

# StormTest Mobile Device Testing

---

*Controlling mobile devices from within StormTest Development Center*

Document ID: ST-14023

Revision Date: December 2015

Product Version: 3.3

MPS Version: 1.3

Web: <http://www.accenturestormtest.com>

The contents of this document are owned or controlled by Accenture and are protected under applicable copyright and/or trademark laws. The contents of this document may only be used or copied in accordance with a written contract with Accenture or with the express written permission of Accenture.

## Contents

<b>1</b>	<b>Preface.....</b>	<b>5</b>
1.1	StormTest.....	5
1.2	About This Document .....	5
1.3	Who Should Read This Document .....	5
1.4	Related Documentation.....	5
1.5	Definitions and Acronyms.....	5
1.6	Prerequisites .....	6
<b>2</b>	<b>Introduction .....</b>	<b>7</b>
2.1	Adding the Mobile Test Capability.....	7
2.2	Architecture .....	7
2.3	App Control.....	8
2.4	Restrictions .....	8
<b>3</b>	<b>Setting up an App for test .....</b>	<b>9</b>
3.1	iOS.....	9
3.1.1	Resigning.....	9
3.1.2	Setup the Device .....	12
3.2	Android .....	13
3.2.1	Setup the Device .....	13
<b>4</b>	<b>Controlling the App .....</b>	<b>15</b>
4.1	Configure Slot.....	15
4.2	Valid Commands .....	15
4.2.1	Setting the current live mechanism (iOS DUTs only).....	16
4.2.2	START-ANDROID/START-IOS/START-BT .....	16
4.2.3	STOP-ANDROID/STOP-IOS.....	17
4.2.4	TAP:<X>:<Y>[:count[:<DURATION>]] .....	17
4.2.5	SWIPE:<X1>:<Y1>:<X2>:<Y2>[:TIME] .....	17
4.2.6	PINCH:<x1>:<y1>:<x2>:<y2>: <x1'>:<y1'>:<x2'>:<y2'>.....	17
4.2.7	LOCK:<DURATION> .....	18
4.2.8	SHAKE.....	18
4.2.9	VOLUME:UP DOWN:<DURATION>.....	18
4.2.10	BACKGROUND:<DURATION> .....	18
4.2.11	ORIENTATION:<DIRECTION>.....	18
4.2.12	TAPELEMENT:desc text index resourceId:element.....	18

4.2.13	RAW:<javascript> .....	19
4.2.14	SENDTEXT:<string>.....	19
4.2.15	PRESS:<key> .....	19
4.3	Discovering UI Elements .....	19
4.3.1	Android.....	19
4.3.2	iOS .....	20
<b>5</b>	<b>Bluetooth Control of Android Devices .....</b>	<b>23</b>
5.1	Setting up the Mac Mini.....	23
5.1.1	Enter the Bluetooth settings application on the Mac Mini .....	23
5.1.2	Turn on Bluetooth.....	23
5.1.3	Verify that Bluetooth is On .....	24
5.1.4	Verify the Mac is visible to other devices .....	24
5.2	Connecting the Device to Stormtest.....	24
5.2.1	Configure the DUT.....	25
5.3	Initiating pairing .....	25
5.3.1	Launch the Pairing Application on the Stormtest Server .....	25
5.3.2	Reserving the Slot for pairing.....	26
5.3.3	Initiating pairing on an Android Device .....	27
5.4	Important things to remember.....	29
<b>6</b>	<b>Writing Tests for an App .....</b>	<b>30</b>
6.1	Creating the Control Files for UI Support.....	30
6.1.1	RC Buttons.....	30
6.1.2	Associate RC Buttons with DUT model .....	31
6.1.3	Map File.....	32
6.2	Navigator.....	34
6.3	Remote Control Skin .....	34
<b>7</b>	<b>Troubleshooting .....</b>	<b>35</b>
7.1	FAQ.....	35
7.1.1	Why don't I see my app in the remote control video? .....	35
7.1.2	When I open the video to the slot I see a 'dongle not found' error .....	35
7.1.3	Can I connect two devices at once to a slot?.....	35
7.1.4	Can I test more than one app?.....	36
7.1.5	Does the MPS have to be physically in the slot? .....	36
7.1.6	Can I test audio and video in my app? .....	36



## 1 Preface

### 1.1 Accenture StormTest

Accenture StormTest Development Center is the leading automated test solution for digital TV services. It is designed to reduce the cost of getting high quality digital TV services to market faster.

StormTest Development Center greatly reduces the need for time-consuming, expensive and error-prone manual testing and replaces it with a more accurate and cost-effective alternative. It scales easily to large numbers and types of devices and integrates with existing infrastructure to give much greater efficiency in testing. It can be used to verify and validate services on a virtually every piece of consumer premises equipment (CPE), from set-top boxes to games consoles and from iPads to Smart TVs. It has been specifically designed to meet the needs of developers and testers of these CPE devices and the applications which run on them.

### 1.2 About This Document

This document describes how to use the Mobile Application Testing Solution for StormTest Development Center.

### 1.3 Who Should Read This Document

This document is aimed at test developers that wish to create StormTest tests for mobile applications on both iOS and Android operating systems

### 1.4 Related Documentation

The StormTest user documentation set comprises of the following documents:

- 1) StormTest Developer Suite User's Manual
- 2) StormTest Programmer's Guide
- 3) StormTest Client API
- 4) StormTest Hardware Installation Guide
- 5) StormTest Software Installation Guide
- 6) StormTest Server Monitor User's Manual
- 7) StormTest Administration Console User's Guide
- 8) StormTest Administration Tools User's Guide

The latest version of these documents can always be found on our support website, in the "StormTest Development Center Documentation Set" section: <https://stormtest.zendesk.com/hc/en-us/sections/115000203169-StormTest-Development-Center-Documentation-Set>

### 1.5 Definitions and Acronyms

Acronym	Description
MPS	Mobile Proxy Server
SPS	Slave Proxy Script
IR	Infra-red

STB	Set-top box
DUT	Device Under Test
AUT	Application Under Test
UDID	Unique Device Identifier

Table 1 – Definitions and Acronyms

## 1.6 Prerequisites

The following prerequisites are required. Please contact the StormTest support team to ensure that all the necessary configuration is done correctly and you are using the correct version of StormTest

- Mobile device control is made possible by the OEM IR plugin feature; this feature is available in StormTest Development Center version 3.2 or greater.
- A Mac mini will be required to host the MPS service for each slot
- The IP address of the Mac mini Server must be configured in the admin console when configuring the OEM IR Plugin
- The IP address of the Mac mini must be contactable from the StormTest server. It does not have to be on the internal StormTest LAN. In fact, it will be easier to have the Mac mini on a LAN with an internet connection to allow for easy update of tools such as Xcode and ADB.
- The UDID of iOS devices used for testing must be configured in the StormTest Admin Console for each slot containing an iOS device.

## 2 Introduction

The StormTest automated test system provides out of the box support for the testing of any device that can be controlled using an IR protocol, such as set-top boxes (STBs) and Smart TVs. However, as the market for digital video services has changed, an increasing number of consumers are viewing their video content on mobile devices such as iPads or Android Tablets. There is, therefore, a growing demand for the ability to create test scripts that can drive and validate applications on these devices that deliver video services.

In version 3.2 of the product, Accenture StormTest introduced a mechanism to enable the test developer to do this. This document will enable the test developer to get up and running with their app and develop powerful and portable test scripts.

### 2.1 Adding the Mobile Test Capability

To enable the testing of a mobile application, one or more slots in a StormTest server must be upgraded to have the mobile testing capability.

This upgrade consists of adding a mobile proxy server (MPS) to the slot which provides the interface to the device under test (DUT). This MPS both enables control of the app under test (AUT) and also enables validation of each of the test steps.

The MPS is an Apple Mac Mini device, which is either supplied by the customer or by Accenture. This Mac Mini must have a defined set of software installed on it, including the StormTest MPS service. It must also be licensed by Accenture. For the purposes of this document, we assume that the Mac Mini has all software installed and is properly configured. For more information on this process, please see the document: ST-14022: "Setting up a Mobile Proxy Server".

