M-eux Test

Test Automation on iOS using Eclipse

Getting Started Guide

Abstract

This Getting Started Guide describes how to install, configure and use M-eux Test for testing iOS applications using Eclipse. This document is intended for Quality Assurance (QA) engineers and testers who wish to get acquainted with the functionalities of M-eux Test.

Part Number MT-GS-IEC
Revision 3408, April 2014

Contents

Chapter 1: Introduction

Welcome to the Test Automation on iOS using Eclipse Getting Started guide. This document will assist you with installing, configuring and using M-eux Text. To make sure you can get started automating your test cases as soon as possible, we invite you to closely follow the instructions contained in this guide.

We have made every attempt possible in making the instructions in this guide as clear as possible. However, we recognize that we are unable to cover everything in a single guide. Should you require further assistance, please do not hesitate to visit our www.jamosolutions.com website or to contact our support team at http://www.jamosolutions.com/member-area/support/.

After reading his document, you should be familiar with the following topics:

- Test automation on real iOS device
- Test automation on iOS Simulator
- Creating first scripts on Eclipse
- Best practices on iOS testing

1.1. M-eux Test overview

To test mobile applications using M-eux Test, you need three main components. The **iOS Device** or **iOS Emulator** hosts the application that you want to test. **Eclipse** is the scripting environment in which you write and execute your scripts. Finally, the **M-eux Test Device Manager** is the core of our application and acts as the gateway between the mobile devices and scripting environment.

M-eux Test currently supports the following products:

- **Eclipse**: M-eux Test extends the 32-bit version of Eclipse Galileo (3.5).
- **iOS Devices:** M-eux Test supports iOS devices (iTouch, iPhone and iPad) and simulators iOS 4.2.x, iOS 4.3.x, iOS 5.x, iOS 6.x and iOS 7.x. Both normal and jail broken devices are supported.

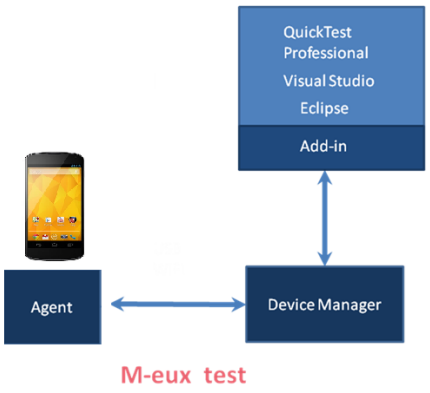


Figure 1: M-eux Test Architecture

1.2. How to use this document

This document will get you started with using M-eux Test for testing applications on iOS using Eclipse. To get started, here's what you need to do:

- **Get your PC ready**. You will first need to install M-eux Test on your PC.
- **Get your device or emulator ready**. Depending on your testing strategy, you may want to run your test on a physical device or an emulator. M-eux Test supports both.
- **Get your app ready**. M-eux Test captures information of your app through instrumentation. To make your app testable, you need to instrument your app.
- **Create your first script**. Once your pc, device and app are ready, you can start creating your first script.

1.3. Where to go from here

This document will get you started creating your first script. If you want to further explore the features of M-eux Test, please visit our website at www.jamosolutions.com for further information on the features of M-eux test.

Chapter 2: Get your PC ready

2.1. System Requirements

The supported systems and requirements:

Computer/processor	IBM-PC or compatible with a Pentium III or higher (Pentium IV or higher recommended) microprocessor.
Operating System	Windows Desktop • Windows XP 32-Bit Edition—Service Pack 3, • Windows Vista (32-bit edition & 64-bit Edition), • Windows 7 (32-bit Edition & 64-bit Edition) • Windows 8 (32-bit edition & 64-bit edition) • Windows 8.1 (32-bit edition & 64-bit edition) Windows Server • Windows Server 2003 32-bit Edition—Service Pack 1, • Windows Server 2003 R2 (32-bit x86), • Windows Server 2008 (32-bit & 64-bit edition) • Windows Server 2008 R2 • Windows Server 2012 • Windows Server 2012 R2
Memory	Minimum of 2 GB.
Color Settings	Minimum of High Color (16 bit).
Free disk space	1 GB of free disk space for application files and folders and an additional 1 GB of free disk space on the system disk (the disk on which the operating system is installed). The free disk space requirements do not include disk space required for any prerequisites that may need to be installed before installing M-eux Test. After M-eux Test is installed, it is recommended to have at least 1 GB free disk space on the system disk for the operating system and M-eux Test to run correctly.

2.2. Prerequisites

To install M-eux Test on your computer, you first need to install the following prerequisites:

- Java SE Development Kit (JDK) 7, 32-bit edition
- Eclipse IDE Galileo version and above 32 bit.
- Optionally, OCR components

Once all prerequisites have been installed, you can continue with the installation of M-eux Test. You may be required to execute some post-configuration tasks to ensure M-eux Test will work correctly in your environment.

2.2.1. Mobile Device Prerequisites

In case of Windows Mobile devices and emulators, ActiveSync from Microsoft is needed in order to connect the PC to the mobile device for Windows XP and Windows 2003. Windows Vista and Windows 7 require the Windows Mobile Device Center.

In case of Android devices, the SDK of Android needs to be installed together with the USB drivers from Android in order to connect the mobile device with the PC using a USB cable.

In case of BlackBerry devices, the "BlackBerry Desktop Software" needs to be installed.

For iOS devices – you can download the latest iTunes from apple website.

2.2.2. Required Privileges

You need to have local system administration rights in order to install the product.

2.2.3. Visual C++ 2010 Redistributable

M-eux Test requires the Visual C++ 2010 Redistributable to be installed on your PC. You can download the Visual C++ 2010 Redistributable from http://www.microsoft.com/en-us/download/details.aspx?id=8328 .

2.2.4. .NET Framework 4.0

M-eux Test requires version 4.0 of the .NET Framework to be installed on your PC. You can download the .NET Framework from http://www.microsoft.com/en-us/download/details.aspx?id=17718 .

2.2.5. Apple iTunes

To support connecting iOS devices to your PC over USB, Apple iTunes needs to be installed. You can download iTunes from https://www.apple.com/itunes/download/.

2.2.6. Installing Eclipse

M-eux Test supports Eclipse packages from Galileo and above (32 bit versions only). You can download it from: http://www.eclipse.org/downloads/packages/

2.2.7. Enable OCR functionality (optional)

M-eux Test does not require OCR functionality for you to create automated test scripts. In some scenarios, your app or web page may include generate pictures (images) on which you want to recognize some text. In this scenario, you need to install the OCR components. M-eux Test leverages the OCR functionality of Microsoft Office.

Microsoft Office 2007 is required to make use of the OCR functionality**.** You don't need to uninstall and reinstall Microsoft Office; just follow this procedure:

1. Click the **Add and remove** button in **Control Panel** and select **Change**.

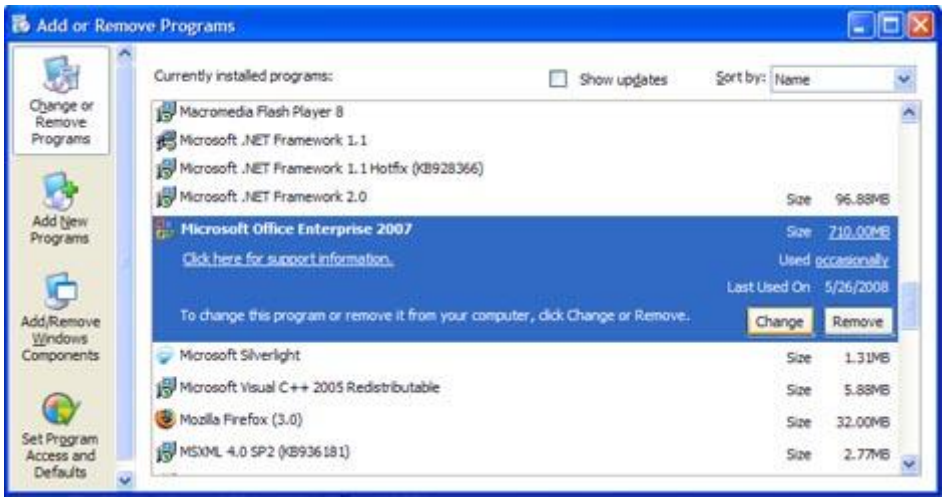


Figure 2 MicroSoft Office from control panel

2. In the Office setup window, select **Add or remove Features** and click **Next**

Figure 3 Add or remove features on Microsoft 2007

3. In the following dialog box select **Office Tool**, **Microsoft Office Document Imaging**, **Scanning**, **OCR and Indexing Service Filter**. Under the drop down list choose **Run from computer** and proceed with the installation.

2.3. Installing M-eux Test

In order to test applications on Android devices or Android emulators, you need to install the M-eux Device manager. As shown in the architecture, the device manager is used to communicate between the device and the test tools.

You can download the set up on our website: M-eux Test Setup at http://www.jamosolutions.com/memberArea/downloads.php. You will need to have a credential to log in the member area network. If you do not have the credentials yet, please email us at info@jamosolutions.com

It is mandatory to install the application with admin rights, simply right click on the **MeuxSetup.msi** and **run as admin rights**.

2.3.1. Preparation

1. First step is to execute the **MeuxSetup.msi** file from the M-eux build you downloaded on your PC. The dialog boxes will guide you through the installation.
2. Click **Next**

Figure 4 Installing M-eux Test

3. Setup will check the components which are already installed on the pc and will indicate if any additional component is required. Review all warnings and errors and click **Next** to continue the installation process.

   Note: The *Android SDK check will issue a warning if you have downloaded and copied the zipped version of the SDK. No warning will be displayed if you have used the installer to install the Android SDK. Please see the instructions here for more information.*

4. If you agree with the License Agreement, select **I accept the terms in the License Agreement** and click Next.

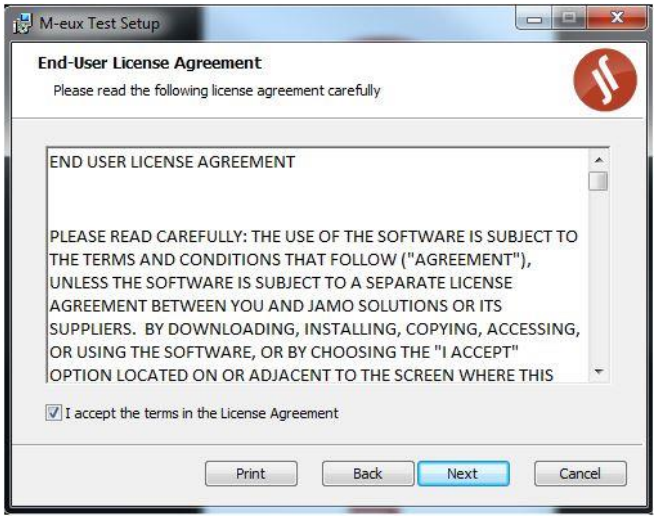


Figure 5 Accepting license

5. Select:
    a. Unified Functional Testing (QuickTest Professional) if you want to use M-eux Test with QTP/UFT.
    b. Select Visual Studio 2005 if you want to use M-eux Test with Visual Studio 2005 sp1.
    c. Select Visual Studio 2008 if you want to use M-eux Test with Visual Studio 2008 sp1.
    d. Select Visual Studio 2010 if you want to use M-eux Test with Visual Studio 2010.
    e. Select Visual Studio 2012 if you want to use M-eux Test with Visual Studio 2012.
    f. Select Visual Studio 2013 if you want to use M-eux Test with Visual Studio 2013.
    g. Select Execution M-eux Visual Studio scripts if you want to execute only Visual Studio based scripts on this PC.
    h. Select Eclipse Extension if you want to use M-eux Test with Eclipse

    Note: *You cannot extend Visual studio 2008 and Visual studio 2005 at the same time.*

