplify code for readability and maintenance.
  - Override an existing object method.
  - Enable reuse of abstract code in scripts or recovery scenarios.

Types Of Procedures

- There are two type of procedures:
  - Subroutines – Series of VBScript statements that perform actions but do **not return** a value.
  - Functions – series of VBScript statements that perform actions and **return** a value.

Characteristics	Subroutines	Functions
Contains a series of Statements	V	V
Return a value	X	V
Accept arguments	V	V
Can call other procedures	V	V
Can call itself	V	V

Subroutine – Syntax

- The syntax of the Sub ... End Sub statement is:

```
[Public | Private] SubName (arglist)
   'Group Of Statements
End Sub
```

Note : A subroutine can be marked as public or private. By Default Subroutine assume as Public.

- Example :

```
Sub CleanDBObjects (mCnDB,mRsData)
  If (mRsData) Then
     mRsData.Close
     Set mRsData=Nothing
  End If
  If (mCnDB) Then
     mCnDB.Close
     Set mCnDB=Nothing
  End If
End Sub
```

Function– Syntax

- The syntax of the Function... End Function statement is:

```
Function FunctionName (arglist)
'Group Of Statements
FunctionName=expression
End Function
```

- Example :

```
Function ExecuteQuery(sSQL,mCnDB,mRsData )
  If  (mCnDB.State)Then
      mCnDB.Open
  End If
  ExecuteQuery = mRsData.Execute(sSQL)
End Function
```

Steps To Create New Procedure

- In order to create procedure use the following steps:
  1. Identify series of statements that perform specific task.
  2. Create the procedure declaration.
  3. Define the arguments for the procedure.
  4. Set return values for the procedure.
  5. Handle errors in the procedure.

Identify Series Of Statements

- After we add steps using following methods:
  - Record steps in the application under test.
  - Use the ACTIVE SCREEN.
  - Use the STEP GENERATOR.
  - Type the steps directly in the script.
- Now we need to Identify series of statements that perform specific task(for example , Open Connection , Login to application)

Declaring The Procedure

- After creating code for a procedure ,we need to decide which type of procedure we want to declare , subroutine or function , according to our needs.
- Declare the procedure according to Figures below:

```
Sub SubName (arglist)
'Group Of Statements
End Sub
```

```
Function FunctionName (arglist)
'Group Of Statements
FunctionName=expression
End Function
```

- Add the statements in the relevant place.

Defining an Argument

- After declaring a procedure, identify the values that the procedure will need passed in from the calling script. Type these values in the script as argument variables for the procedure.
- There are two types of argument variables for procedure:
  - **ByVal** keyword – Prevent a procedure from changing the value of the variable passed into it.
  - **ByRef** keyword – Enables a procedure to assign the argument variable to a new value. By default, UFT uses the reference keyword.

```
Function OpenOrderByNum(ByVal OrderNumber)
   Window("Flight Reservation").WinButton("Button").Click
   Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "ON"
   Window("Flight Reservation").Dialog("Open Order").WinEdit("Edit").Set OrderNumber
   Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click
End Function
```

Define Return Values

- After building a new procedure, the procedure might be expected to return a value based on the operation it performs. For example, the open order procedure is expected to return a variable that indicates the order was opened.
- In addition to a return value of the function, functions and subroutines have other output values. Procedures can set any ByRef argument to new values. These new values can be passed back to the line that calls the procedure.

```
Function OpenOrderByNum(ByVal OrderNumber)

    Window("Flight Reservation").WinButton("Button").Click
    Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "ON"
    Window("Flight Reservation").Dialog("Open Order").WinEdit("Edit").Set OrderNumber
    Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click
    OpenOrderByNum = True

End Function
```