### 2.2 Architecture

The architecture of the solution can be seen below:

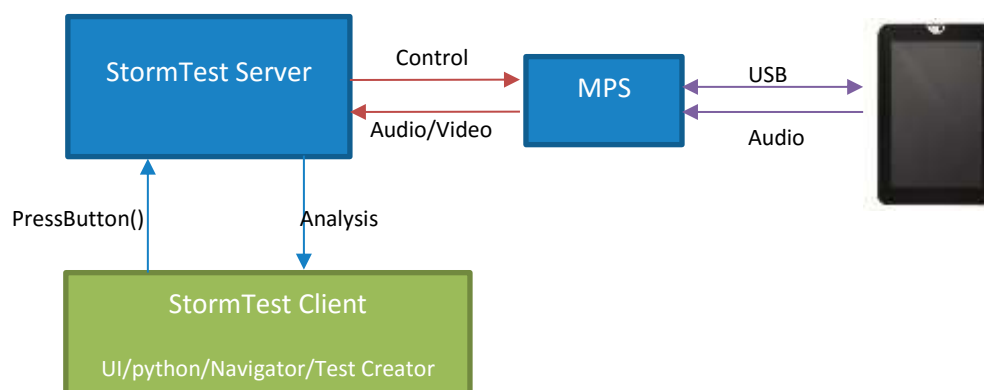


Figure 1 : Mobile Testing Architecture

The StormTest client uses the standard control APIs (such as `PressButton()`) to send control commands to the AUT. The MPS can then translate these commands into native commands according to the OS being used by the application.

Validation of each step is done using the standard StormTest audio and video analysis capability. This includes the StormTest Navigator feature and supports test scripts written in both python and Test Creator.

## 2.3 App Control

The StormTest mobile testing solution supports the following types of control:

- Start app
- Stop app
- Tap at specific co-ordinates or on specific UI element
- Swipe from one location to another
- Pinch – two finger pinch movement
- Lock: locks the device for a specific duration
- Shake: emulate shake event on device
- Volume up/down
- Background: puts the AUT into the background for a specific duration
- Orientation: alter the orientation of the device between landscape/portrait and also orientated face up/down

As the interface to the MPS from a StormTest script or the Developer Suite user interface is via the standard 'IR' control mechanism, there needs to be a mapping from an IR 'button' to the relevant control request.

So, to use any of the above control mechanisms, a corresponding 'button' must be defined in StormTest. This extends to the parameterised versions of the above controls. So, tapping on two different locations on screen requires two different 'buttons' to be defined. It is possible to map different application buttons with the same name on different screens to the same StormTest button. More on this later.

The next section of this manual will deal with this aspect of setup in more detail.

## 2.4 Restrictions

The iOS operating system and the native Xcode toolset imposes some restrictions on what is possible when testing an iOS app. Effectively, when testing an iOS device, it is only possible to test a specific app that has been tagged with the correct entitlements and signed with the correct profile.

It is NOT possible to interact with any other app or to interact with the device in general. Due to the signing and entitlement requirements, it is not possible to interact with any 'system' apps or any other apps where you do not have access to a developer build.

The android OS does not impose these restrictions and when testing in this environment, it is possible to control the full device and interact with any app on the device, including system apps.



## 3 Setting up an App for test

In the case of both iOS apps and Android apps, the native toolset is used to actuate control of the app.

For iOS apps, the native toolset is Xcode Instruments. For Android apps, the toolset is the Android Debug Bridge (ADB). These tools impose some requirements that must be met before the apps can be tested.

### 3.1 iOS

In order for Xcode to be able to control the AUT, there are some steps that are necessary before testing can commence.

The application should be supplied by the application developers or vendor in either .ipa or .app format.

The application needs to be signed with a *development provisioning profile* or else instruments will not be able to control it.

If the developers are in-house, they can download the signed app to an iOS device provided to them by the test team.

If they are not in-house, the development profile should be updated to include the UDID of the physical device that will be connected to the MPS for test purposes. The application should be rebuilt using this profile and the binary passed on to the test team

If this is not possible, then the app will need to be re-signed before it can be controlled on the target device.

#### 3.1.1 Resigning

Before attempting to re-sign an app, you must ensure you have:

- A developer provisioning profile, created from the member center on the Apple developer website. When creating this profile, you must use the app ID for the app you will be testing. Download this provisioning profile as you will need it later (it will be a .mobileprovision file)
- A valid developer certificate from the above developer profile downloaded to the Mac that you are going to use to sign your app
- A .ipa or a .app file built using a development profile.

### 3.1.1.1 Use iResign tool

The easiest way to resign an app is to use the open source tool iResign. The source code can be downloaded to your mac from <https://github.com/maciekish/iResign>.



Argument 1 : The ipa file you wish to resign

Argument 2: The .mobileprovision file containing the device on which the app will be running. The signer also needs to be associated with this file.

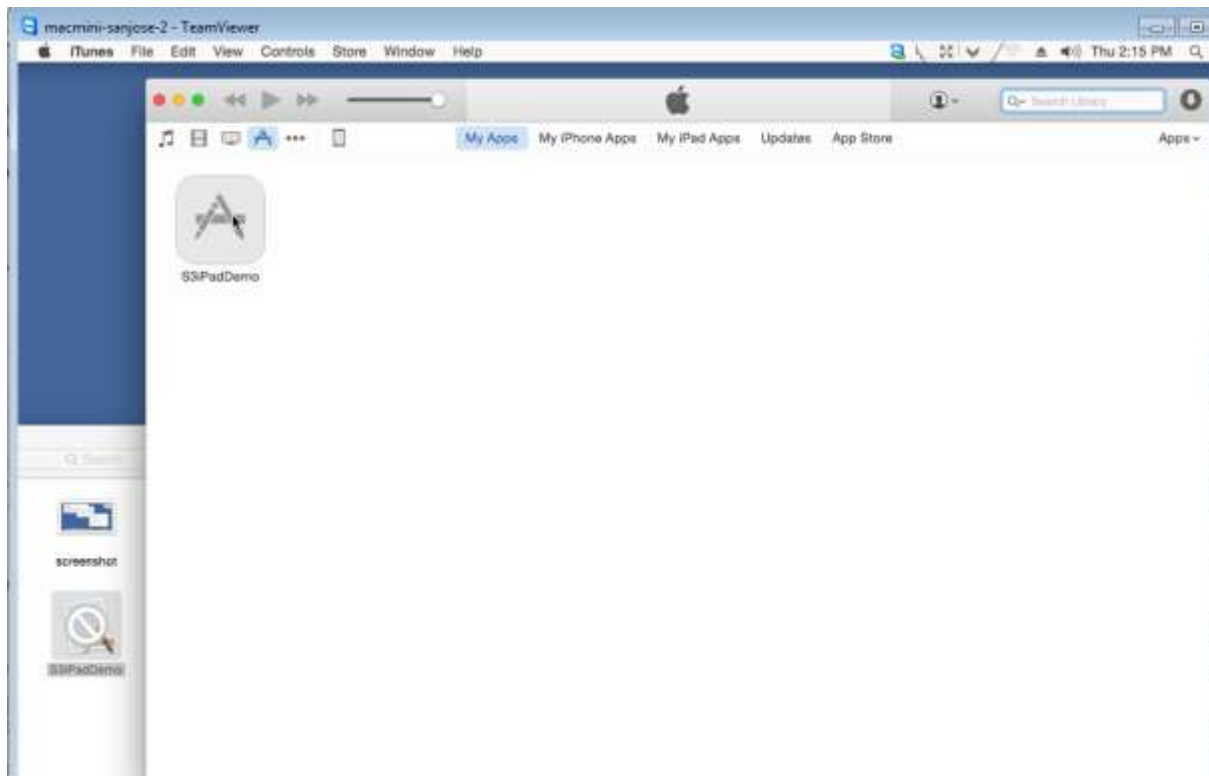
Argument 3: Leave empty – the iResign app will create an entitlements file

Argument 4: E.g. com.accenture.StormTest - You can get this string from an administrator of your apple developer group.