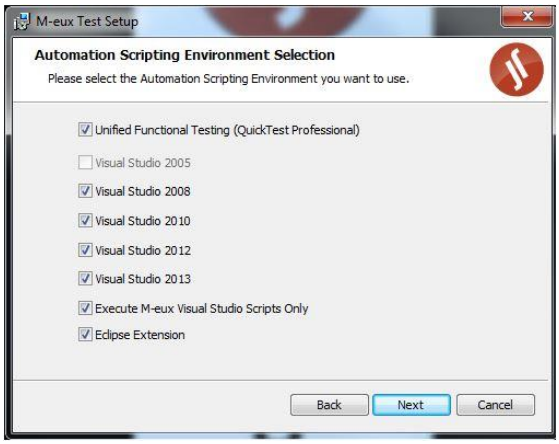
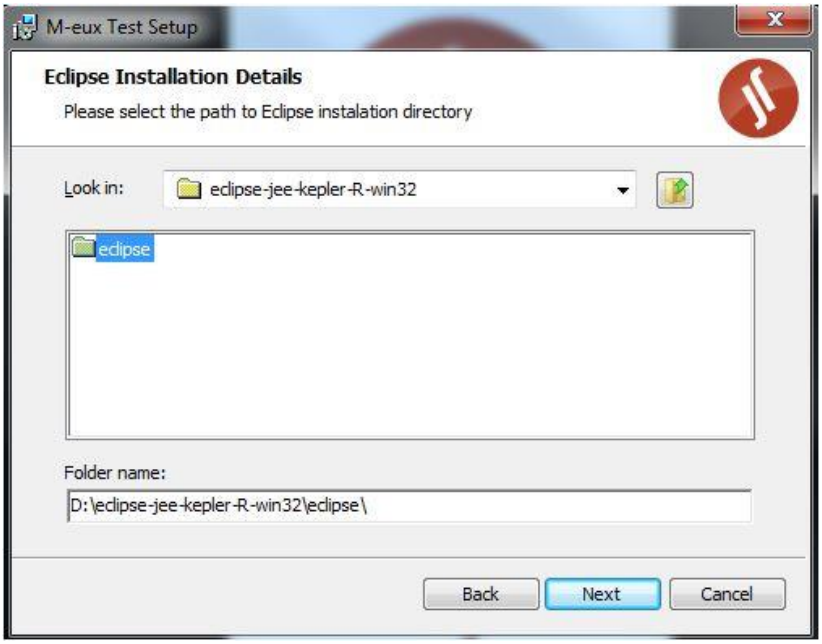
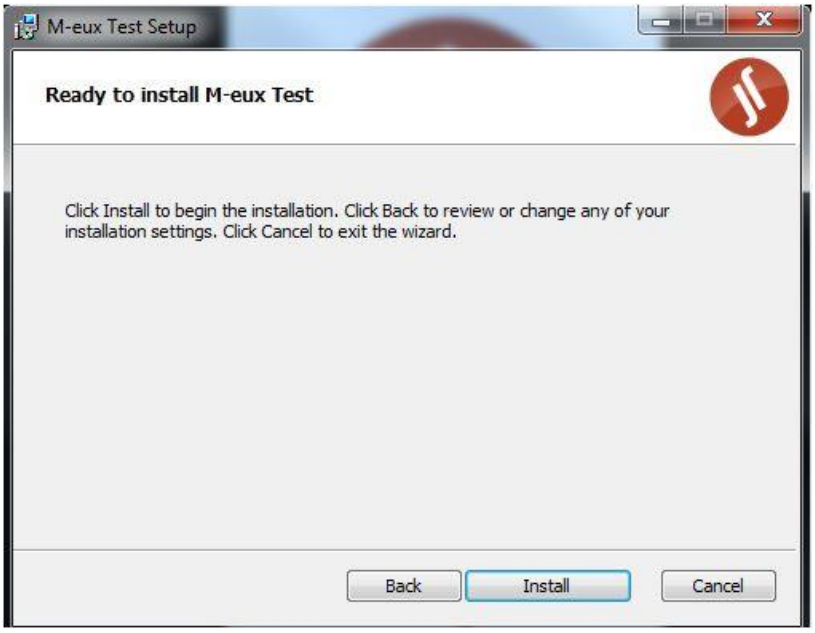


Figure 6 Choose Extensions

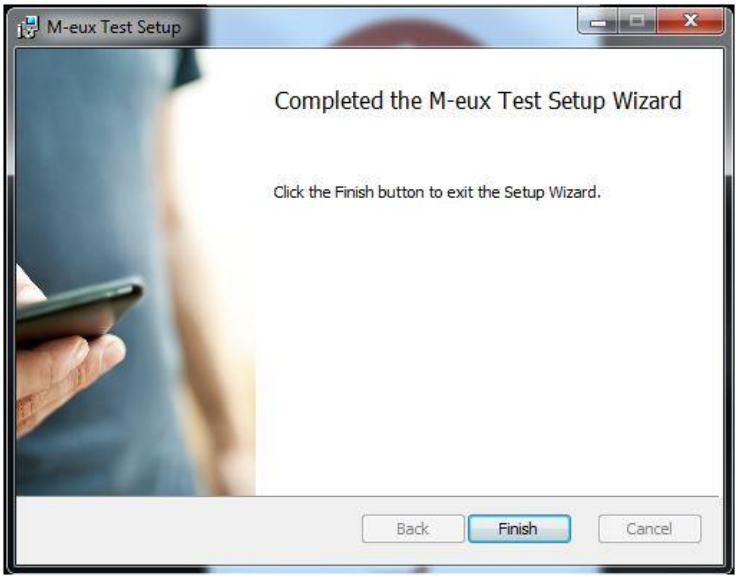
6. If you have selected the Eclipse Extension then setup will ask you to point to the directory where Eclipse is installed. Please make sure you point to a 32-bit version of Eclipse.



7. Click on “Install” to start the actual installation.



8. Wait until Setup is has completed. Depending on your environment, this may take up to 10 minutes as setup is preparing your PC for smooth automation.



2.3.2. License activation

Start the device manager application on the PC.

When starting the first time the device manager, you get the device manager without possibility to connect devices or tools. You have to install a license first. Depending on the purchased license(s), you can choose between seat, site and token license.

Seat License activation

In case of a seat license, select the menu **Tools**, **License**, **New license**, **Seat**. Write down the locking id and send this id to Jamo Solutions: info@jamosolutions.com . Jamo solutions will return the file with the license keys. A path name to this file needs to be entered in the edit field linked to the Seat License Details section. You can write the path manually or use the 'Select button' to find and select your license key file.

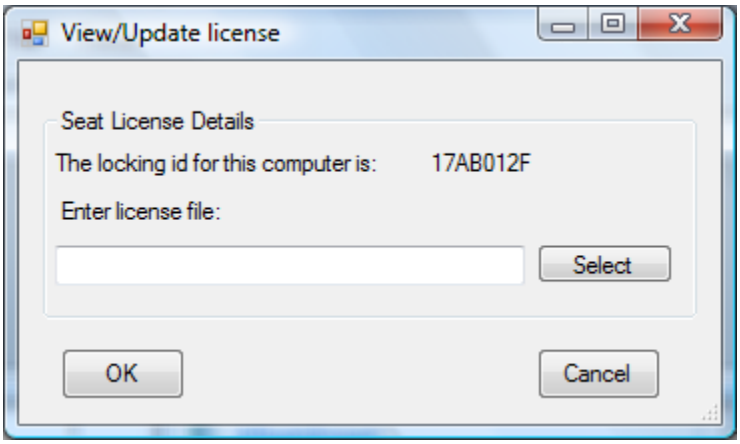


Figure 7 Seat license activation

After selecting "OK", the file will be read and the seat license will be installed.

Site license activation

In case of a site license, select the menu **Tools**, **License**, **New license**, **Site**. Enter the IP or hostname and port of the license server. The license server URL will be generated automatically.

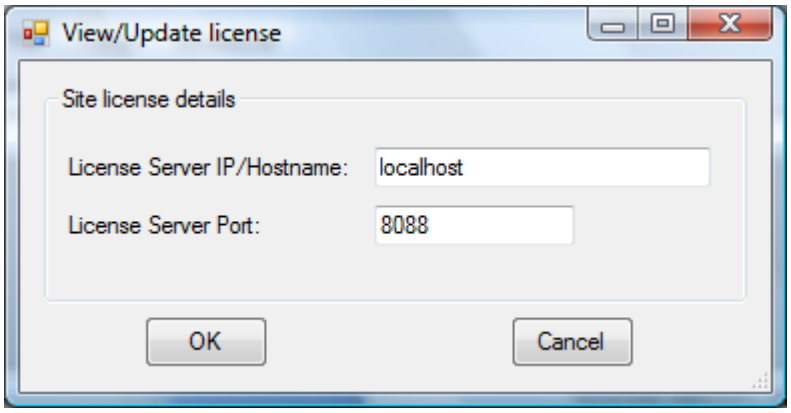


Figure 8 Site license activation

Note that prior to connecting to the license server machine, the license server needs to be installed on that machine. The licenses will be checked at run-time, so you need to be connected to the license server all the time while using the Device Manager.

If you have problems connecting to the license server, then please contact your product administrator who is responsible for the M-eux license server.

Note the site license will be released if you shut down your device manager. So if you want to transfer the site license to another tester in your company. You just need to shut down your M-eux Device Manager.

Token license activation

Token licenses are distributed by your local license server. They allow you to use the tool while not being connected to the license server.

In case of a token license, select the menu **Tools**, **License**, **New license**, **Token**. In the details enter the IP or hostname and port of the license server. The license server URL will be generated automatically.

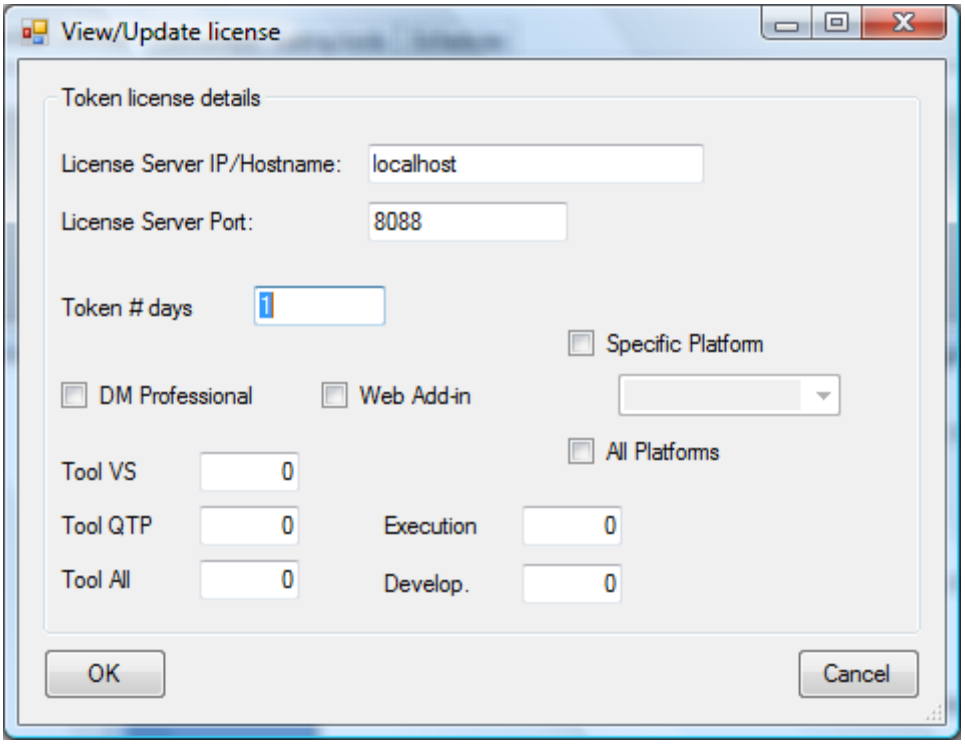


Figure 9 Token license activation

Note that prior to connecting to the license server machine, the license server needs to be installed activated and running on that machine.

- The checkbox **Web Add-in** consumes a web support for each tool demanded license.
- The **Token # days** edit field contains the number of days that the token will be used by the machine, i.e. the number of days that the token will be valid.
- The checkbox **DM Professional** denotes the license for the device manager professional edition. A license device manager professional activates the WAN Connector and local scheduler functionality.
- The checkboxes **Specific Platform** and **Platform All** denote the platform of your connected devices. Only one of these checkboxes can be checked.
- If you check **Specific platform** you have to select in the combo box under the platform you wish. If you select nothing, the Windows Mobile/CE platform will be levied. Selecting **Windows**

Mobile/CE denotes that Mobile-based or Windows CE-based devices can connect to the Device Manager directly or indirectly through the WAN connector. Selecting **Android** denotes that android-based devices can connect to the Device Manager directly or indirectly through the WAN connector.

- Checkbox **Platform All** denotes that any supported mobile device can connect to the Device Manager directly or indirectly through the WAN connector.
- The other fields contain the number of token-licenses you want to have active for the specified time period indicating that all tokens have the same duration as specified in **Token #days**. Following table explains the tokens one can request:

Token Edit field	Description
Development	Denotes the number of development tokens. With one development token, the QA engineer can create, modify and execute a script. Normally this number is one since most of the time the QA engineer is working on one script at a time.
Execution	Denotes the number of execution tokens. With one execution token, you can execute a script. With N execution tokens, you can execute N scripts simultaneously.
Tool (Eclipse)	Denotes the number of Eclipse tokens. With one Eclipse token, you can launch one Eclipse based test script. With N Eclipse tokens, you can launch N Eclipse based test scripts simultaneously.
Tool QTP (QuickTest Professional)	Denotes the number of QuickTest Professional tokens. This number is 0 or one since you can run only one QuickTest Professional instance at a time.
Tool ALL	Denotes the number of tool instances you can launch simultaneously. Currently Eclipse and QuickTest Professional are supported.

After clicking OK, you will receive an overview of requested and received licenses. If all requested licenses are not granted, it means that they are in use by other users.