Handling Errors

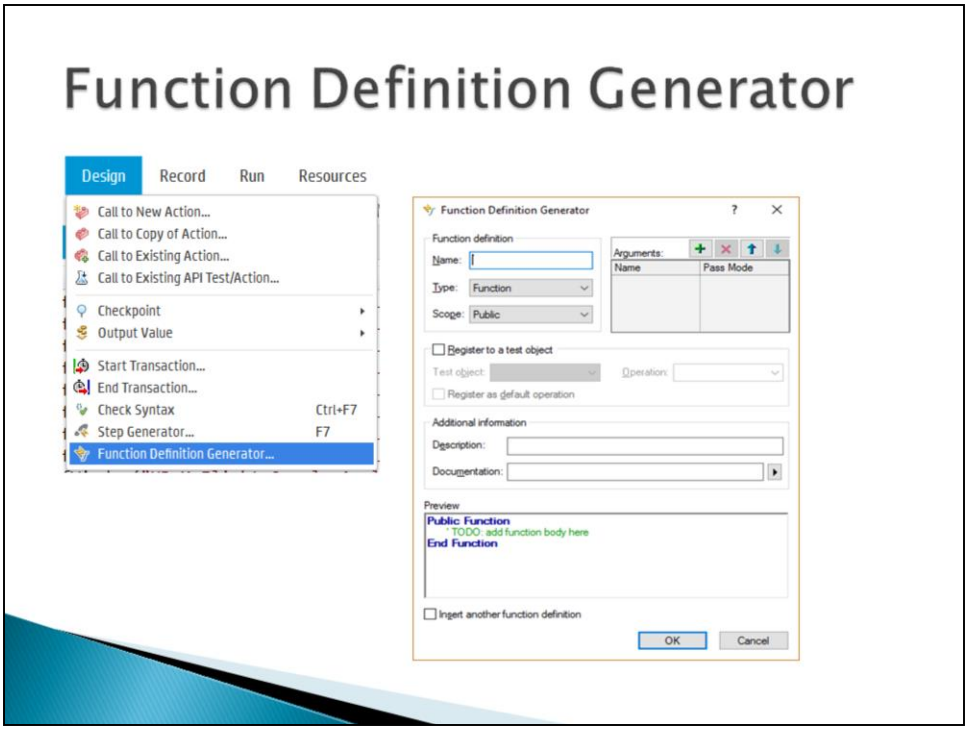
- Use the IF... THEN statement to test the existence of the error dialog box and return a value of False to the calling script.

```
Function OpenOrderByNum(ByVal OrderNumber)

   Window("Flight Reservation").WinButton("Button").Click
   Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").Set "ON"
   Window("Flight Reservation").Dialog("Open Order").WinEdit("Edit").Set OrderNumber
   Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click

   If  Window("Flight Reservation").Dialog("Open Order").Dialog("Flight Reservations").Exist Then
     Window("Flight Reservation").Dialog("Open Order").Dialog("Flight Reservations").WinButton("OK").Click
      Window("Flight Reservation").Dialog("Open Order").Activate
      Window("Flight Reservation").Dialog("Open Order").WinButton("Cancel").Click
      OpenOrderByNum = False
   Else
      OpenOrderByNum = True
   End If

End Function
```



UFT provides a FUNCTION DEFINITION GENERATOR, which enables you to generate definitions for new user-defined functions and add header information to them.

To open the FUNCTION DEFINITION GENERATOR:

1. Ensure that the function library or test where you want to insert the function definition is the active document.

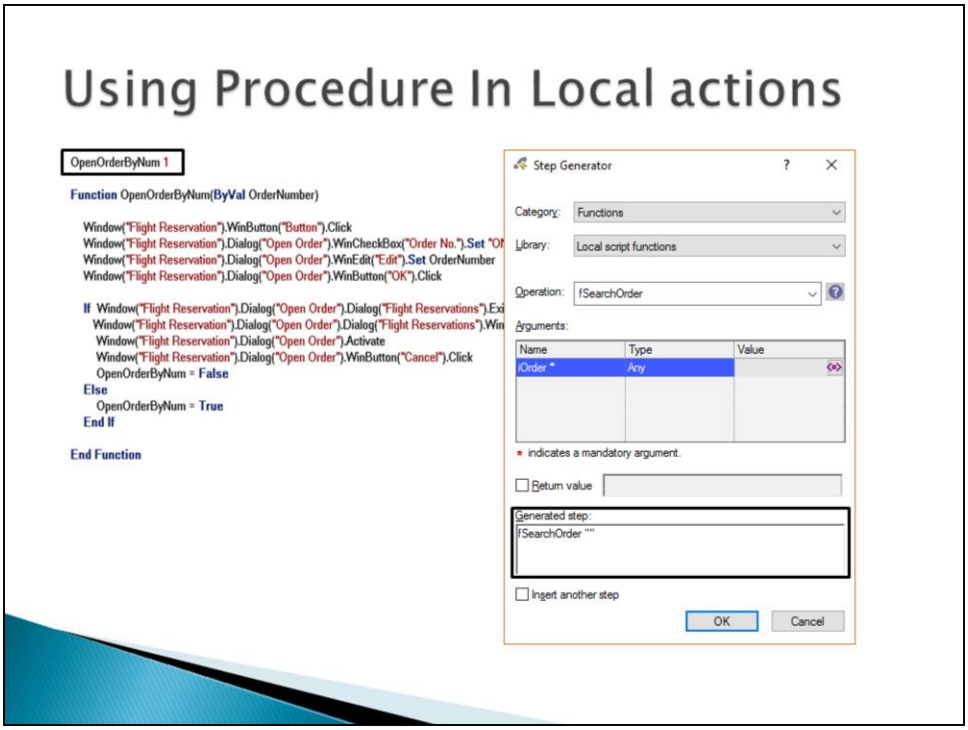
2. Select **DESIGN→FUNCTION DEFINITION GENERATOR** to display the **FUNCTION**

DEFINITION GENERATOR dialog box. In the **FUNCTION DEFINITION GENERATOR** dialog box, fill in the required information.

The **FUNCTION DEFINITION GENERATOR** creates the basic function definition and associates it with the test.

Making A Procedure Available

- Procedures can be made available from:
  - **Local actions**: Procedures that are useful only in the current UFT action or scripted component.
  - **Test object classes**: Procedures that are associated with a particular UFT test object class. These may be new methods added to the class or an override of an existing method.
  - **Library files**: Procedures that are used in multiple UFT scripts by associating library files with that script.



To use a procedure locally in the same script where it is defined, add the procedure definition code to the current script.

After adding the procedure to the script, use the procedure either by typing its name directly into the script or by using **STEP GENERATOR**

Associating A Procedure With A Test Object

- When you create a procedure and associate it with a test object, this association enables you to use the procedure whenever you use an instance of the test object.
- To create a procedure for use as a UFT test object method, perform the following steps:
  - Register the procedure with an object class.
  - Override an existing object method.
  - Unregister the procedure.
- To register a procedure for use in a business component, use a library file to create the procedure and register it in the library file.

Registering The Procedure With An Object Class

- ***RegisterUserFunc*** Statement– associate a procedure with a test object class.
  - **Syntax**
  - *RegisterUserFunc <class>, <method>, <procedure>*
  - **Example**

```
RegisterUserFunc "Image", "Click", "fMyImageClick"

Public function fMyImageClick(objImage)
    Dim sCurrentValue, i
    rc = Setting.WebPackage("ReplayType")
    Setting.WebPackage("ReplayType") = 1
    On error resume next
    reporter.Filter = rfDisableAll
    Do
        i = i + 1
        err.clear
        objImage.Click
        If err.number = 0 Then
            fMyImageClick = True
            On error goto 0
            reporter.Filter = rfEnableAll
            Setting.WebPackage("ReplayType") = rc
            Exit Function
        End If
        If i > 60 Then
            fMyImageClick = False
            reporter.Filter = rfEnableAll
            reporter.ReportEvent micFail, "Click", objImage.tostring & " failed"
            On error goto 0
            Setting.WebPackage("ReplayType") = rc
            ExitRun
        End If
        wait 1
    loop while True
    Setting.WebPackage("ReplayType") = rc
    fMyImageClick = True
End Function
```

After associating a procedure with a test object class, the procedure remains registered with the object class until it is unregistered explicitly or until the next UFT run session begins.