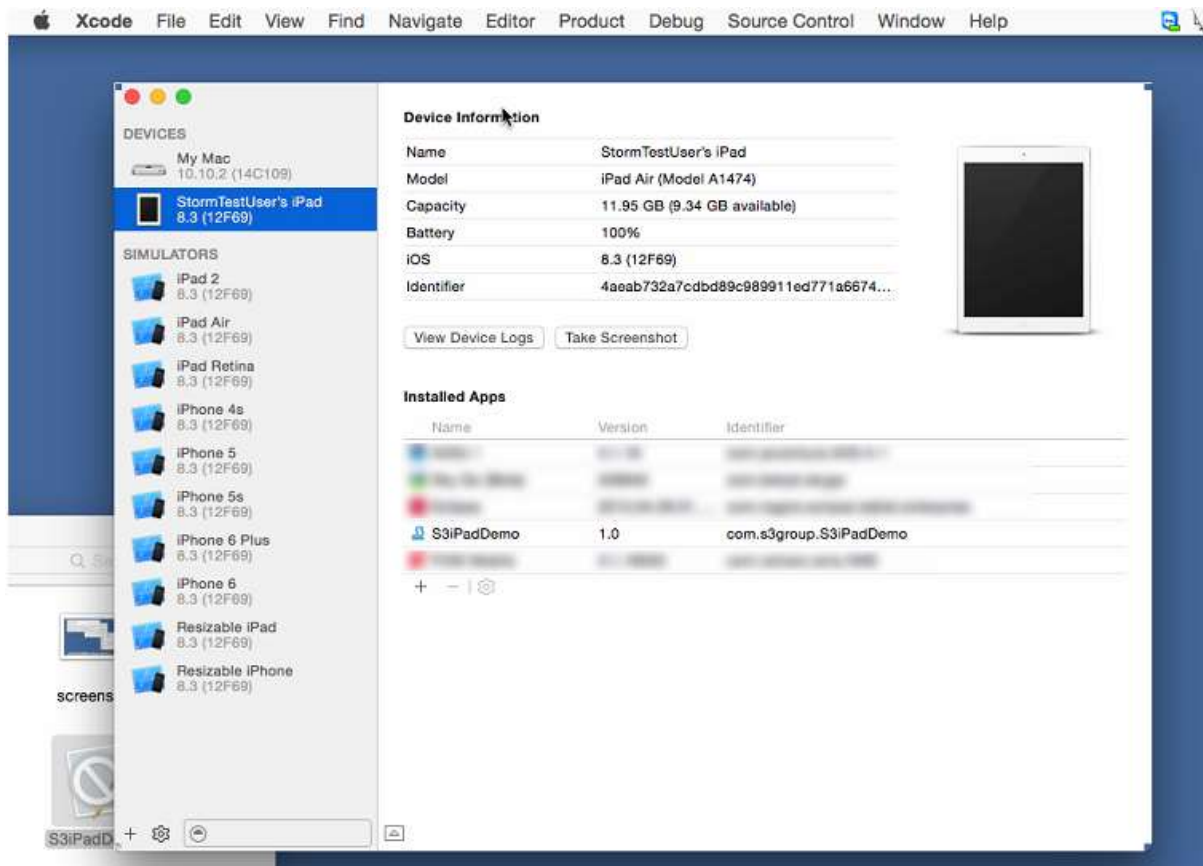
iPhone Developer: The Developer who will be re-signing this app. His/her certificate needs to be installed on the machine.

### 3.1.1.2 Deploy the app

You can use iTunes to drag the ipa to the Apps tab in iTunes. No iTunes login necessary.



You can also use XCode:

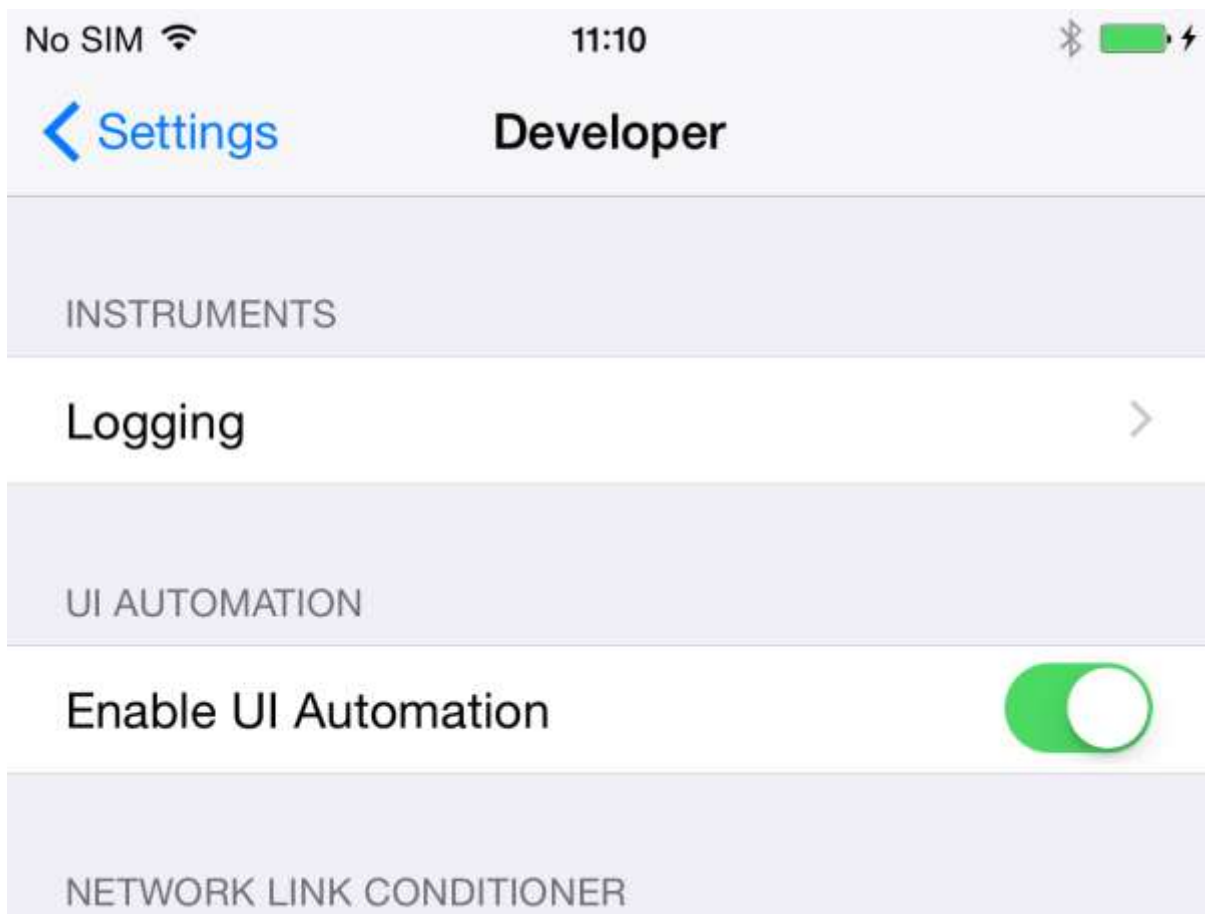


- Go to Window->Devices.
- Select the device connected to the mac.
- Click on the + button and add the .ipa you wish to download.

### 3.1.2 Setup the Device

If the iOS device that you will be using is running iOS 8 or later, then you must enable “UI Automation”.

This option can be found under **Settings > Developer > UI Automation**.



Note that the developer option will **only** appear once the device has been provisioned for development.

## 3.2 Android

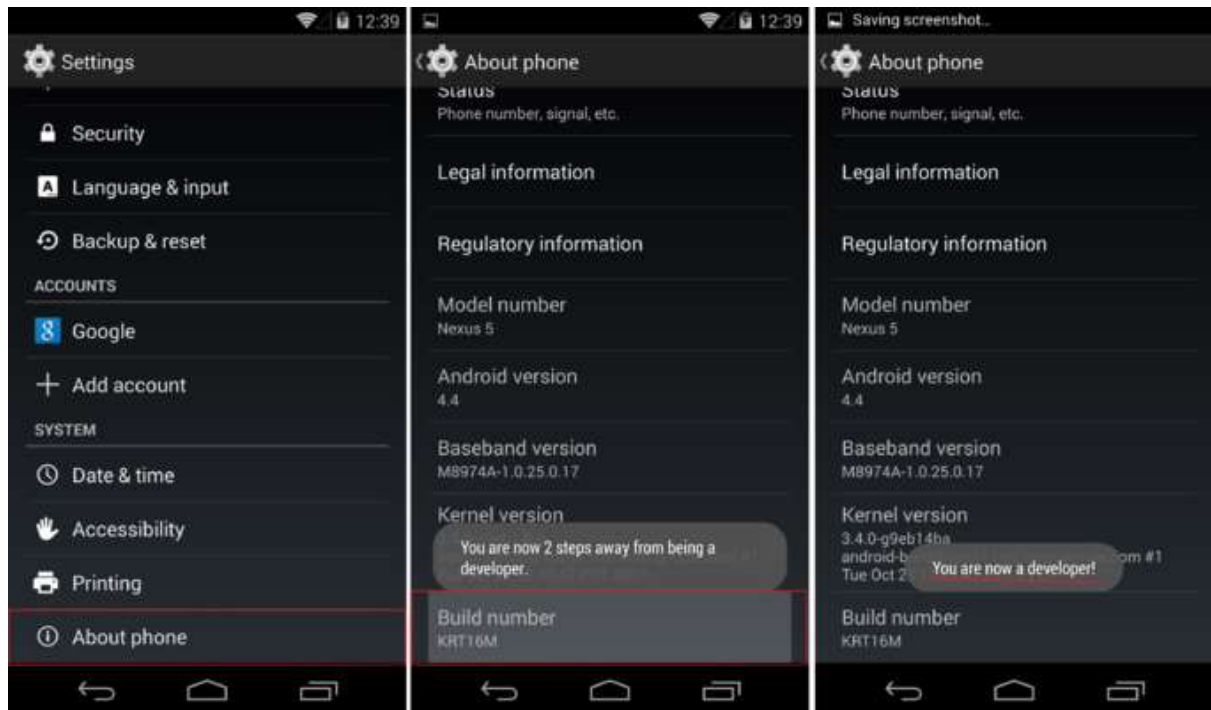
The android OS and tools do not place the same restrictions on the tester as the Apple OS. There is no need to re-sign the application or to change entitlements.