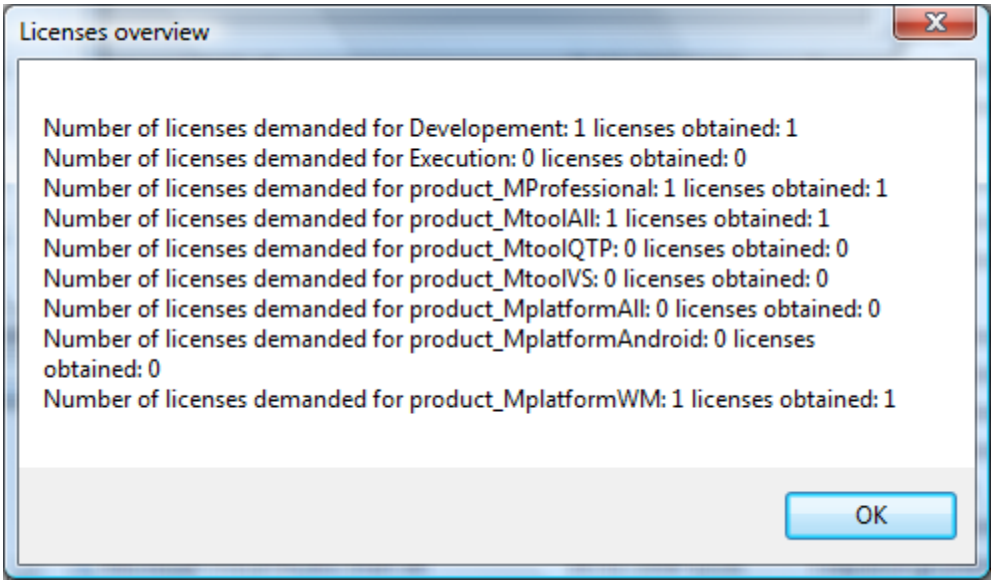


Figure 10 Token license overview

Note that for any license installation (install new license, switch to another license or release token license) all devices and tools have to be disconnected.

Handling the token license

All token licenses have an expiration date. If you want to release these tokens before the expiration date, you can do it by calling the menu item **Tools**, **License**, **Release Token License** or by installing a new license (seat, site). Before calling one of these menu items, you need to connect your machine to the network so that the license server is reachable. The menu item **Release Token License** will release all tokens and switch your license to site license connecting to the same license server address that was used to book and un-book the tokens.

Managing the professional license

The Device Manager will at startup time look to book a Professional license. You can release this license and use the Device Manager in Standard mode by selecting **Tools**, **Licenses**, **Release Device Manager Pro**. Performing this action will disconnect all connected WAN connectors and disable the scheduler.

If the Device Manager is in Standard mode, you can book a Professional license by selecting the menu **Tools**, **Licenses**, **Get Device Manager Pro**.

You can verify the mode the Device Manager is in by inspecting the menu entry in **Tools**, **Licenses**. The menu entry **Release Device Manager Pro** indicates that the Device Manager is in Professional mode. The menu entry **Get Device Manager Pro** indicates that the Device Manager is in Standard mode.

License overview

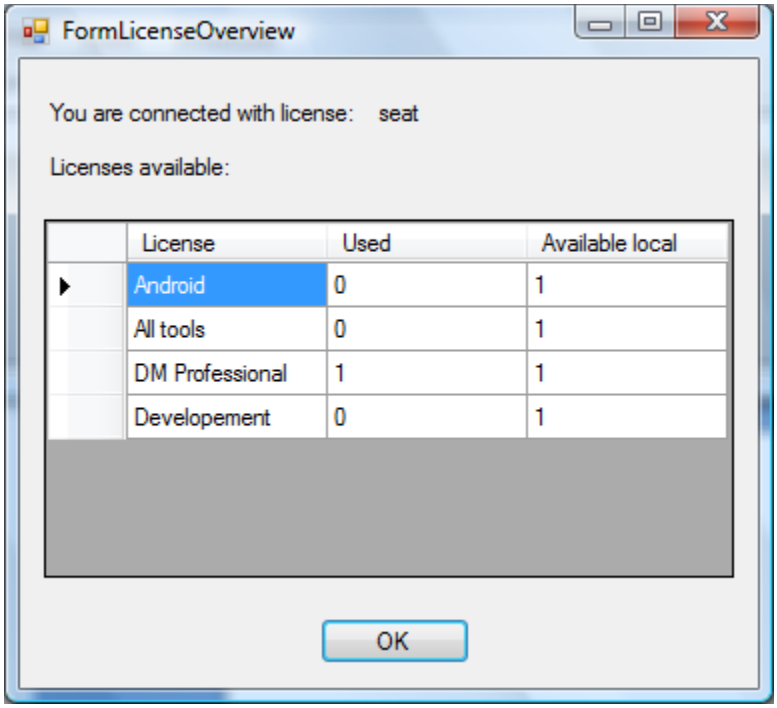


Figure 11 License overview

At each time you can check your available licenses in menu **Tools, Licenses**, **License Overview**. The displayed form shows the number of licenses in use and if using a seat or token license, the number of licenses available locally on your PC.

2.4. Troubleshooting

2.4.1. The device is not able to (re)connect to the M-eux device manager

The communication between the PC and the mobile device is using the TCP/IP network protocol. A local firewall installed on the PC may block the communication between the PC and the mobile device. The TCP/IP protocol uses **ports** and **IP addresses** to establish communications.

We recommend you allow communication on the following IP-address blocks:

- 169.254.*.* (autoconfiguration IP addresses)
- 192.168.*.* (private IP range)
- All other IP ranges that are used for establishing connections between the mobile device and the desktop (e.g. Wi-Fi, Bluetooth, …)

And on the following ports:

- 4444, 555, 5580-5590

2.4.2. The device manager ports are already in use

Since the communication between the Device Manager and the script (and vice versa) is using .Net Remoting technology, ports are needed for communication. The ports that are used by the M-eux Test tool can be modified in "MeuxDeviceManager.exe.config" (see the bin folder of the installation directory). **After launching Device Manager or Wan Connector you get message(s) that a port is already used on this machine**

You have to go to the configuration file of Device Manager (Wan Connector) and change the specified port there. For details see the User guide, section Wan Connector configuration.

1. Make sure your local firewall does not block the communication of the device manager, you can enable the communication by: Control panel—Firewall—Allowed programs and make sure you have device manager listed in there and it can communicate through firewall in both home and public network.

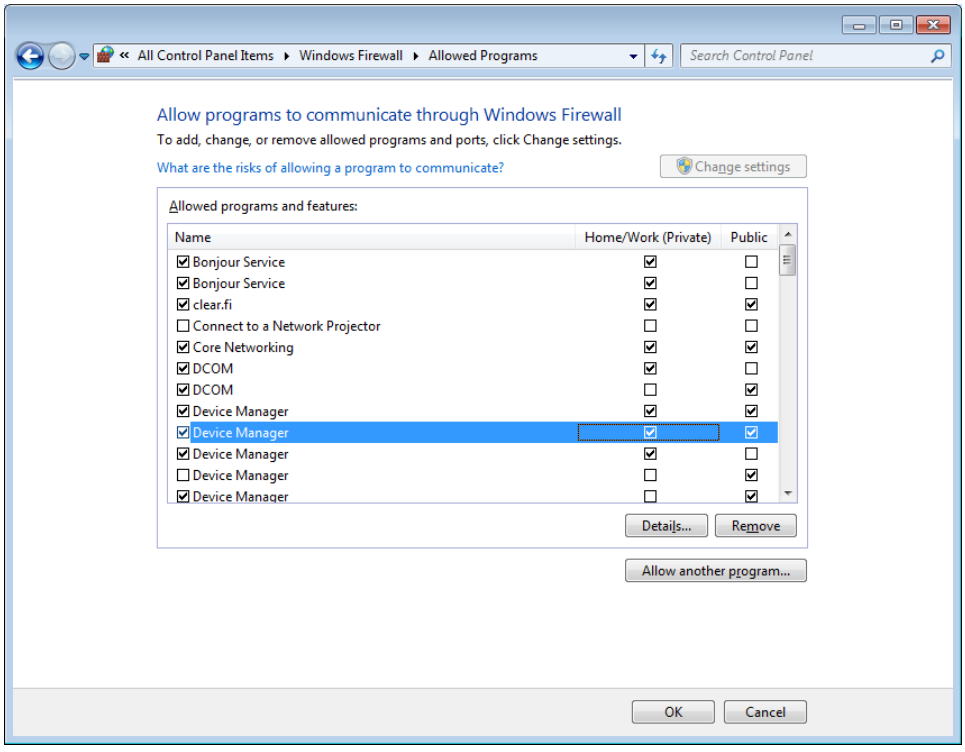


Figure 12 allow the communication of Device Manager on firewall

If the Device Manager program does not appear in the above window, you can click the button **Allow another program** in the above picture and point to **MeuxDeviceManager.exe** in the bin directory of the M-eux Test.

Chapter 3: Getting your device ready

3.1. Installing the M-eux Applications

There are multiple M-eux Applications that you can install on your device to facilitate the creation of your test scripts:

1. The **M-eux Agent** is an application you will need to install on the device in order to :
   a. Forward the communication between the application under test and M-eux Device Manager.
   b. Initiate the connection from the device to the M-eux Device Manager.

   After installation, there will be an application with the logo of Jamo Solutions appearing on your application list:

Figure 13 M-eux Agent icon

2. If you want to start testing with our sample application, which is already made testable, you can also install the Sample Application. The sample application is called **UI Catalog**. After installation, there will be a new application on your device:

Figure 14 UI Catalog icon

   If you want to start testing your own application, you will need to make your application testable follow here: How to make your application testable.
3. If you want to test web application, you need to install this sample browser. We call this sample browser the **Test Browser**. It is the default iOS browser but made testable. After installation, your device will have an additional application:

Figure 15 Test browser icon

3.1.1. Preparing the M-eux Test Applications

For each of the M-eux Test Applications, you need to prepare them first sign them with your own certificates and install them on the device.

The process for building the M-eux Test Applications is the same for all applications. In this section, we will describe the process in detail or the M-eux Test Agent. You can re-use the same steps for the other applications. Please refer section 4.2.

3.1.2. Install M-eux Agent using iTunes or iPhone Configuration Utility on Windows

Please note: although you can install M-eux agent and applications using iTunes or the iPhone Configuration Utility on Windows, you need to resign the M-eux agent and applications on an Apple machine. Please refer to section 3.1.1 for more information on how to do this.

If you do not have an Apple machine at your disposal, please contact our support team at support@jamosolutions.com for further assistance.

M-eux Agent, Test Browser and Sample Application:

Before you start, make sure the **iTunes** or **iPhone Configuration Utility** application is already installed in your windows PC. You can download from apple support.

- iPhone Configuration Utility : http://support.apple.com/kb/DL1466
- iTunes : http://support.apple.com/downloads/#itunes

After you have compiled or prepared (using testable creator) the M-eux Agent and applications on an Apple machine, copy the distribution files to a directory on your hard disk. The distribution of an iOS application is a directory containing different files.

To deploy the application:

1. Open the iPhone configuration utility program.
2. Select in the left hand vertical bar the Applications entry.
3. Click on the add button as shown in figure 5.

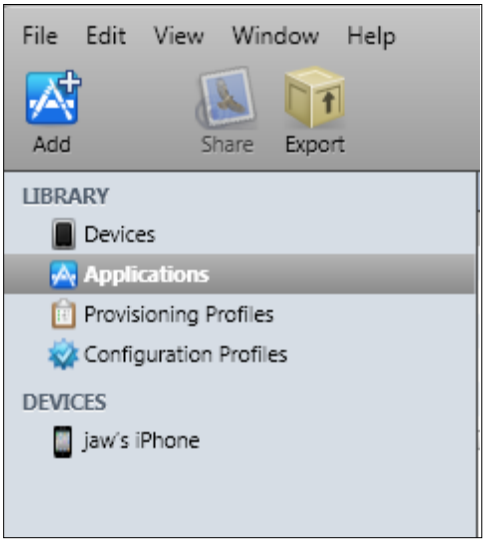


Figure 5: iPhone configuration utility

4. A dialog box opens. Select the directory representing the application.
5. The agent application will be listed. The folder has separate .ipa files for each iOS version, Select the correct file based on your device OS version.

For example: If you are working on iPhone OS version 6.1 , copy file << *meuxAgent6_1AdHoc_b2468.ipa*>>, If you are working on iPhone OS version 5 , copy file << *MeuxTestBrowser5AdHoc_b2468.ipa*>>. You can also add the application by dragging and dropping the .ipa file in the list as shown in Figure 6.

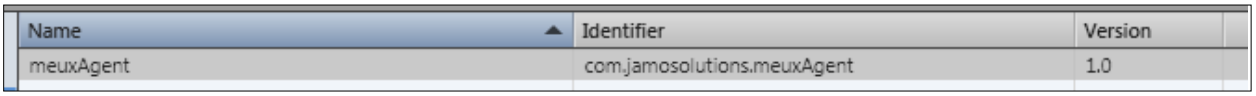


Figure 6: The application list

6. Connect your device to the PC using the USB cable. The device will be listed in the device section of the left hand vertical bar.
7. Select the device.
8. Select the tab labeled 'Applications' and click on the install button as in Figure 7.

Figure 7: The application that can be installed for you device

Please follow same steps to install Sample Test browser and Sample application as well.

3.2. Connect agent with M-eux Device Manager

You will need to use the agent to initialize the connection from the device. There are two ways of connecting the agent and the device manager, Wi-Fi and USB.