Use the RegisterUserFunc statement to override an existing method of a test object class.

Unregistering A Procedure

- *UnregisterUserFunc* Statement– dissociate any user defined procedures that associate with a test object class.
  - **Syntax**
  - *UnregisterUserFunc <class>, <method>*
  - **Example**

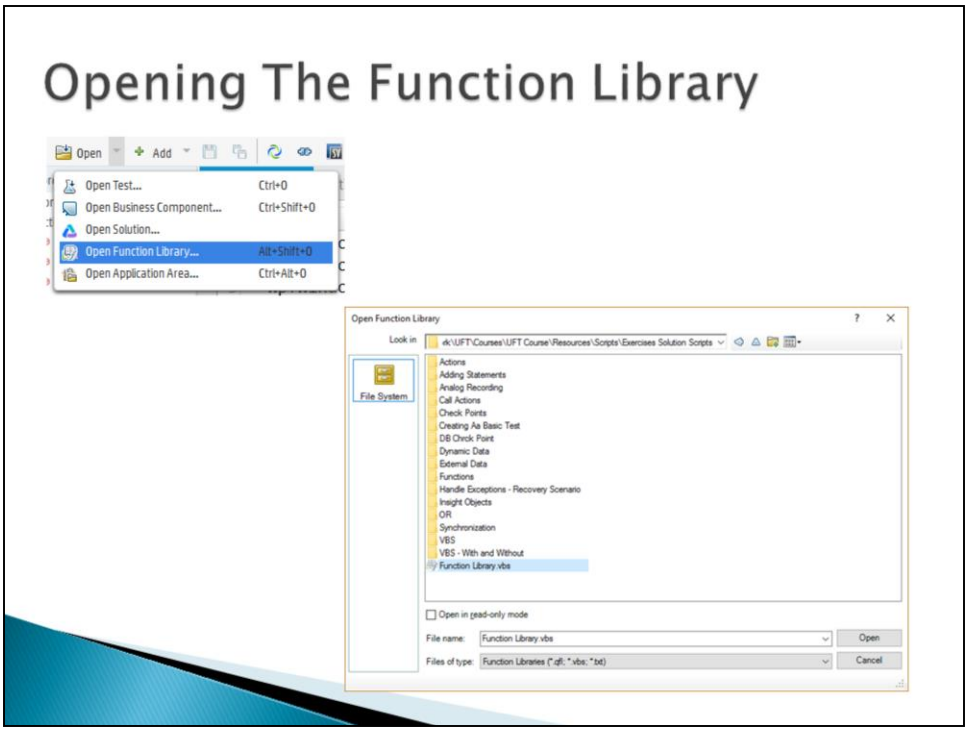
```
RegisterUserFunc "WebEdit", "Set", "fMyEditSet"

Public function fGuiLocateAndSetFieldByLabel(ByRef objTable, sLabelCol, sEditCol, sLabel, sValue)
    Dim i, objEdit
    If JavaWindow("Amdocs Customer Interaction").Page("Amdocs ClarifyCRM Customer")._
        Frame("AttrDetailsFrame").WebTable("Properties").Exist(60) = "False" then
        fGuiLocateAndSetFieldByLabel = False
        Exit function
    End if
    For i = 1 to JavaWindow("Amdocs Customer Interaction").Page("Amdocs ClarifyCRM Customer")._
        Frame("AttrDetailsFrame").WebTable("Properties").RowCount - 1
        If instr(ucase(objTable.GetCellData(i, sLabelCol)), ucase(sLabel)) > 0 then
            set objEdit = objTable.ChildItem(i, sEditCol, "WebEdit", 0)
            wait 1
            'objEdit.Set sValue
            reporter.Filter =rfDisableAll
            objEdit.Click
            reporter.Filter =rfEnableAll
            fSendKeyStroke sValue
            fSendKeyStroke "{TAB}"
            fGuiLocateAndSetFieldByLabel = True
            UnRegisterUserFunc "WebEdit", "Set"
            Exit function
        End If
    Next
End Function
```