In fact, it is perfectly possible to install the app from the Play store and test the actual app deployed to customers.

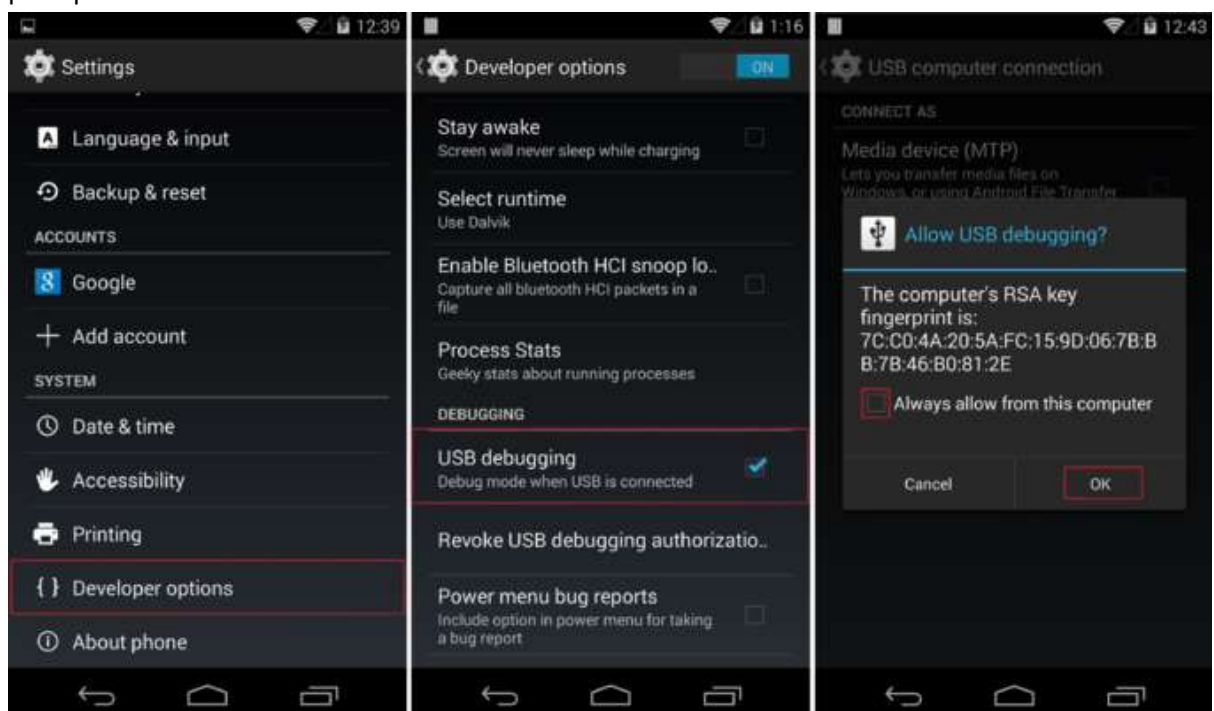
### 3.2.1 Setup the Device

On the Android device you are using for testing, you have to enable **USB Debugging**. To do this, follow these steps:

- Navigate to Settings > About Phone > scroll to the bottom > tap Build number seven (7) times. You'll get a short pop-up in the lower area of your display saying that you're now a developer.



- Go back and now access the Developer options menu, check 'USB debugging' and click OK on the prompt.



## 4 Controlling the App

To control the app, you need to send specific defined commands via the MPS using the StormTest `PressButton()` API. This API will take any string as a parameter and it is in this string that we pass the commands.

Note that there are more steps required to control the app from one of the StormTest UI tools, but in this chapter we will just concentrate on control via the python API.

### 4.1 Configure Slot

Before the app can be controlled, you must first have followed the steps in the previous chapter. Then you must configure StormTest to ensure that it knows that the slot you wish to use contains an MPS and a tablet. This is done by editing this configuration file on the StormTest server (must be done as an administrator, so run notepad as an admin to edit the file):

```
C:\Program Files (x86)\StormTest Server\oem_ir_mobile_devices.conf
```

In this file, there is an entry for all slots with an MPS. If the slot you are using is not in the file, add it here, along with the IP address of the MPS in the form `<slot_no>::mps::<host_or_ip>`. For example:

```
2::mps::10.11.12.110
```

Configures an MPS in slot 2 with IP address 10.11.12.110 (this is the IP address of the Mac mini hosting the MPS software).

The StormTest sever process will need to be restarted to pick up any change to this configuration file. Use the StormTest server monitor to do this.

### 4.2 Valid Commands

The commands available vary depending on whether you are testing using an iOS device or an Android device.

The general syntax is of the form

**COMMAND:param1:param2[:optparam1]..[:optparamN]**

Where the command must be upper case. For example:

**TAP:200:300**

Will send the tap command at co-ordinates (200,300)

Sending the command in a python script is as simple as:

```
From stormtest.ClientAPI import PressButton

# Assumes slot is already reserved
PressButton("START-IOS:myappname")
PressButton("TAP:200:300")
```

All supported commands are shown the table below:

Common to iOS and Android	Android Only	iOS Only	Bluetooth commands
TAP:x:y[:count[:duration]]	SWIPE:x1:y1:x2:y2[:time]	SWIPE:x1:y1:x2:y2	START-BT
ORIENTATION:up left right	TAPELEMENT:desc text index resourceid:element	SHAKE	STOP-BT
BACKGROUND:time	START-ANDROID:application[:NORESTART][:LIVE FRAMES]	ORIENTATION:down	TAP:x:y[:count[:duration]]
LOCK:time	STOP-ANDROID	PINCH:x1:y1:x2:y2: x1':y1':x2':y2'	SWIPE:x1:y1:x2:y2[:duration[:startHold[:endHold]]]
VOLUME:up down[:time]	SENDTEXT:textstring	START- IOS:application[:NORESTART][:LIVE FRAMES]	PRESS:<key>
		STOP- IOS	SENDTEXT:textstring
		RAW:command	
		NOVIDEO	
		FRAMES	
		LIVE	
		CCW90	

Table 2: Supported Commands

More detailed descriptions of the commands are shown in the following sections:

#### 4.2.1 Setting the current live mechanism (iOS DUTs only)

The mode used to capture the live video from the iOS DUT can be configured independently of starting the app using the below commands (NOVIDEO/FRAMES/LIVE). If FRAMES or LIVE is called, then START-  
IOS should be called with the video mode of APP\_ONLY to ensure that the previously selected video mode is not overridden.

##### 4.2.1.1 NOVIDEO

Stops the current live display mechanism, if any. This leads to the MPS OS X desktop being displayed through the HDMI out of the Mac mini.

##### 4.2.1.2 FRAMES

Starts the screenshot based live mechanism. This provides a limited number of frames per second but will work for all apps, devices and iOS versions.

##### 4.2.1.3 LIVE

Starts the QuickTime live display mechanism. Only works with lightning equipped iOS devices and iOS 8+. Delivers higher frame rate video to StormTest.

##### 4.2.1.4 CCW90

Rotate the live display counter clockwise with an angle of 90 degrees. This command is only supported when using the FRAMES live mechanism.

#### 4.2.2 START-ANDROID/START- IOS/START-BT

Usage: START-XX:<Application\_Name>[:NORESTART][:VIDEOMODE]



Attempts to start the named application on the device. Apart from the video mode commands in section 4.2.1 above, this must be the first command for iOS applications. For Android devices, it is possible to send commands even without the application under test in the foreground.

If the specified application is already running, it will be terminated and restarted unless the NORESTART option is specified.

The VIDEOMODE is optional and can be one of:

- APP\_ONLY: starts the application only without modifying the current video mode capture mechanism – this requires the tester to have started a video capture mechanism previously using the commands in section 4.2.1
- FRAMES: Captures screen shots from the app under test and displays them as a video stream. This is the default mode.
- LIVE: Display full frame rate live video. This mode only works in the following limited circumstances:
  - iOS devices using the **lightning connector** only
  - Greater than or equal to iOS 8 on the target device
  - The app under test **must** support mirroring of the video

START-BT is intended for Android DUTs to which an HDMI adapter is connected to (and only these). In this configuration the MPS can no longer use ADB over USB and then rely on BLUETOOTH to send commands to the device.

The main advantage is that the live video is much smoother but since ADB is no longer used no command referencing a UI element can be used (only XY commands are allowed like TAP/SWIPE/...)

### 4.2.3 STOP-ANDROID/STOP-IOS

This command stops the active application.