Figure 8: User interface of Meux Agent

3.2.1. Connect using Wi-Fi

To connect using Wi-Fi, please follow the following step:

1. On your PC, start the **Device Manager**.
2. On your iOS device, make sure that the device is on by navigating to **Settings**, **Wi-Fi** and make sure Wi-Fi is switched on.
3. On your iOS device, make sure the device is connected to the correct Wi-Fi network and make sure the device and PC on the network are able to communicate.
4. On the device, launch the **M-eux agent.**

5. In the M-eux Agent home screen, click **Connect**.
6. In the **IP Address** text field, type the IP address of the PC.
7. In the **Device Name** field, type the name that will show up in the device manager as in the below figure.
8. Toggle the **Connect WIFI** button to **ON** as shown in Figure 8.
9. If you want your device to connect with the device manager automatically after booting the device or launching the agent, you can turn the second switch labeled as **Connect @ boot/launch** on. If you do so, but please ensure that your device manager is already started the PC before the device is booted to avoid the device becoming black screen.
10. On the device manager on the PC side, you will see the following:

Figure 9: Device Manager

3.2.2. Connecting using USB

Before you connect using USB, make sure the **iTunes** application is already installed in your PC. You can download from apple support.

- iTunes : http://support.apple.com/downloads/#itunes

And then, please follow the steps below:

1. Connect iOS device to your PC through USB.
2. On your PC, start the **Device Manager**.
3. On the device, launch the **M-eux agent.**
4. In the M-eux Agent home screen, click **Connect**.
5. Click on USB button at the bottom.
6. In the **Device Name** field, type the name that will show up in the device manager as in the below figure.
7. Toggle the **Connect USB** button to **ON** as shown below.

8. On the device manager on the PC side, you will see the following:

Figure 10: Device Manager

In this way, you created the connection between the device and device manager using USB cable.

Note: M-eux test currently does not support multiple devices connect to Device Manager. This feature will be included in future release.

3.2.3. Connecting using USB-HOTSPOT (Optional)

Prerequisites-

1. You have installed iTunes on your PC
2. You have the hotspot functionality on your device.

To check if you have this functionality on the device:

1. Go to the **Settings**, **General** and **Network**.
2. Check if you have an entry for **Personal Hotspot**. The hotspot functionality is only activated if you have a SIM card with a data plan (iPad 3 or higher). In this way the agent/device manager connection will go then through the USB cable and not through the air.

If you have both the above two available, please follow the below steps to set up USB connection between agent and device manager.

1. Enable **Personal hotspot** in the settings of the iPhone as described above.
2. Enable the internet tethering on the iOS device:
    a. Go to **Settings**, **General** and click **Mobile Data**
    b. Go down to **Mobile Data Network**
    c. Fill in the internet tethering data. It might differ depends on the data carriers.
    NOTE: For iOS 5 devices, you will need to reset the network by going to **Settings**, General, **Reset** and tapping **Reset network settings**.
3. Connect the iPhone using USB to the PC.
4. ITunes will start up on the PC.
5. ITunes will create a new network card on your PC automatically.
6. Open through the control panel the network cards, select the one from iTunes and ask for the details.
7. You will see the IP address assigned to this network card.

    Alternatively, you can open a command window and type: ipconfig /all. This will list all your network cards and their configuration.
    Write down the IP address of the one created by iTunes. Most likely the IP address is: 172.20.10.2

11. On your PC, start the **Device Manager**.
12. On your iOS device, make sure that the device is on by navigating to **Settings**, **Wi-Fi** and make sure Wi-Fi is switched on.
13. On your iOS device, make sure the device is connected to the correct Wi-Fi network and make sure the device and PC on the network are able to communicate.
14. On the device, launch the **M-eux agent.**
15. In the M-eux Agent home screen, click **Connect**.
16. Click on Hotspot button at the bottom.
17. In the **IP Address** text field, type the IP address of the PC.
18. In the **Device Name** field, type the name that will show up in the device manager.
19. Toggle the connect button to **ON.**

In this way, you created the connection between the device and device manager using USB Hotspot option.

3.2.4. Trace route your network

In case you are experiencing some network problems and you want to check through which network hubs your device is connected, you can use the trace route functionality from the M-eux Agent. It will show you the first a few IP-addresses which your device uses to reach the destination. (You may specify either the IP address or the Host name).

The trace route looks like below:


Jamo's iPhone5
Size
Spy
Encoding
Input
Quality
No SIM
14:40
Back
Wifi Connection
Connection
Connect WIFI
ON
Status:
Connected
Settings
IP address:
10.32.153.159
Port number:
4444
Device name:
Jamo's iPhone5
Connect @ boot/launch
OFF
TraceRoute
WIFI
USB
HOTSPOT

Chapter 4: Getting your app ready

For any application that needs to be tested, it needs to be testable; that is, it must include our library. Our library will allow the application under test to communicate with your testing environment.

You can make an application testable for testing on both the simulator and a physical device.

4.1. Testable Creator: Making your iOS application testable without source code access

In order to identify the objects of applications that are going to be tested on an actual device, the application needs to be made testable. To make an application testable, you need to have access to:

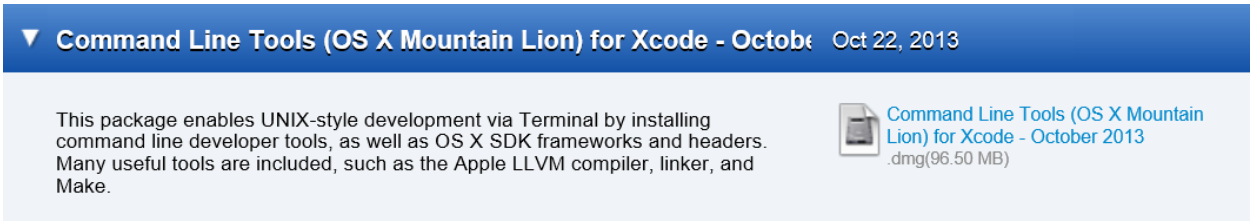
- A Mac machine with XCode 6 and the XCode command line tools installed.
- The ipa (binary application) file.
- A valid Apple provisioning file and code signing certificate.

You do not need to have access to the source code.

4.1.1. Requirements

In order for you to install the testable creator, you need to have the following ready:

1. An Mac OS X machine with the latest version of XCode 6 installed.
2. The Command Line Tools (OS X) for XCode for your version of Mac OS X. They need to be installed in addition to XCode 6. You can download them from https://developer.apple.com/downloads/index.action.



3. Java 1.7 or later for your Mac OS X machine. You can download Java from http://www.java.com/en/download/index.jsp.
4. A valid provisioning profile. You need the provisioning profile to sign the testable ipa file. For more information on how to get a provisioning profile, please refer to the Apple Documentation.
5. A code signing certificate. Please use iPhone Distribution certificate. If the certificate list is empty, please make sure the current have the developer certificate

4.1.2. Installing the Testable Creator

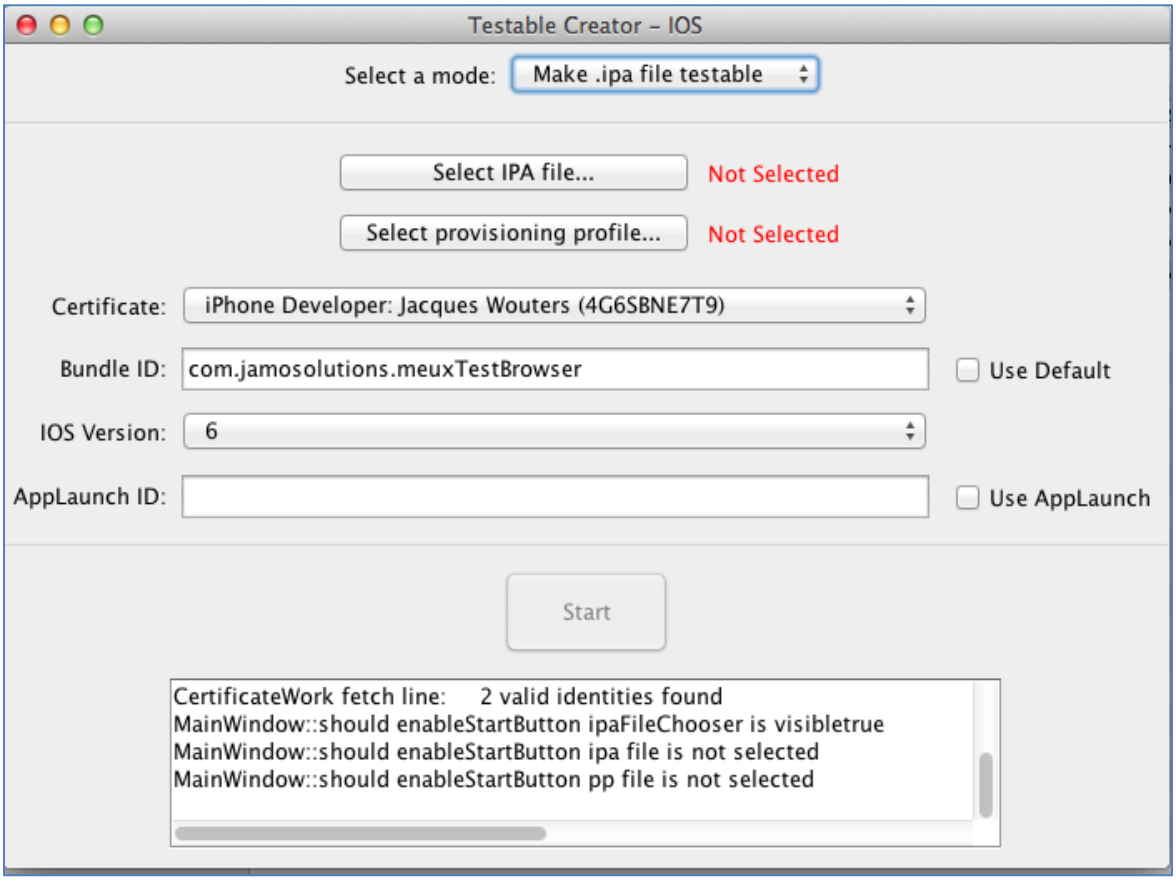
To install the Testable Creator, please follow these steps

1. Log on to a Windows PC where you installed M-eux Test
2. Navigate to C:\Program Files (x86)\Jamo Solutions\M-eux Test\setup\iOS
3. Copy the file testableCreator.zip file from your Windows PC to your Mac OS X machine
4. On your Mac OS X machine, double click on the file to unzip it

4.1.3. Starting the Testable Creator

To start the Testable Creator, please follow these steps:

- Navigate to the folder where you extracted the testableCreator.zip file.
- Double click on TestableCreator.jre file. If the application does not open, refer to the Requirements section.



4.1.4. Making your application testable using the UI

To make your application testable, please follow these steps:

1. Open the Testable Creator.
2. In the **Select a mode** dropdown, choose **Make .ipa file testable.**
3. Click on **Select IPA file** and choose the application (.ipa file) you want to make testable.
4. Click on **Select Provisioning profile** and choose a provision profile file.
5. In the **Certificate** dropdown list, select a **iPhone Distribution** certificate. If the list is empty refer to Preparation point 7.
6. Check the "Use Default" checkbox to use default bundle if of your application **(or)** Type in the bundle identifier of your app (e.g com.jamosolutions.uicatalog).
7. Select your iOS system version. This is the version of iOS of the device on which the application is going to be installed.
8. Check the **Use AppLaunch** checkbox and Type in any string in **AppLaunch ID** text box. For example, you can use an AppLaunch ID such as UICatalog. This string is used as an argument for AppLaunch() function to launch application from script. Please see our function reference for

more details. You can navigate to **Device Manager**, **Help**, **Function reference**, **Contents**, **Non-GUI Test Objects** and **iOS Mobile Device**.

9. Click **Start**.
10. A file named Testable+yourAppName.ipa file will be generated in the same directory where your ipa file is situated.
11. Install the testable+yourAppName.ipa file using iPhone configuration utility software on MAC or PC.
12. If you have any problems in installing or creating testable App, please send logs that are displayed below "Start" button to our support support@jamosolutions.com

4.1.5. Prepare the M-eux Agent using the UI

You need to sign the MeuxAgent with the same certificate as the certificate used to sign the testable version of the application under test.

1. Copy the MeuxAgent.ipa file from C:\Program Files (x86)\Jamo Solutions\M-eux Test\setup\iOS
2. Double click on TestableCreator.jre file. If the application does not open, refer to Preparation point 3
3. On **Select a mode**, choose **CodeSign .ipa file**
4. Click on **Select IPA file** and choose meuxagent.ipa file from ad hoc folder
5. Click on **Select Provisioning profile** and choose provision profile file.
6. Select Certificate as **iPhone Distribution**. If the list is empty refer to Preparation point 7.
7. Check the **Use Default** checkbox to use default bundle if of your application **(or)** Type in the bundle identifier of your app (e.g com.jamosolutions.uicatalog).
8. You do not need to provide any value for the **AppLaunch ID** text box.
9. Click **Start**
10. A file named codeSigned+agentName ipa file will be generated in the original directory where your agent.ipa file is situated.
11. Install the generated codeSigned+agentName.ipa file using iPhone configuration utility software on MAC or PC. Refer here to know how to install ipa files using iPhone configuration utility software.

4.1.6. To Make application Testable using the Command Line

To make your application testable using the Command Line, please follow these steps:

1. Open **Terminal**
2. Use following command to make your **application testable**.
   Command: java -jar "PathToTestableCreatorFolder/testableCreator.jar" "testable" "PathToIpaFile/example.ipa" "PathToProvisioningProfile/jamo.mobileprovision" "Certificate Name" "UniqueBundleID" "iOSVersion" "AppLaunchID"

Parameter	Description
PathToTestableCreatorFolder/ testableCreator.jar	The "Path of Testable Creator" including the file testableCreator.jar
testable	The testable creator mode. This can be **testable** or **codesign**. To make an application testable, use testable.