You should unregister a procedure in the same context in which the procedure was registered. If a procedure is registered within a UFT action, unregister it in the same action. Similarly, if a procedure is registered within a UFT library file, unregister the procedure in the same library file.

Building a Library file

- You can build library files to make procedures accessible to multiple UFT scripts.
- In order to build a library file:
  1. Create a library file by using one of the following:
     - Standard text editor
     - FUNCTION LIBRARY editor
  2. Associate the library to a test or an application area.

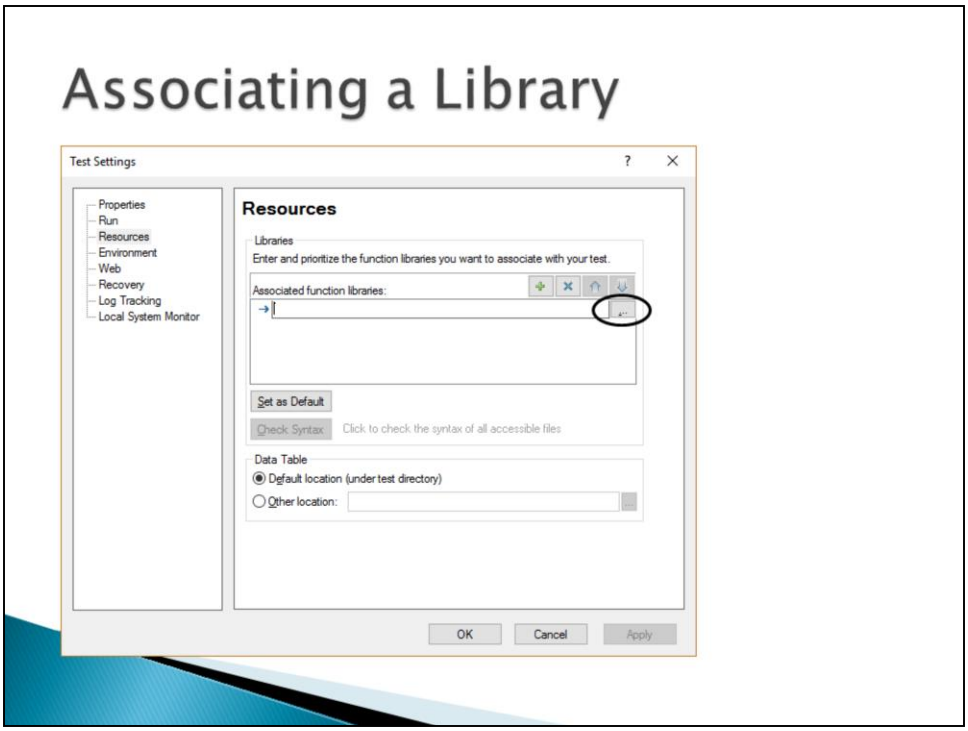


UFT has a built-in **FUNCTION LIBRARY** editor. The **FUNCTION LIBRARY** editor enables you to create and edit function libraries that contain VBScript functions and subroutines.

You can create, open, and work on multiple function libraries simultaneously. Each **FUNCTION LIBRARY** editor opens in its own document window. When working with a **FUNCTION LIBRARY** editor, another testing document, such as a test, application area, or component, is always open. This open testing document enables you to add calls to functions as you create or modify them.