### 4.2.4 TAP:<X>:<Y>[:count[:<DURATION>]]

This command emulates a single finger tap at the specified coordinates. To specify that the user has touched and held a specific point on the screen, the <DURATION> parameter should be added. The duration specifies, in seconds, how long the user has held the touch point. For example, to tap and hold a point at (100,200) for 2 seconds: "TAP:100:200:2"

### 4.2.5 SWIPE:<X1>:<Y1>:<X2>:<Y2>[:TIME]

This command indicates a swipe between two points between (<X1>,<Y1>) and (<X2>,<Y2>). The <TIME> parameter is used to specify how fast the swipe is executed. Note that the TIME parameter is only supported for Android apps.

For Bluetooth it allows the addition of a startHold and endHold parameter. Duration should be 0. This is useful for dragging items on the screen.

### 4.2.6 PINCH:<x1>:<y1>:<x2>:<y2>:<x1'>:<y1'>:<x2'>:<y2'>

This command indicates a pinch or zoom with two fingers. Finger one start and end positions is indicated by (x1, y1), (x2, y2) and finger two is indicated by (x1', y1'), (x2', y2').

For example: "PINCH:10:10:100:100:210:210:110:110" would indicate two fingers starting at (10,10) and (210,210) respectively and ending up at (100,100) and (110,110).

#### 4.2.7 LOCK:<DURATION>

Indicates that the device should be locked for the specified <DURATION> in seconds. It is essential that the passcode entry must be disabled on iOS devices.

#### 4.2.8 SHAKE

Indicates that the device should simulate being shaken. Supported for iOS only.

#### 4.2.9 VOLUME:UP|DOWN:<DURATION>

Specifies whether the user has pressed the hardware keys to increase (UP) or decrease (DOWN) the volume. If duration is omitted, a single click of the volume control is dispatched. If duration is specified then it indicates a single click that is held for the specified number of seconds. For example, to press and hold the UP volume button for 2 seconds: "VOLUME:UP:2" or to perform a single click of the volume down button: "VOLUME:DOWN"

#### 4.2.10 BACKGROUND:<DURATION>

Sends the application into the background for the specified <DURATION> in seconds and then brings it back to the foreground.

#### 4.2.11 ORIENTATION:<DIRECTION>

Instructs the application to alter its orientation. Note that the LEFT and RIGHT directions can be issued multiple times and the app will continue to rotate in that direction by 90°. So issuing "ORIENTATION:LEFT" four times will bring the app back to exactly the same state.

<DIRECTION> can be one of the following values:-

- LEFT – rotate left by 90°
- RIGHT – rotate right by 90°
- UP – orientate the device face up
- DOWN – orientate the device face down (supported on iOS only)

#### 4.2.12 TAPELEMENT:desc|text|index|resourceId:element

*This is an Android only option*

In Android applications, there are three different fields which may be used to 'label' a button or control: content-desc, text and index.

Different applications will use these as they choose. All are valid options, so we have three different ways to interact with them. In addition we support the use of a resource ID to interact with the element. The options available then are:

- TAPELEMENT:desc:someDescription
- TAPELEMENT:text:someText
- TAPELEMENT:index:indexNum
- TAPELEMENT:resourceId:com.sec.android.app.popupcalculator:id/bt\_03

If you are unsure which option to choose, please speak to the app developer. Alternatively, a method for discovering the elements is described in chapter 4.2.14

#### 4.2.13 RAW:<javascript>

*This is an iOS only option*

Allows the user to pass raw JavaScript to the native toolset. This is mainly used for interacting with the UI elements in the app, but any valid JavaScript can be included here.

The correct JavaScript to interact with a specific UI element can be provided by the app developer. Alternatively, a method for discovering the elements and JavaScript is described in chapter 4.2.14.

#### 4.2.14 SENDTEXT:<string>

*This is an Android only option*

In Android applications it is possible to send simple strings of text to a text field, for example a login screen.

There are some caveats however: the characters allowed are limited to the set:

**[a-zA-Z0-9]** and the following extras: **!\$%^\*-\_+=~@ ,.{}[]**

Accented characters (e.g. áüñ etc) are not supported, nor are **£?<>&|()"**

Newlines can be entered using an escaped 'n', for example

```
SENDTEXT:This is on\n two lines
```

A tab character can be entered with an escaped 't', for example

```
SENDTEXT:This is \t tabbed apart
```

#### 4.2.15 PRESS:<key>

This allows the Bluetooth solution to press some physical buttons, namely:

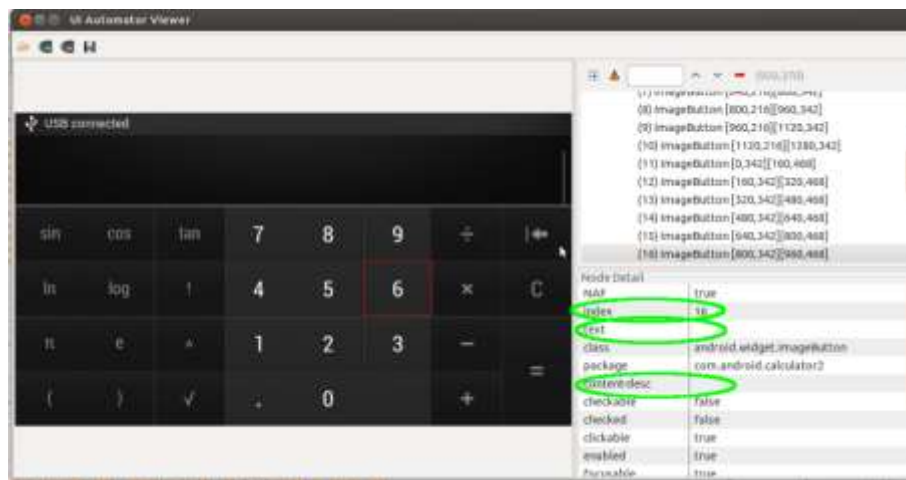
**"UP", "DOWN", "LEFT", "RIGHT", "VOL\_UP", "VOL\_DOWN", "HOME", "MENU", "BACK"**

### 4.3 Discovering UI Elements

Interacting with UI elements instead of using co-ordinates can make your tests more reliable and resistant to UI change. To discover the UI elements in your app you can ask your developers for help, so follow the methods described here.

#### 4.3.1 Android

The Android SDK comes with a useful tool called `uiautomatorviewer` which is a GUI that allows you to scan and analyse the UI components of an Android application



In the above screenshot, the UI Automator Viewer is analysing the standard calculator app. Note the three green identifier fields mentioned previously in section 4.2.12.

In this instance, the only way to interact with the highlighted button (6) is by index as the other options are empty. Thus the command would be:

```
TAPELEMENT:index:16
```

UI Automator Viewer is available on the Mac Mini in /Applications/AndroidSDK/tools

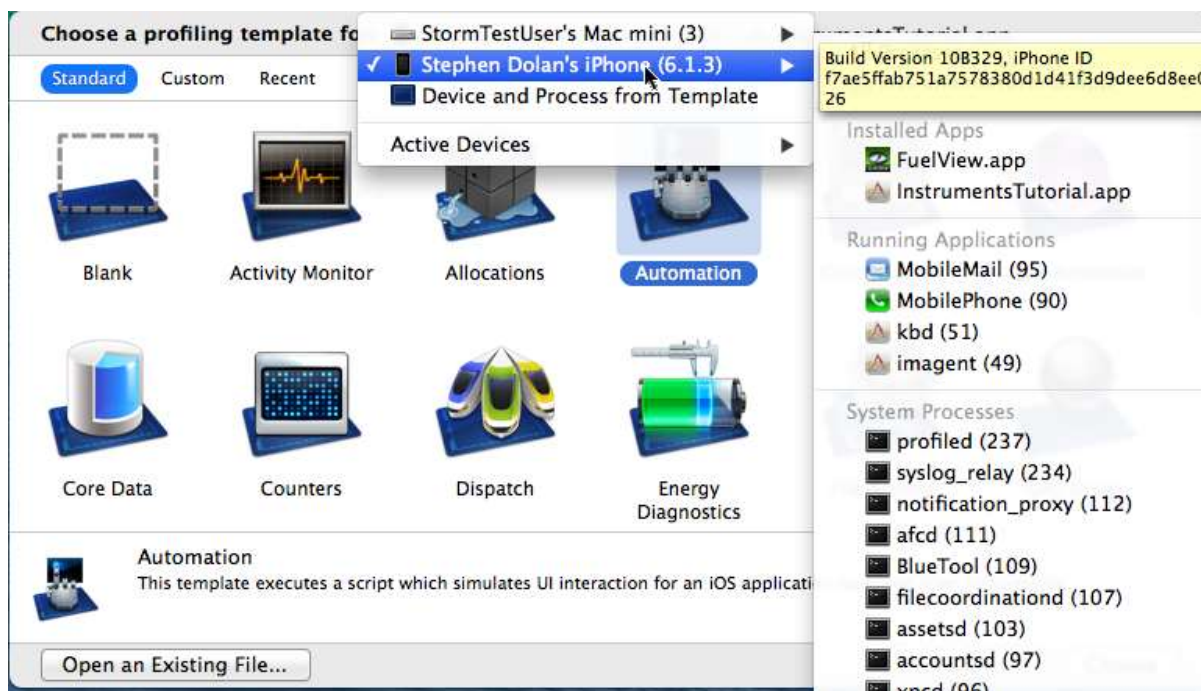
It is also available on Linux or Windows if you install the Eclipse ADT bundle.