PathToIpaFile/example.ipa	the path of the ipa file including the .ipa extension
PathToProvisioningProfile/jamo.mobileprovision	The path of provisioning profile
Certificate Name	The certificate string. If you do not know the certificate name, you can open the GUI of testable creator and copy the Certificate Name displayed there. For Example, “iPhone Distribution: Jamo Solutions NV (498B2MENE5)”
UniqueBundleID	The unique bundle id. For Example, com.jamosolutions.uicatalogue
iOSVersion	The version of iOS for which you want to make the application testable. Valid values include "4.3", "5","5.1", "6","6.1" and "7"
AppLaunchID	App Launch ID, For example, you can use an AppLaunch ID such as UICatalog. This string is used as an argument for AppLaunch() function to launch application from script. Please see our function reference for more details. You can navigate to **Device Manager**, **Help**, **Function reference**, **Contents**, **Non-GUI Test Objects** and **iOS Mobile Device**.

An example command line is:

```
java -jar "/Users/jacqueswouters/Documents/new
testablecreator/testableCreator.jar" "testable"
"/Users/jacqueswouters/Documents/new testablecreator/Apps/UICatalog.ipa"
"/Users/jacqueswouters/Documents/new testablecreator/pp/wildcard1.mobileprovision"
"iPhone Distribution: Jamo Solutions NV (498B2MENE5)"
"com.jamosolutions.UICatalog" "6.1" "UICatalog"
```

4.1.7. Prepare the M-eux Agent using the Command Line

To make prepare the M-eux Test Agent using the Command Line, please follow these steps:

1. Open **Terminal**
1. Use following command to codesign your agent.
2. **Command:** java -jar "PathToTestableCreatorFolder/testableCreator.jar" "codesign" "PathToIpaFile/example.ipa" "PathToProvisioningProfile/jamo.mobileprovision" "Certificate Name" "UniqueBundleID" "" ""

Parameter	Description
PathToTestableCreatorFolder/testableCreator.jar	The “Path of Testable Creator” including the file testableCreator.jar
testable	The testable creator mode. This can be **testable** or **codesign**. To make codesign the agent, use **codesign**.
PathToIpaFile/example.ipa	the path of the ipa file including the .ipa extension
PathToProvisioningProfile/jamo.mobileprovision	The path of provisioning profile

Certificate Name	The certificate string. If you do not know the certificate name, you can open the GUI of testable creator and copy the Certificate Name displayed there. For Example, “iPhone Distribution: Jamo Solutions NV (498B2MENE5)”
UniqueBundleID	The unique bundle id. For Example, com.jamosolutions.uicatalogue
iOSVersion	For codesign, you can leave this parameter blank (“”)
AppLaunchID	For codesign, you can leave this parameter blank (“”)

An example command line is:

```
java -jar "/Users/jacqueswouters/Documents/new
testablecreator/testableCreator.jar" "codesign"
"/Users/jacqueswouters/Documents/new testablecreator/Apps/meuxAgent5.1.ipa"
"/Users/jacqueswouters/Documents/new testablecreator/pp/wildcard1.mobileprovision"
"iPhone Distribution: Jamo Solutions NV (498B2MENE5)"
"com.jamosolutions.meuxagent" "" ""
```

4.2. Making your iOS application testable on Simulator

Depending on the availability of the source code of the application, there are two different ways to make iOS application testable on simulator.

4.2.1. Without iOS application source code

When the source code of your application is not available, the application can be made testable by linking Dynamic Link ClientAgent to Xcode Simulator App. Please follow the steps below.

1. Install your App on the IOS Simulator:
    a. Go to your Project folder open your App with Xcode by double click (usually your xcode project will be named App_name.xcodeproj).
    b. In the Xcode Scheme (left corner in Xcode) specify your Simulator version i.e iPad 6.1 Simulator.
    c. Press the Play button (cmd+R), after the build process has been finished please press the Stop button to stop running the App from the Xcode.
    d. Right click on Xcode icon -> open developer tool -> iOS Simulator, make sure that you recognize your App in the simulator launch it and check if it function as expected.
    e. Quit your App by double pressing on Home button of your Simulator, long Press on the in the bottom of your Simulator Screen and then press the “x” symbol.
    f. Close the Simulator.
2. Moving our Library to */usr/lib*:
    a. Let’s assume that the name of our Library is “clientdylib6.dylib” and right now it is located on your Desktop.
    b. Open your Terminal. By default Terminal will be opened in your user directory.
    c. Go to your Desktop directory type: “cd Desktop”.

   d. Right now you should be in your Desktop, type: "ls –l" and make sure that you recognize the library "clientdylib6.dylib".
   e. Type: *"cp clientdylib6.dylib /usr/lib"*
   f. *Go to /usr/lib* and make sure that our library was copied.
3. Find your App on Mac machine:
   a. Go to directory where your App is located:

      In Finder: press "GO" -> go to folder ->

      Type:" /Users/**Your_user_name**/Library/Application\Support/iPhone\ Simulator/**IOS_simulator_version**/Applications" i.e "/Users/Solutions/Library/Application Support/iPhone Simulator/6.0/Applications"

      In Terminal type: " *cd /Users/Solutions/Library/Application*\ *Support/iPhone*\ *Simulator/6.0/Applications*". (Change your user name and Simulator version)

   b. In that Folder you should see all your installed Apps on the simulator.
   c. Those folders are named with numbers and letters find the folder which contains your app.
   d. Go to this folder.
   e. Go to the App type:*"cd "My_app_name""*
4. Link our *clientdylib6.dylib* to your App:
   a. Assuming that you are in the directory of your App in the Terminal.
   b. Type the command: *"install_name_tool -change /usr/lib/libobjc.A.dylib /usr/lib/clientdylib6_1.dylib"*. The following command will replace *libobjc.A.dylib* with our library (don't worry the replaced library will be linked through our library).
   c. Please check that you link command succeeds type: *"otool -L "Your_App_name"".*
   d. You will see all the Linked Libraries to your App. Verify that you can recognize our library(*clientdylib6_0.dylib*) in the list.
   e. In case the list is too long Type:" otool -L name_of_app | grep *clientdylib6_0.dylib"*
   f. You should see a line with the name of our library and its location.
   g. If you **don't** see the name of our library it means that the link command didn't succeed. Repeat the **(3), (4)** steps.
5. Final Verification:
   a. Open IOS Simulator.
   b. Open the Agent in the Simulator.
   c. Connect the Agent to device manager.
   d. Launch your App.
   e. Go to Device manager on your Windows machine.
   f. Open RDS.
   g. Verify that your App can function in our RDS.

4.2.2. With iOS application Source Code

The process of making application testable on Simulator is same as the real device. Please follow here: How to make Application Testable.

4.3. Making your iOS application testable on real device (when Source Code available)

In order to identify the objects of applications that are going to be tested on an actual device, the testability interface of M-eux Test needs to be included. Making application testable on real device is also possible when source code of the application is available. (Already described how to make application testable using ipa file in section 4.2)

Note that the screen shots in this section are taken from XCode 4.2. You can compile also in XCode 3.2 and up.
This is done by execution of the following steps:

1. Copy from the installation directory of M-eux Test on the PC the files libClientAgentx.a(x is the version of the ios version on your device) and ClientAgent.h to the Mac machine where you make adapt the application.
2. Open the XCode project of your application.
3. Go to your application delegate class and code following additions:
   a. Import the ClientAgent.h file:
      #import “ClientAgent.h”

```
#import "AppDelegate.h"
#import "MainViewController.h"
#import "ClientAgent.h"
```

   b. Define inside the implementation section the variable clientAgent:
      ClientAgent *clientAgent = nil;

```
@implementation AppDelegate

@synthesize window, navigationController;

ClientAgent *clientAgent = nil;
```

   c. Go or add the applicationDidFinishLaunching protocol method of the UIApplication delegate and add in the body of this method following statement:

```
        clientAgent = [[ ClientAgent alloc] init];
- (void)applicationDidFinishLaunching:(UIApplication *)application
{
    // To set the status bar as black, use the following:
    // application.statusBarStyle = UIStatusBarStyleBlackOpaque;

    // this helps in debugging, so that you know "exactly" where your views are placed;
    // if you see "red", you are looking at the bare window, otherwise use black
    // window.backgroundColor = [UIColor redColor];

    // add the navigation controller's view to the window
    [window addSubview: navigationController.view];
    [window makeKeyAndVisible];
    clientAgent = [[ ClientAgent alloc] init];
}
```

d. Go or add the applicationWillTerminate protocol method of the UIApplication delegate and add in the body of this method following statement:
[clientAgent close];

```
- (void)applicationWillTerminate:(UIApplication *)application {
    [ clientAgent close];
}
```

e. Go or add the applicationDidBecomeActive protocol method of the UIApplication delegate and add in the body of this method following statement:

[clientAgent resume];

```
- (void)applicationDidBecomeActive:(UIApplication *)application {
    [ clientAgent resume];
}
```

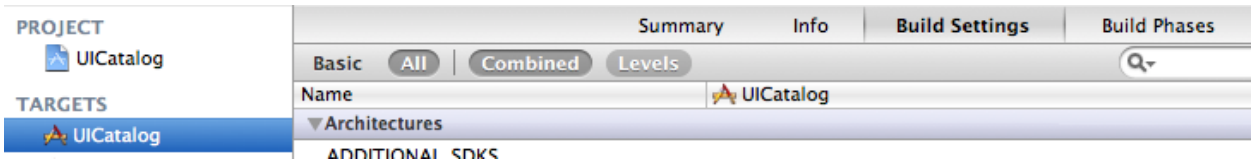
f. Go or add the applicationDidEnterBackground: protocol method of the UIApplication delegate and addin the body of this method following statement:
[clientAgent pause];

4. If you want to launch the application under test from the M-eux Agent, then you implement in the application delegate class the method application: handleOpenURL. No specific actions need to be specified in the body:

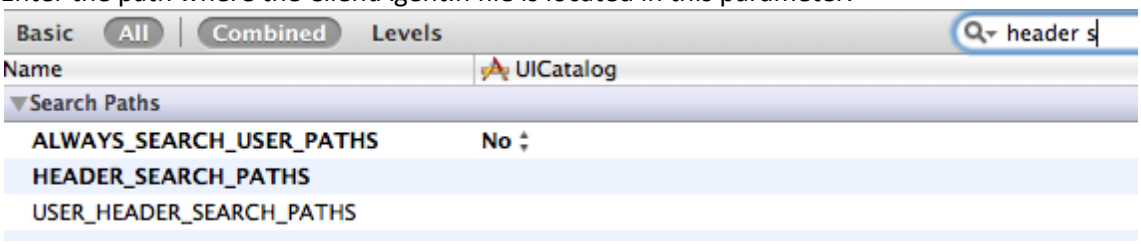
```
- ( BOOL) application ( UIApplication *) application handleOpenURL:( NSURL *) url {
        return YES;

}
```

Next you open the info.plist file of your project. Add a row and select the “URL types” as key for this row. Expand “Item 1” and provide a value for the URL identifier. This can be any value, but the convention is to use a “reverse domain name” (example: “com.myapp”). Add a new row to “Item 1”. Select “URL Schemes” as the key. Enter the characters that will become as your URL scheme, for example: “UICatalog”. This name will be used to launch the application from the script.

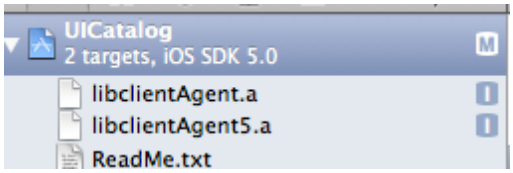
5. Select the project in the project navigator. Select the target and select the Build Settings tab.



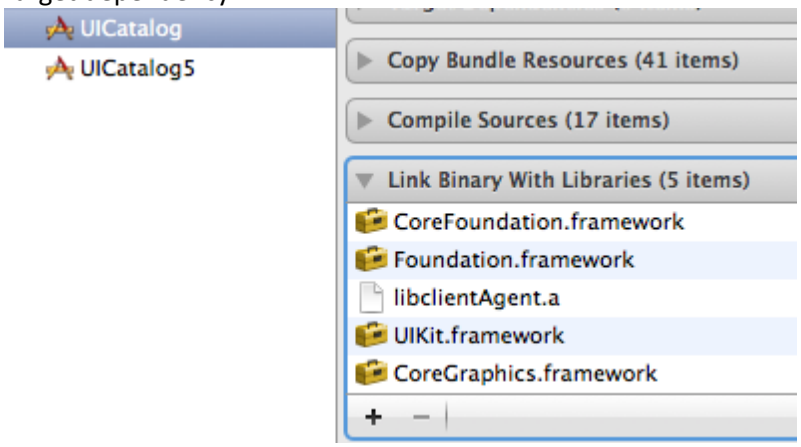
6. Type into the search edit field header search. You will see the HEADER_SEARCH_PATH setting. Enter the path where the ClientAgent.h file is located in this parameter.