To open a FUNCTION LIBRARY editor, from the UFT menu bar, select **FILE → OPEN → FUNCTION LIBRARY.** The **OPEN FUNCTION LIBRARY** dialog box opens. The **OPEN FUNCTION LIBRARY** dialog box enables you to open the desired FUNCTION LIBRARY editor in UFT.

Note: A function library file is saved with a .qfl extension.



After creating a library, associate the library with the script where you need to use the procedure defined in the library file.

To associate a library to a UFT test:

1. From the UFT menu bar, select **SETTINGS** → **TEST SETTINGS** and open the **RESOURCES** tab.

2. Under LIBRARIES, click the '**+', ADDS A NEW FILE TO THE FILE LIST** button, to associate the desired library files.

3. Use the arrow buttons to edit the order of the libraries.

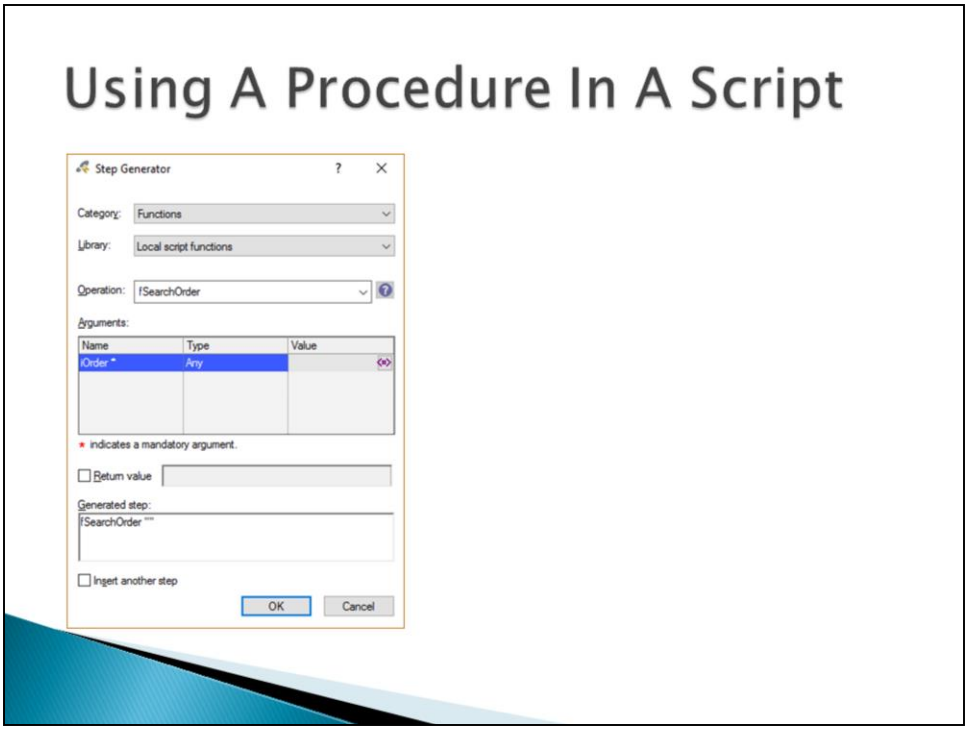
Note: UFT searches for functions in the libraries in the order listed. If a function is defined in multiple libraries, UFT uses the first definition that it finds.

4. Click **CHECK SYNTAX** to review all libraries for syntax errors.

5. Click **SET AS DEFAULT** to automatically associate the selected libraries with your new tests.

Using A Procedure

- You can call a procedure from:
  - Scripts
  - Library Files
  - Recovery Scenarios



The following options are available within the **STEP GENERATOR**:

• New library procedures are available under **LIBRARY FUNCTIONS** in the **FUNCTIONS** category.

• New local procedures are available under **LOCAL SCRIPT FUNCTIONS** in the **FUNCTIONS** category.

• New object methods and the standard object methods are available within the **TEST OBJECTS** category.

What's Next?

- Review Questions
- Next Lesson
  - The next lesson in the course is:
    Custom CheckPoints

End of Lesson