### 4.3.2 iOS

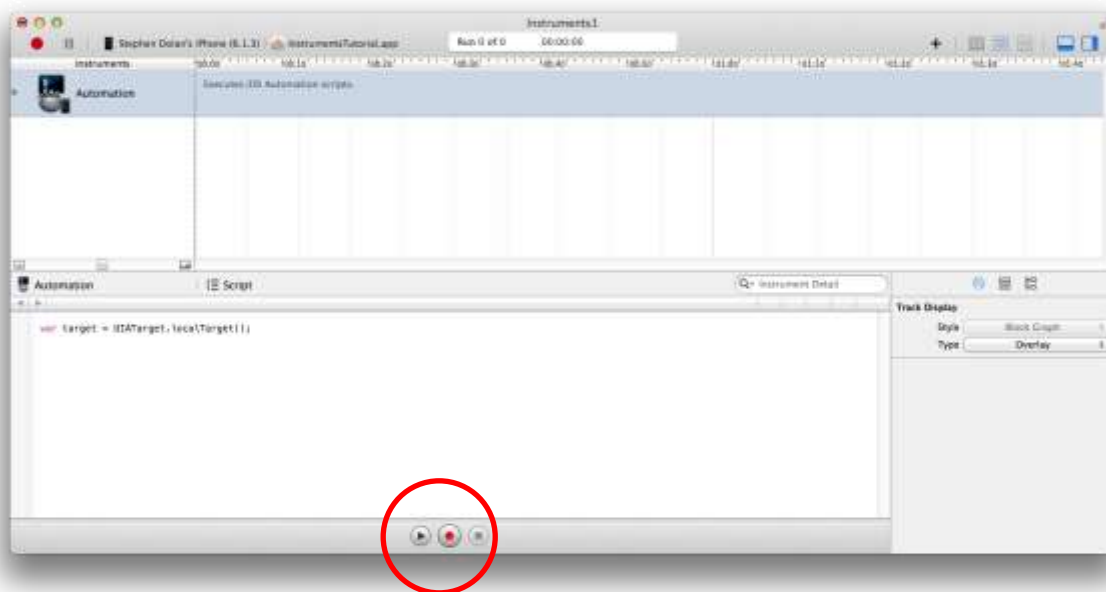
For iOS apps, you must use Xcode to discover the UI elements.

Start Xcode and select *Xcode->Open Developer Tool->Instruments*

Choose the iOS device and your app

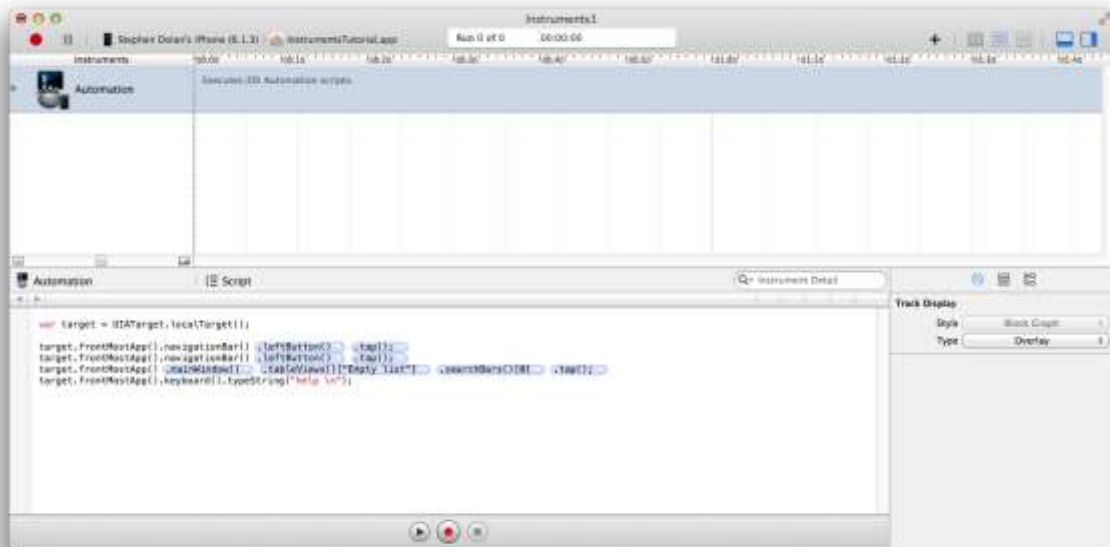


Select the script window and press the *record* button.



Now start interacting with the app. Each interaction (tap, swipe etc.,) will be recorded. Make sure you interact with all the buttons you are interested in.

When done – press stop.



You now have a script with the syntax you need to interact with the UI elements of your app.

The one change needed is that you must 'escape' the quotes, so from the example above, the command you would send to the MPS would be:

```
RAW:target().frontMostApp().mainWindow().tableViews()[0].tap()
list[0].searchBars()[0].tap()
```

## 5 Bluetooth Control of Android Devices

As of version 1.3 of the MPS software, there exists the possibility to use the Bluetooth interface to control the app as a human user, freeing up the micro-USB port for charging and HDMI video output, assuming the device supports it and the app allows mirroring.

In order to provide this functionality, the Mac Mini must have its Bluetooth turned on, and the device must then pair and connect with the Mac Mini.

This chapter provides a detailed guide as to how to perform these steps.

### 5.1 Setting up the Mac Mini

The installation process for the MPS attempts to do this automatically, but in the event of a failure, or if someone manually disables the Bluetooth, this is the sequence to enable it.

#### 5.1.1 Enter the Bluetooth settings application on the Mac Mini

Select the Settings Application, then the Bluetooth Icon



#### 5.1.2 Turn on Bluetooth

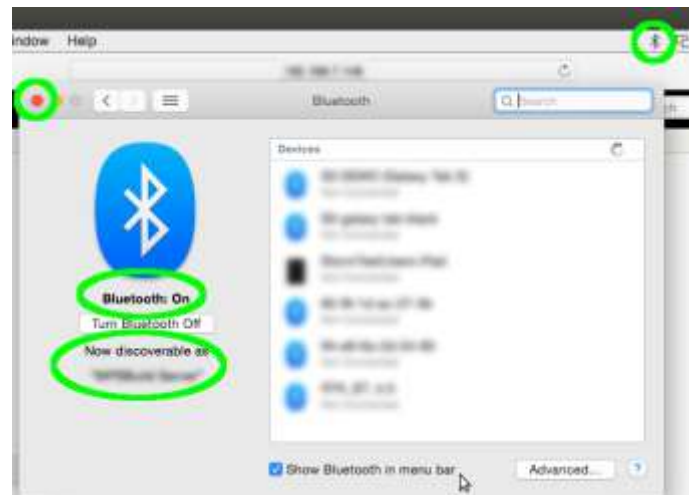
Select the button “Turn Bluetooth On” if the status is “Off”, followed by clicking the “Show Bluetooth in menu bar” box at the bottom





### 5.1.3 Verify that Bluetooth is On

The message should now say “Bluetooth: On” and “Now discoverable as <Mac Mini Name>”. Additionally the Bluetooth Logo should be visible in the title bar. Click the red dot in the top Left Corner to exit this application.



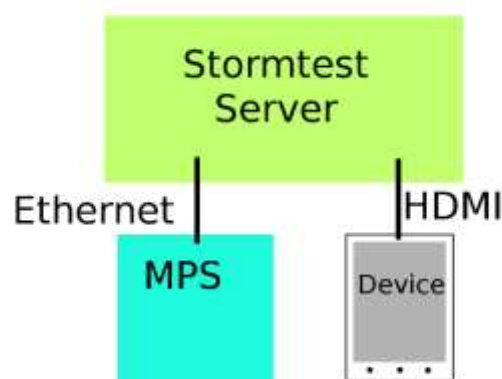
### 5.1.4 Verify the Mac is visible to other devices

Using a device (tablet, phone etc.) verify that it can see the Mac Mini as a Bluetooth device.

## 5.2 Connecting the Device to Stormtest

In the setup for the Android and iOS solutions, the MPS (Mac Mini) is connected to the Stormtest Servers’ HDMI input directly, and the device is connected to the USB port of the MPS.