7. Drag and drop the libClientAgent.a file for iOS4.x or the libClientAgent5.a file for iOS5.x in the XCode project. Confirm/select the targets that will make use of the file.



Target dependency:



8. Rebuild your application.

4.4. Remote Device Screen (RDS)

The RDS is supported for all iOS devices. One can use the RDS by connecting the device through Wi-Fi or USB mode. How to connect device to Device Manager.

You can now launch the RDS by clicking on the device name under the Tools > Remote Device Screen > Device name as shown below

The RDS will be launched as below:

There is an option to re-size (Go to Size >> Select 25%, 50% or100 %) the RDS for the better view in your PC. Spy feature is explained in page 33 or follow the link to know. How to Spy.

The RDS can be stopped by deselecting the device name in Remote Device Screen menu.

Note: Due to the restriction from apple, it is only possible to launch the RDS when the app contains our library is on the foreground (agent/app under test).

Chapter 5: Your first script

M-eux Test extends three well known environments to create your scripts: Quick Test Professional /UFT from Hewlett Packard and Visual Studio from Microsoft and Eclipse. Detailed information on both QTP/UFT and Visual Studio environments can be found in the guide: "User's Guide – QTP" or "User's Guide Visual Studio".

In this document we only explain the creation of the first script in Eclipse.

As stated in page 4 Pre-requisite we support Eclipse multiple version for 32 bit only, you can download it from http://www.eclipse.org/downloads/

5.1. Overview

We start creating our first script by using the M-eux recording functionality. This allows us to quickly create a script, which we can customize later on.

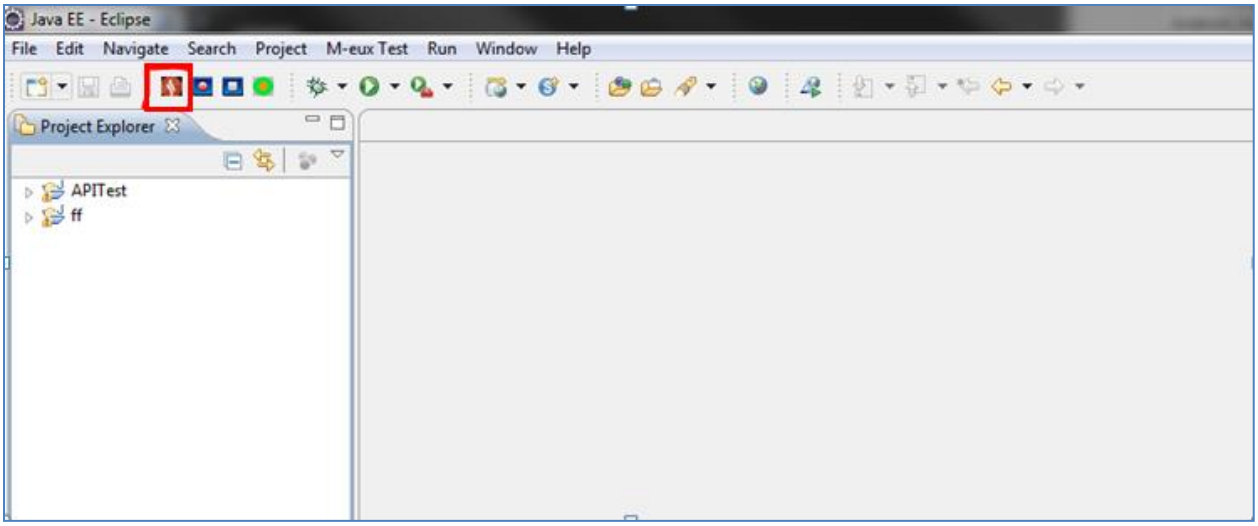
Please note that you can only record on the application that is testable. To make the app testable for a normal real device, please check the section: How to make your application testable. For simulator, please check here: How to make application testable on Simulator.

In this example we will use the UICatalog application for the testing, it is a sample application which has different objects to test.

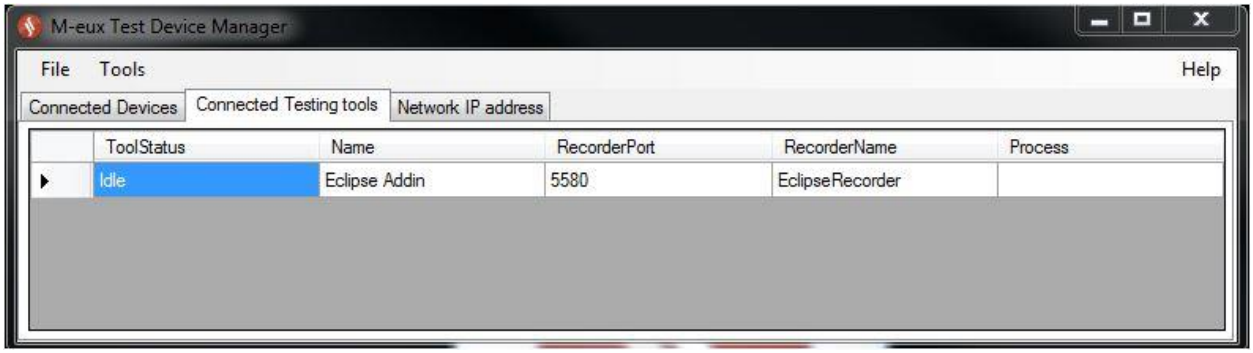
5.2. Connecting your application on the iOS device and Eclipse

1. On your PC, start the **device manager** and **Eclipse** using MeuxEclipseExecutor.exe on your desktop.
2. Connect the device with the device manager, make sure your device appears in the device manager like in the below picture: How to connect device with device manager.

3. Connect to device manager in Eclipse by clicking **Connect** as shown in following picture in red circle, or click **Connect** under **M-eux Test** menu.

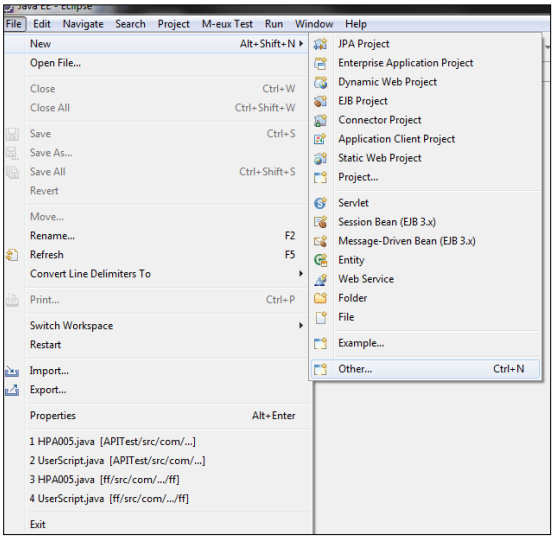


4. Make sure your Eclipse is displayed in the connected test tools tab in the device manager:

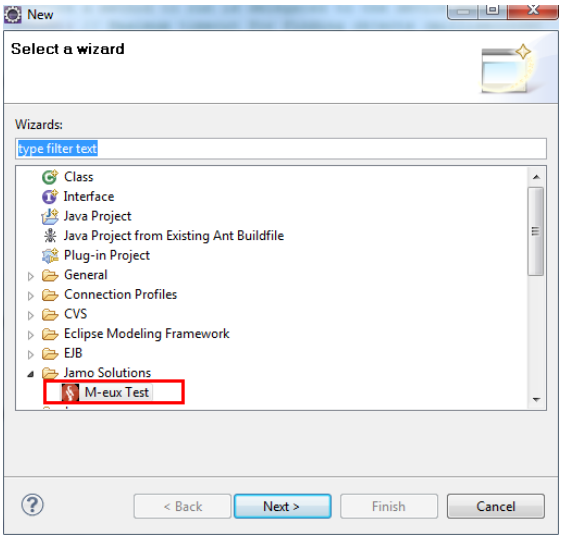


5.3. Creating a test project in Eclipse

1. Create a test project in Eclipse, by clicking **File**, **New**, **Other**.



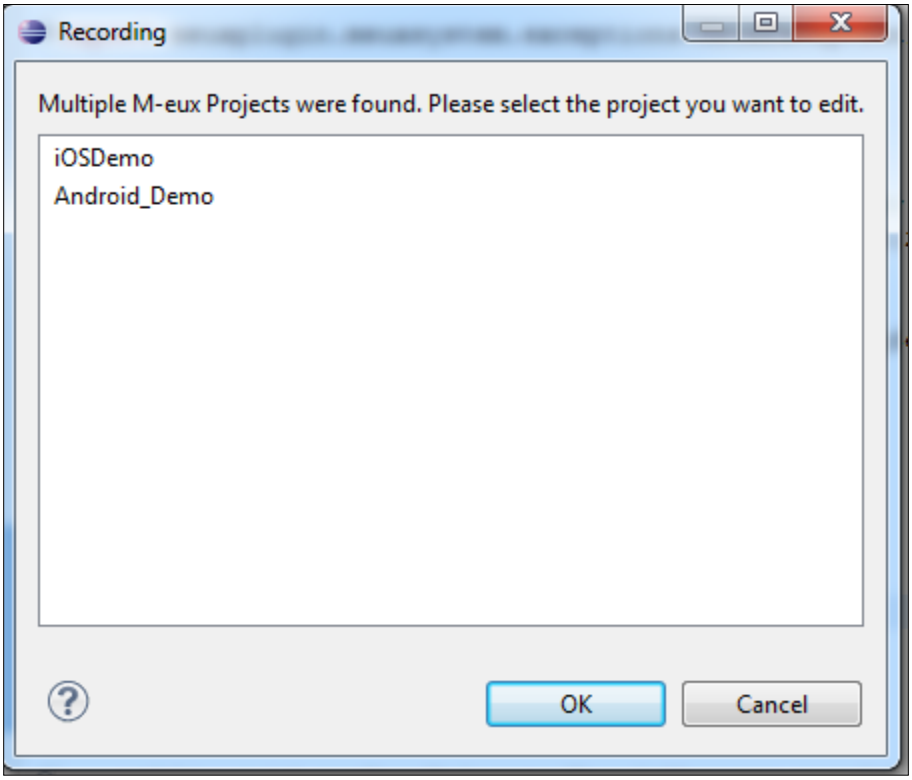
2. Then choose **M-eux Test** in the dialog box. Give the project name a name, such as **iOSDemo** and click finish.



5.4. Record your application by executing a set of actions.

1. Open the UserScript.java in your new project and you will find a runCore() method in UserScript class. Your actions will be recorded in this method.

2. Launch the UICatalog application, Press the start recording button on the Eclipse  and start recording. If you have multiple Test projects in your Eclipse workspace you will get a dialog to choose which project to record in. Just choose your target project **iOSDemo**.



Most of the action on the UI element inside the testable application will be record, below is the example of the recording on UICatalog application. After you click on the start recording button, the below script will be generated in the Eclipse under runCore() method. If you have any problem with generating script, please consult our FAQ document which can be found on our website download page (Member area >> Help Documents) or email to support@jamosolutions.com .

Screen 1 **Screen 2**

Tap on Controls in Screen1.

iOS6_1.ios_uICatalog6_1.ios_tableView_UICatalog.select("#0", "#1");

Switch on "Standard Switch" iosSwitch object as shown in Screen 2.

iOS6_1.ios_uICatalog6_1.ios_tableView_Controls.ios_standard_Switch.ios_uISwitch.set("true");

tap on the "back" button.

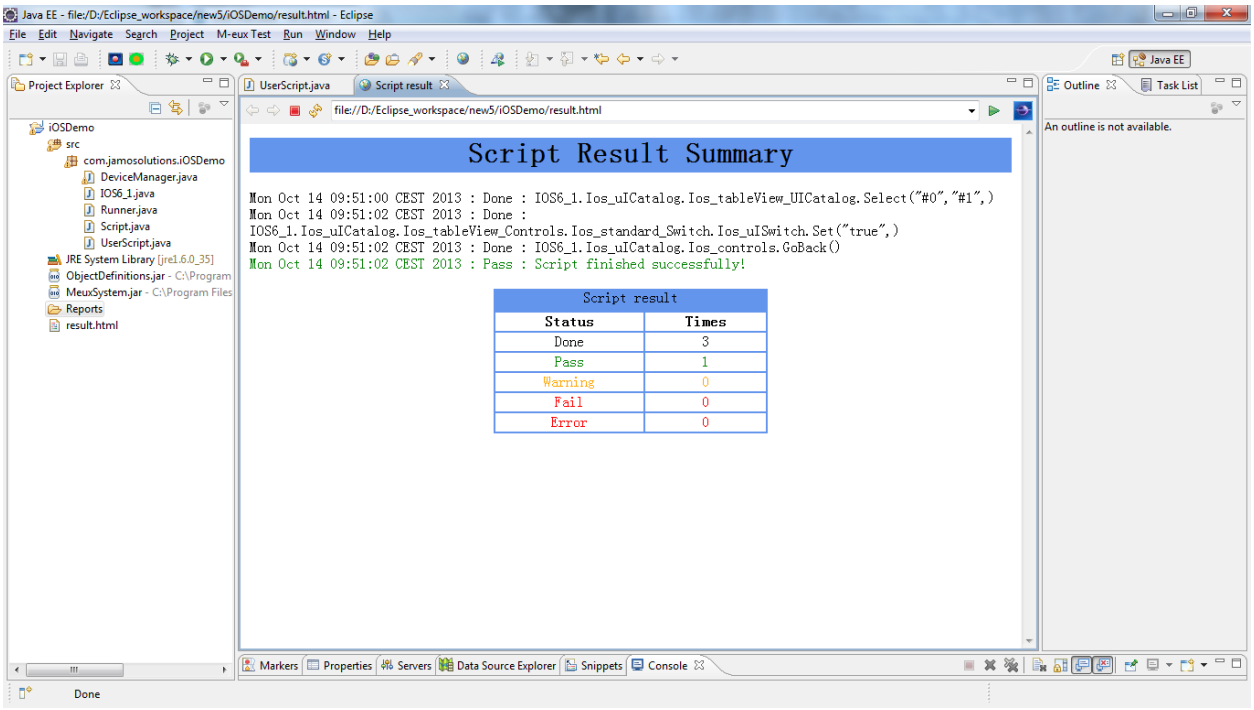
iOS6_1.ios_uICatalog6_1.ios_controls.goBack();

5.5. Replay a recorded script

1. Before you replay you need to set up application to initial state (see Screen1) so our tool can replay according to generated script.
2. Replay the recorded script by running the project as java application. To do so, right click on the Project **iOSDemo** and select **Run As** and **Java Application**
3. Your test script will now be executed on the iOS device

5.6. Viewing the test results

4. When the script has finished executing, refresh you project by pressing F5 on the project **iOSDemo**
5. You can find replay report in your project/**Reports** folder, it is in html type and in Eclipse you can open with web browser and view it.



Chapter 6: M-eux Test best practices on test automation of Eclipse

6.1. Adding check point/validation point

Since M-eux Test uses object based technology to identify the object, it is possible to access all the property of the objects. You can check the M-eux function references (Open **M-eux device manager** – **help** – **function reference**) for the property of the objects.

To retrieve the runtime object property, you can use the function:

Object.getRoproperty(propertyName).value;

To validate a certain property, you can use the function:

Object.checkProperty(propertyName, expected value);

After running the above script, the report file will generate the result pass/fail.

6.2. Access Result Manager

If you want to write some customized report in the result file, you can use the result manager object to create the script:

this.getResultManager().logDone(message);

Besides logDone, you can use logPass, logFail and so on, check how it will look like.

6.3. Scripts needs to be written manually

Note: some script cannot be recorded, but you can add the scripts in order to make navigate the application.

a. The launching of the application will not be recorded:
   You can use our applaunch method to launch one application.

   object. AppLaunch (App Name)
   Device.applaunch ("UICatalog");

b. To close the application, you can use the .kill method
   Device.window.Kill ();
c. To swipe from one point to another point, Zoom to specified rectangle, Scroll to the specified rectangle,HoldTap,HoldPush...etc
   - holdTap() - Touch the object and hold for the specified milliseconds.
     Object. HoldTap x, y, holdTime
   - tapMove() - tap move (swipe) from one point to another according to specified coordinates.

Object.tapmove x1,y1,x2,y2,1

- scrollToRect() - Scroll so that the specified rectangle gets visible
  Object.ScrollToRect (x, y, width, height)
- zoomByScale() - Zoom to the specified scale
  Object.ZoomByScale (scale)
- zoomToRect() - zoom to the specified rectangle
  Object.ZoomToRect (x, y, width, height)

d. There are some buil-in methods implemented on iOS Mobile device object to retrieve Battery info,Memory info,DeviceInfo..etc. Please refer function reference document for more details **(Go to : Device Manager >> Help >>function reference >> Non -GUI Test Objects >>IOS MobileDevice)**

6.4. Description property configuration

Our solution is based on object recognition. To recognize an object on the mobile device, we compare the selected properties (description properties) in the object repository with the ones on the device. The combination of the description properties need to identify the objects uniquely. The combination of the description properties might vary depending on your own application, so in some cases, you will need to configure the selected properties yourself. To do this, please follow the below instruction:

Go to **M-eux Device Manager** – **Tools** – **Object Configuration**, then you can choose your object type and configure the description property. Note after configuration, you will need to restart the M-eux Device Manager.

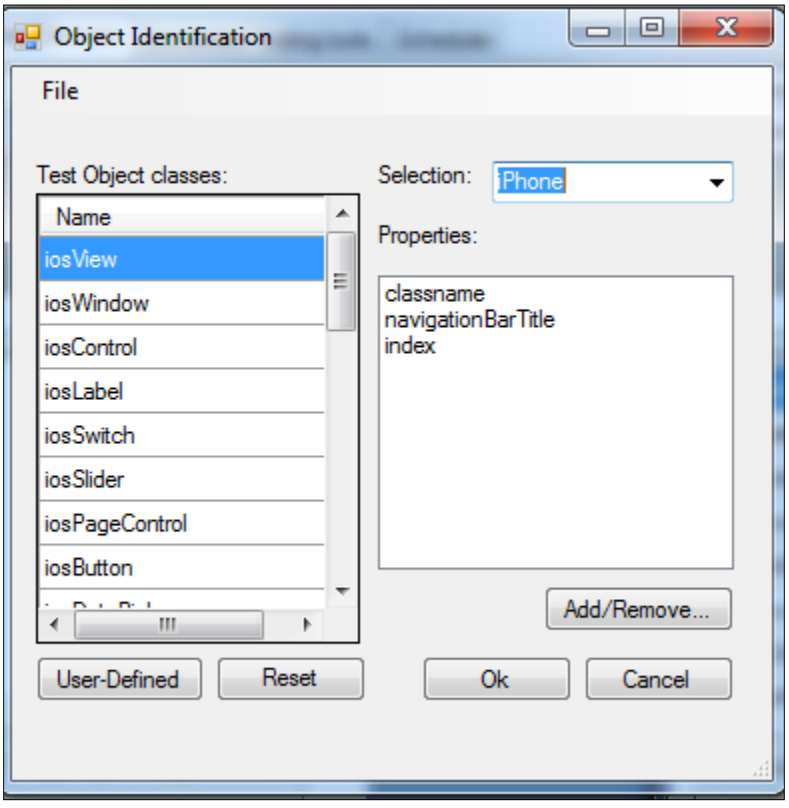


Figure 16 Object configuration from Device Manager

6.5. Object repository

In Eclipse, the object repository is saved in a Java file called yourDevicename.java where all the objects are defined as a Java class. Inside the class, the description properties exist as fields and you can set value to those description properties. The first time after recording, you will find an extra file named DeviceName.java, that is object repository of your project. Check your YourDeviceName.cs file and have a look at the object repository/pool.

On Eclipse, very often you will need to check the object repository in order to create script, after recording/spy+add/learn GUI, there will always be new objects added in the object repository unless the object is already in the repository. As in the below picture, you can see the generated Java ode about repository.

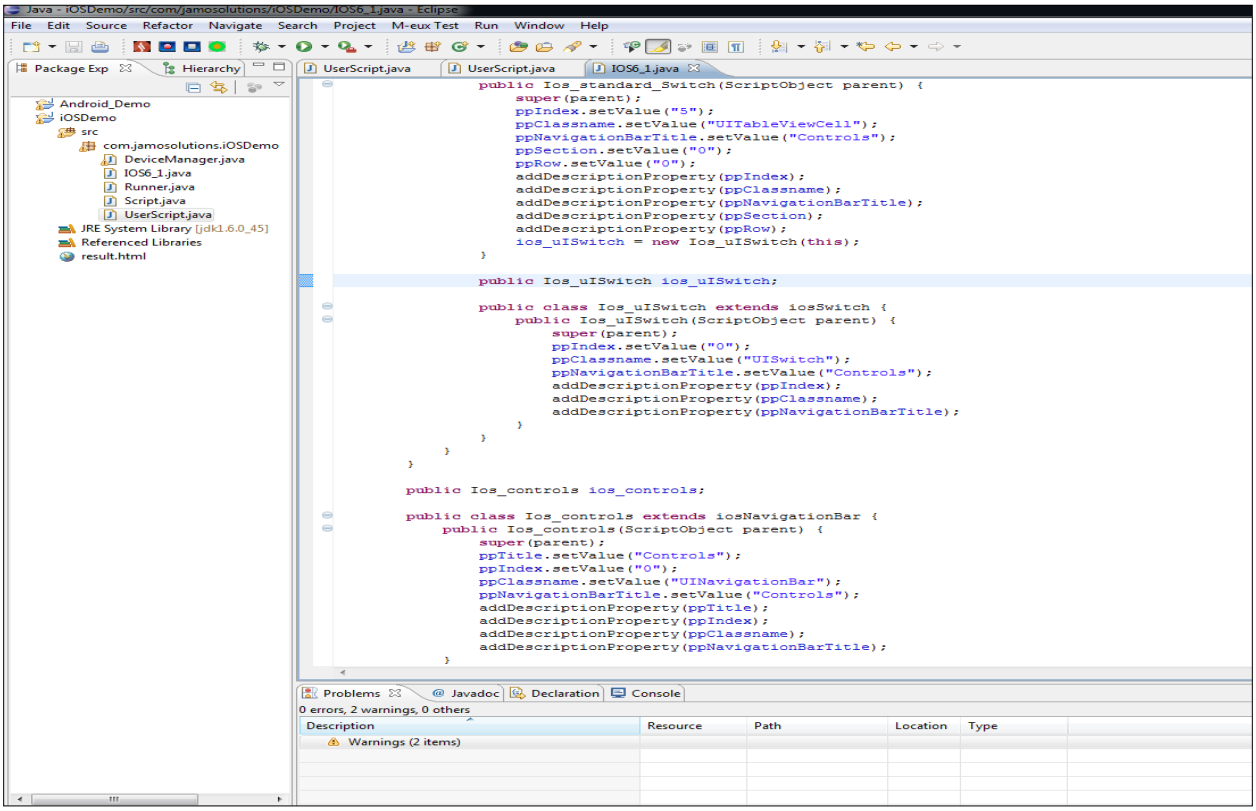


Figure 17 the Eclipse object repository

To have a better graphical view of the objects, you can open with the object pool with **MeuxObjectPoolEditor** to view hierarchy of UI elements.

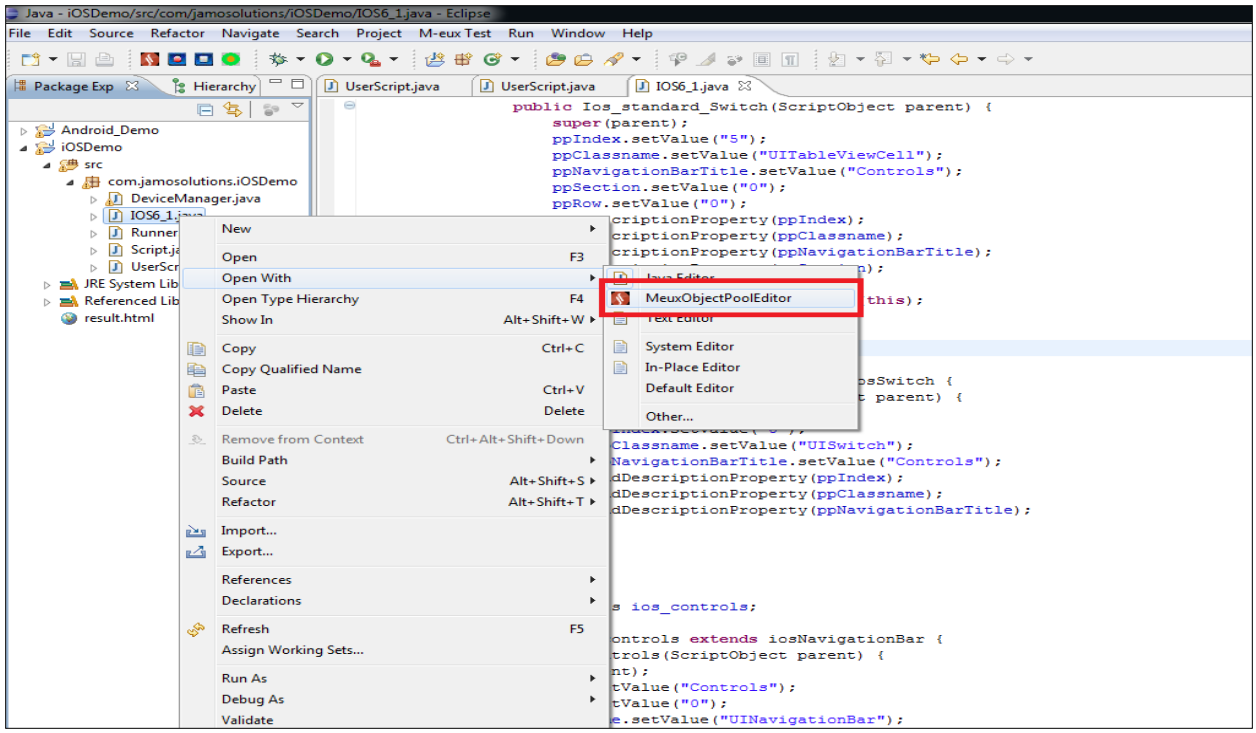
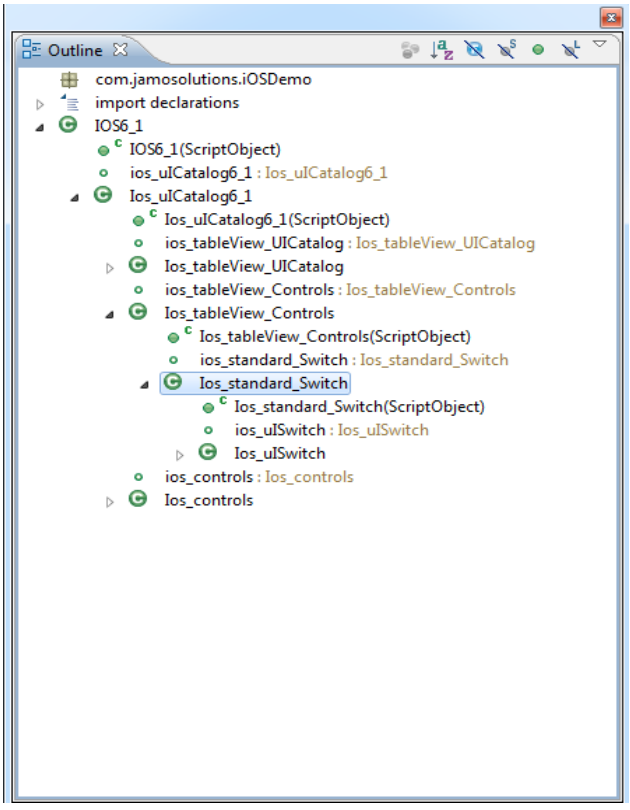


Figure 18 Object poll in user interface

Click on the object pool tab then you can see the hierarchy of the objects and the description property in a clearer way.