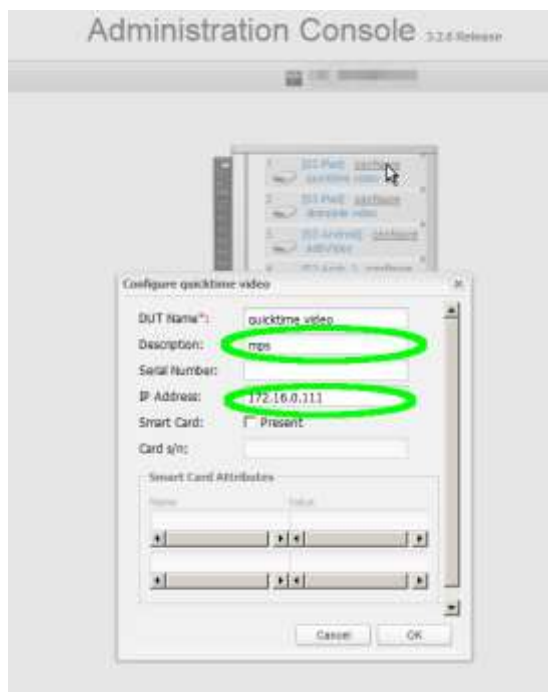
In this installation, the Device is connected directly to the HDMI input of the Stormtest Server, possibly requiring a convertor of some sort to convert from the micro-USB to HDMI (e.g. Samsung use a MHL connector to do such a transformation). Such convertors are **DEVICE SPECIFIC** and need to be used with the correct device. As a result, these convertors are NOT supplied with the StormTest mobile kit and must be sourced by the end user.





### 5.2.1 Configure the DUT

In Stormtest Admin Console, the Description of the device must be set to “mps” (no quotes, all lower case) and the IP Address of the Mac Mini must be set also.



## 5.3 Initiating pairing

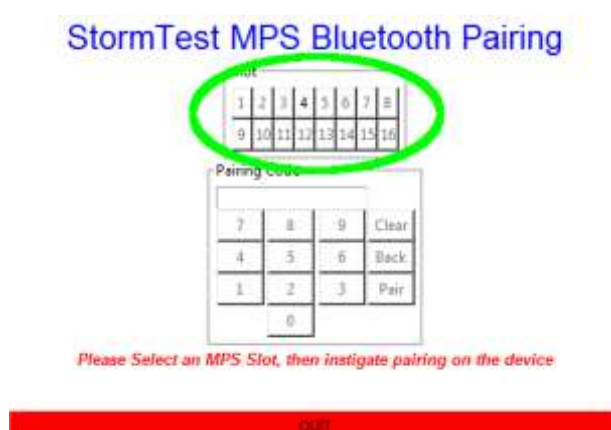
In order for the Bluetooth connection to work, the device must initiate the pairing with the Mac Mini. It cannot be done the other way around, where the Mac mini initiates the pairing connection.

The device needs to be in close proximity to both the MPS and the StormTest Server which the MPS is associated with. Additionally the user will need keyboard and mouse access to the StormTest Server.

### 5.3.1 Launch the Pairing Application on the StormTest Server

As part of the release of MPS there is a Bluetooth Pairing application (called **visualBTPair.py**) which must be copied onto the StormTest Server (suggest to copy it into the standard StormTest Server folder).

Launch this and you will be presented with a GUI similar to this:



The number of slots shown is dependent on the number of slots in the StormTest Server. Non-MPS slots are disabled, cannot be selected and show the status of “This slot is not an MPS, cannot select”:

### StormTest MPS Bluetooth Pairing

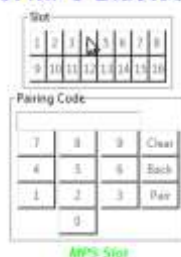


The interface displays a grid of slots (1-16) and a numeric keypad (0-9, Clear, Back, Pair). Slot 1 is highlighted. Below the keypad, a red message states: "This slot is not an MPS, cannot select".

QUIT

MPS Slots are enabled, show a status of “MPS Slot” and can be selected.

### StormTest MPS Bluetooth Pairing




The interface displays a grid of slots (1-16) and a numeric keypad (0-9, Clear, Back, Pair). Slot 1 is highlighted. Below the keypad, a green message states: "MPS Slot".

QUIT

#### 5.3.2 Reserving the Slot for pairing.

Clicking on the slot we wish to pair will attempt to reserve the slot on the server, and if successful the slot will be coloured green, the pairing code buttons will become active and the status will change to “Enter Pairing Code and press ‘Pair’”.

### StormTest MPS Bluetooth Pairing



The interface displays a grid of slots (1-16) and a numeric keypad (0-9, Clear, Back, Pair). Slot 1 is highlighted in green. Below the keypad, a red message states: "Enter Pairing Code and press 'Pair'".

QUIT

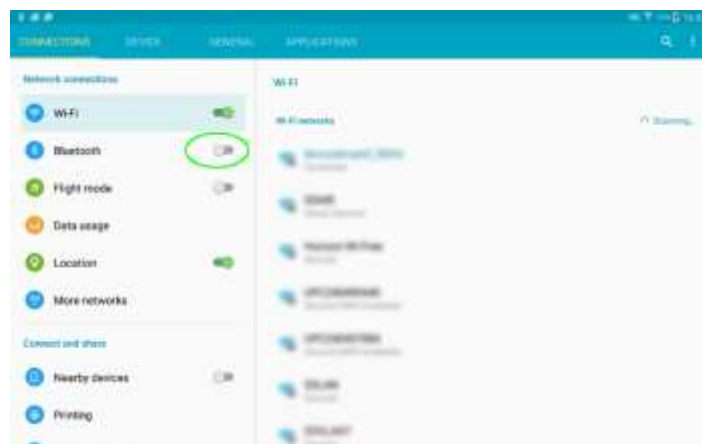
The Pairing code comes from the Device, so at this point the device must be set to initiate pairing. Every device is slightly different, but here is one walkthrough for example purposes.

### 5.3.3 Initiating pairing on an Android Device

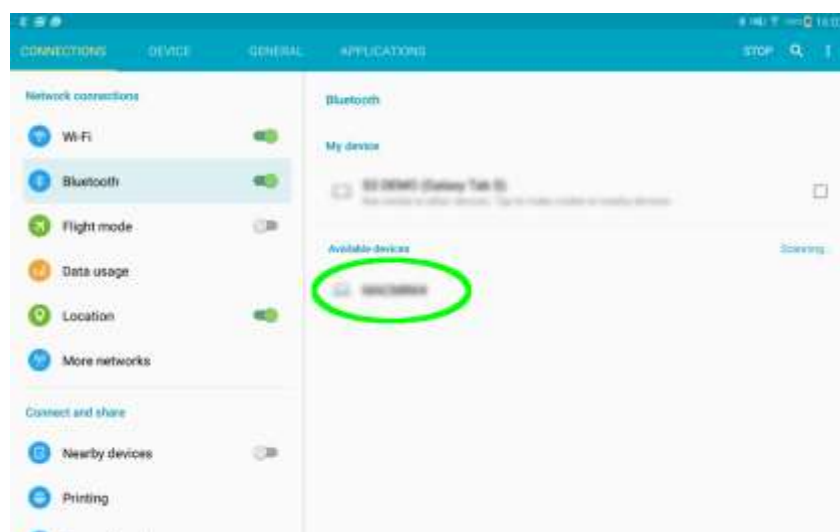
Go to the settings menu of the Android Device



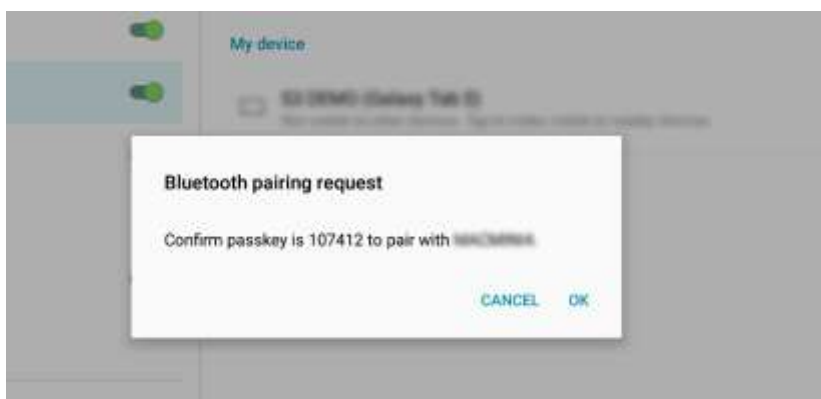
And then turn on Bluetooth if not already turned on.



After the device turns Bluetooth on, it should scan all available servers, and list the MPS with which we wish to pair. There may be many devices listed.