6.6. Highlight functionality in Eclipse

You can also highlight the objects from the Eclipse object pool. When troubleshooting if an object can be correctly identified, it is very useful to use the highlight functionality in Eclipse, please follow the below instructions:

1. Make sure you have the package explorer in the project navigator, you can do this by going to Eclipse—Window—Open perspective – (other)—Java, then on the left hand you will see the package explorer:

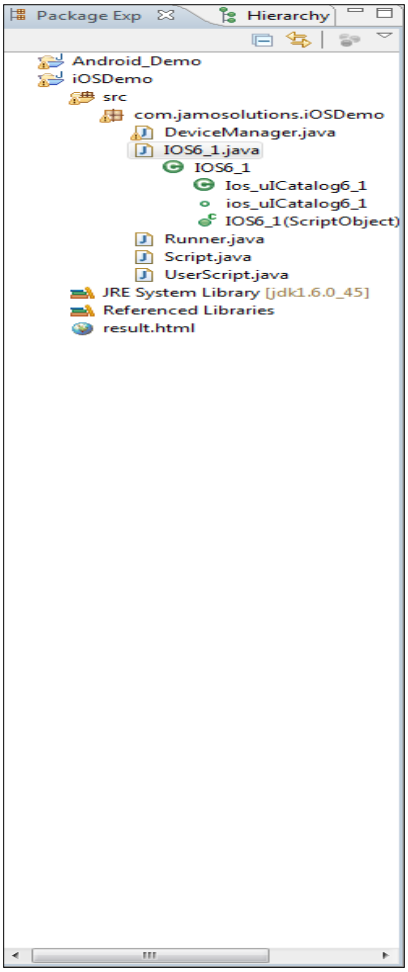


Figure 19 Eclipse package explorer view

2. Expand the object pool and then right click on the object, you will find the last option is called highlight:

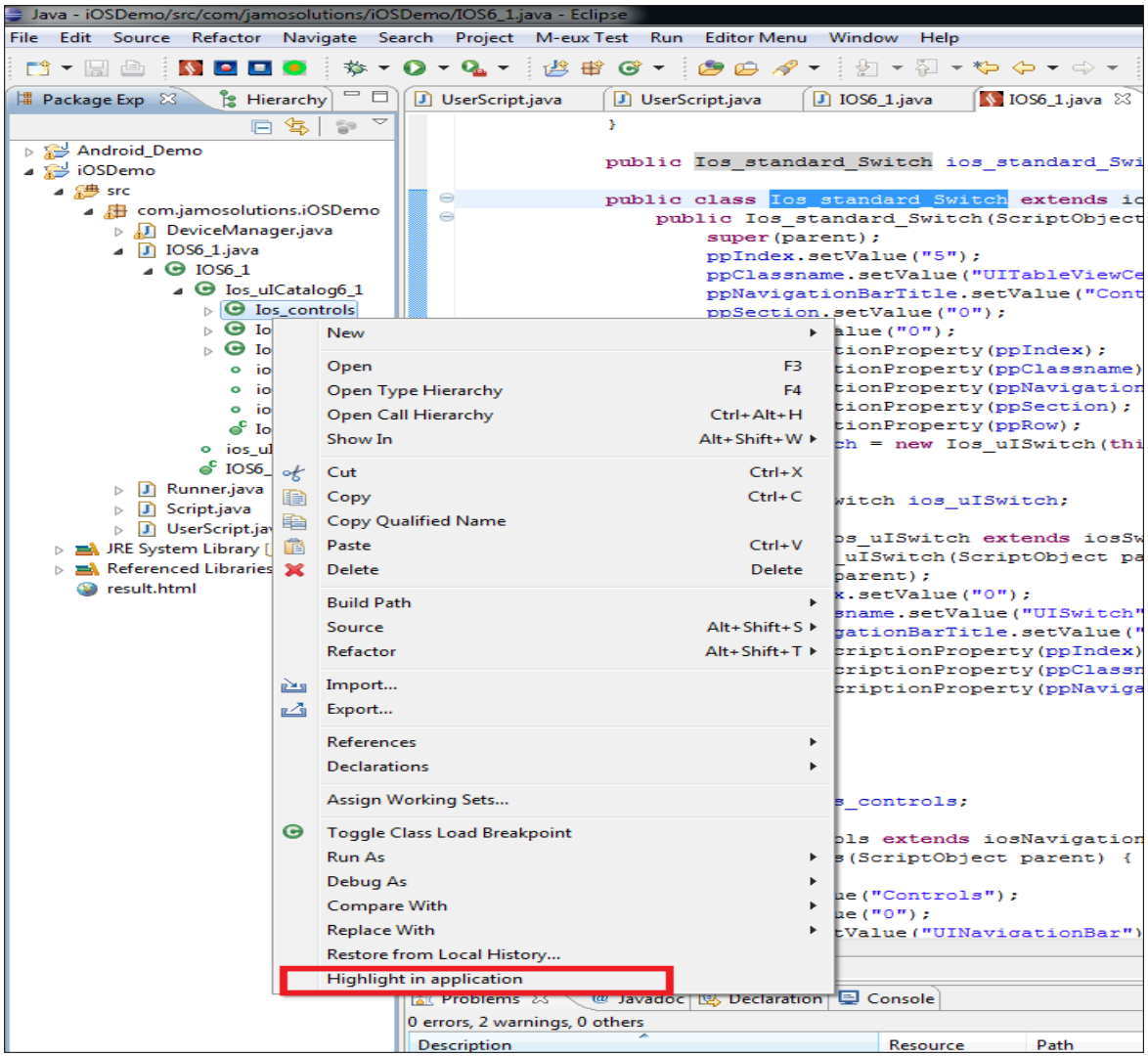


Figure 20 Highlight in Eclipse

6.7. Spy

The Spy function is very useful when you need to check one specific object type or description property or add the object in the Eclipse project repository. You need to open the RDS (Check remote device screen) in order to spy and add the object in the repository. You can open the **Spy** menu from the remote device screen and add objects to the repository in Eclipse.

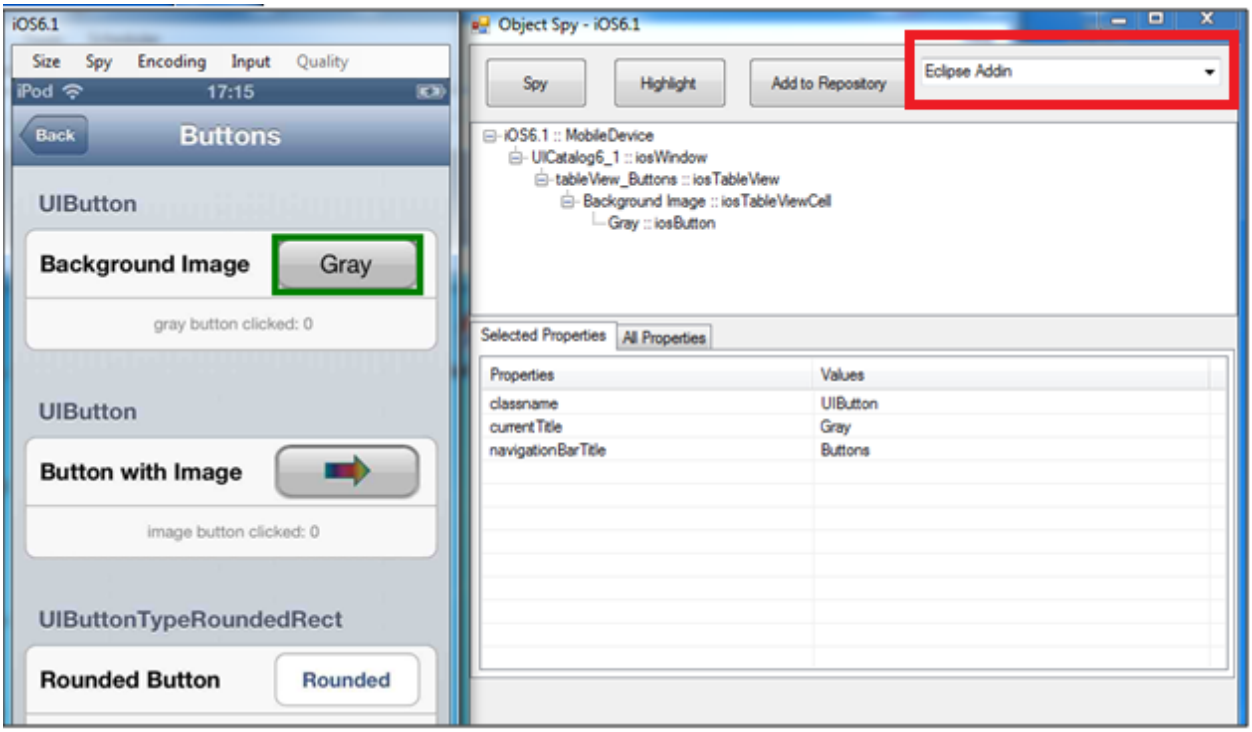


Figure 21 The Spy functionality and add an object to repository from the spy window

You can select Eclipse in the drop down menu on right side of the button "Add to Repository" and click this button, then corresponding code of selected object element and its parents will be added to the Eclipse project.

Moreover, you can see the Highlight button, when you select any object in the objects hierarchy inside the spy window and then click Highlight, there will be a frame showing where the object is located in this object hierarchy.

6.8. Learn GUI

If you want to add all the UI elements on the current window to the object repository, you can use **Learn GUI** function from us.

Simply go to the **Eclipse**, click on the learn GUI button  and start to learn all the objects in the object pool.

6.9. Replay settings

In your UserScript.java you can find these 2 lines of code.

```
getMeuxReplaySettings().setDelay(0); // The Delay can be configured for the project between every script, in milliseconds
getMeuxReplaySettings().setFindObjectTimeOut(20000); // Maximum timeout for finding objects (milliseconds)
```

As the comment explains, we can set time parameters for replay.

6.10. Regular expression

Whenever you have a string as an argument in the command, for example

iOS6_1.ios_uICatalog6_1.ios_tableView_UICatalog.`select("Controls");`

For this string argument, you can use regular expression. When using regular expression, use a "!" sign in front to indicate the string is with regular expression enabled, so:

iOS6_1.ios_uICatalog6_1.ios_tableView_UICatalog.`select("!.*Controls");`

or

iOS6_1.ios_uICatalog6_1.ios_tableView_UICatalog.`select("!.*ontrol.*");` and so on.

For more information about advanced usage of M-eux tool, you can refer to our FAQ document.

6.11. How to run same Script on multiple devices

Suppose you are given a task where you have to automate some test cases for an application. When you are finished automating the application, you have to use same script to execute multiple mobile devices.

- The scripts should run as intended without any issues.
- The code and the flows should be easy to understand.
- The scripts should be easy to maintain in case of any changes.

6.11.1. What to do

Write generic code which run for different devices. Including looping statements to repeat test execution for the multiple devices connected.

Framework should ensure that test is exited as soon test is failed, since there is no point in going forward with execution.

To ensure proper exit from Eclipse test, we can either use error handling using user defined functions. In case of error handling using Java, we can exit a test at runtime in following ways.

Based on Business requirement use generic functions like LaunchApp(), Login()at the beginning of test case and Logout() , KillAPP() at the end of test case. This will help the successful execution on all devices.

6.11.2. Code Example:

'script template to run same test script on multiple devices

```
String newName = deviceManager.getDevice("0", "true");
android.setTOProperty("name", newName);
```

Chapter 7: Summary

Now you should be able to start with the testing on iOS using M-eux Test and Eclipse.

We wish you a good journey with your test adventure using M-eux Test tool. If you are facing any issues with our tool, or having trouble about making your test scripts for your test cases, you can always contact us at support@jamosolutions.com.