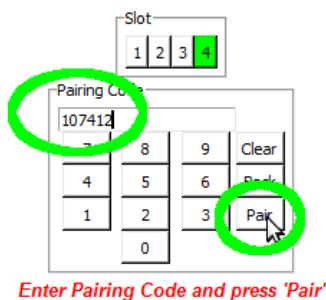


Be sure to select the correct MPS, and click on it. A dialog similar to the one shown below should appear, giving the code which we shall use to pair on the MPS. In this instance our code is 107412.



Going back to the StormTest Server we enter this code either using the keyboard or clicking the buttons, and then press “Pair”. Mistakes can be changed with the “Back” button, and the entire field can be erased with “Clear”.

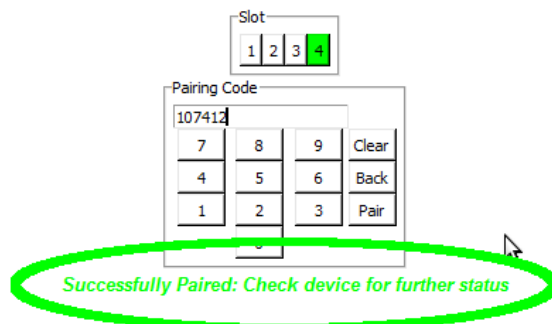
### StormTest MPS Bluetooth Pairing



QUIT

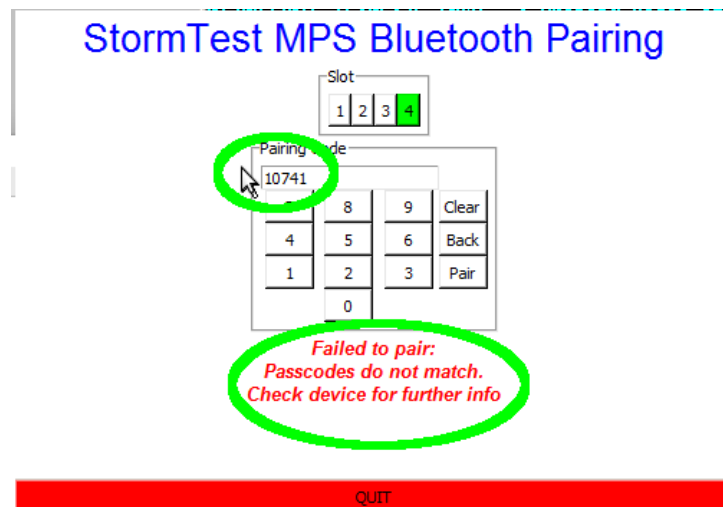
On success, the status will change to “Successfully Paired: Check device for further status”. At this point the device may well be expecting that you click a button to complete the transaction. It may also be in the “Paired” but not “Connected” state: Clicking on the MPS name on the devices paired list should connect it.

### StormTest MPS Bluetooth Pairing



QUIT

If the Pairing number entered is incorrect, the status will be updated to reflect this:



If you wish to pair another device with a different MPS, select the slot for that MPS and repeat the above procedure with the second device. Click "Quit" when done, and it will release any reserved slots on the StormTest Server.

## 5.4 Important things to remember

- Each Device must only be paired with ONE MPS. If moving a device between MPSs, the device MUST be told to unpair from the existing MPS, otherwise strange behaviour may be observed when testing.
- The Device must be connected directly to the StormTest Server to obtain video. If the device does not support video output, or the app being tested does not allow video to be output, then there is no point in using this interface – instead the alternative low frame rate video capture interface should be used.
- Previously, we would tell the device which application to start. In this instance we must navigate to the app and launch it with a tap.
- Different screen sizes will require a modification of the test scripts to account for the different coordinates on screen
- Different specific devices of the same type may struggle to run the same test if there are different applications installed on each device, such that the location of the required application on-screen on each device is different.
- All interaction with the device is now constrained to x-y coordinate based inputs. There are no available button names, fields etc. as per the "TAPELEMENT" command in Android interface.
- Orientation changes cannot be achieved using Bluetooth.

## 6 Writing Tests for an App

As the mobile test solution described here is integrated into StormTest, it is possible to write tests in the same way you are used to for an STB. Any commands that you wish to send to the device can be sent using either

- `PressButton()` call from python (as described in the previous section) or
- `PressButton` block from Test Creator

Validation of a screen in the app uses the same tools in StormTest as an STB test script would use, such as image compare, OCR, colour compare and motion detection.

To support the sending of commands to the app from the UI tools such as Test Creator and Navigator, there are some additional steps that are needed.

Once the RC and Map files are created and in place, controlling the app can be done from the standard StormTest Developer Suite remote control applet.

Simply select the slot containing the device and reserve it.

The remote control shown will be the generic IR remote control and all the other buttons defined will be shown as 'custom' buttons. Clicking on this buttons will send the required commands to the device.

To initiate any interaction, the first button pressed MUST be the `StartApp` button if you are testing on an iOS device.

### 6.1 Creating the Control Files for UI Support

Once the app is properly signed and deployed on the device, StormTest must be configured to allow the GUI to interact with the app.

Interaction is via RC buttons. These buttons can represent


- System commands such as start/stop app or change orientation
- Co-ordinate based interactions (e.g. tap on a co-ordinate)
- UI element based interactions (e.g. tap on the 'menu' button)

The choice as to whether to use co-ordinate based interaction or UI element based interaction is up to the tester. In reality, most testers will used a mix of both and there is no reason not to take this approach.

#### 6.1.1 RC Buttons

The first step is to create a txt file with a list of all the buttons that you need to interact with your app. At the very least, you should create a button to start the app, stop the app and change the orientation.

[illegible]

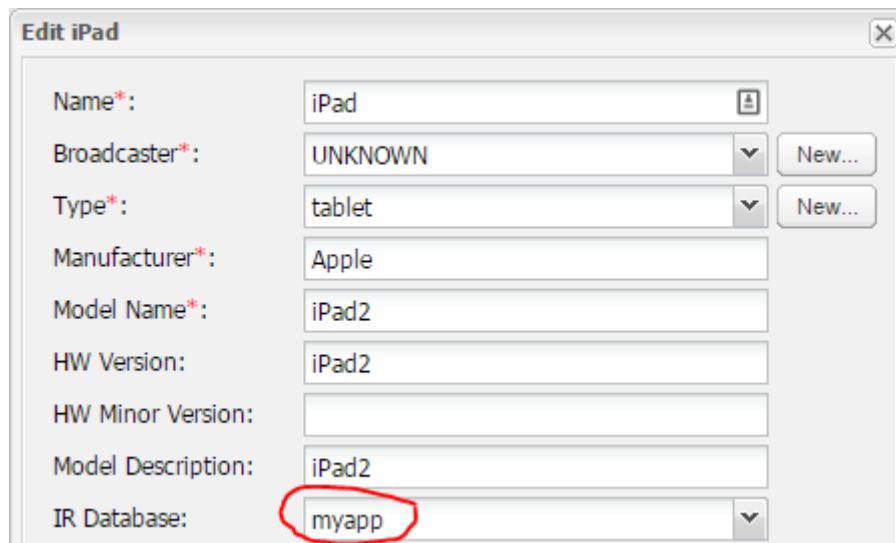
Click on the *import* icon  and select the .txt file you created. This will create the buttons in StormTest and sync the IR database to all your StormTest servers. The file will be given the name **myapp.txt** in this case as we named the RC 'myapp' in the file above. This will be important later when naming the map file.

ButttonName → MOD SIG → 16 000...

If at this point, if you wish to add more buttons, you can edit the original .txt file and re-import it. When prompted, choose to overwrite the existing entry.

Now, you must associate this new ‘remote control’ with the DUT model that represents your iPad or tablet. You can do this through the StormTest Admin Console, a web app installed on your StormTest Config Server.

If it doesn't already exist, create a new DUT model to represent your device. Be sure to set the IR Database to be the new entry you created in Admin Tools



The screenshot shows a dialog box titled "Edit iPad". It contains several fields for configuring a device model:

- Name\*: iPad
- Broadcaster\*: UNKNOWN (with a "New..." button)
- Type\*: tablet (with a "New..." button)
- Manufacturer\*: Apple
- Model Name\*: iPad2
- HW Version: iPad2
- HW Minor Version: (empty)
- Model Description: iPad2
- IR Database: myapp (highlighted with a red circle)

Finally, drag the newly created DUT model into a slot on your server to create the DUT instance.

### 6.1.3 Map File

#### 6.1.3.1 Create Map File

We now have a list of RC buttons that can be used by StormTest in test scripts, navigator and test creator. We need to map these buttons to the underlying commands to control the app.

This is done using a map file that has entries that correspond to the RC buttons you've just created.

The map file takes the form of a dictionary with a key for each RC button and a value that is the command to be executed. For example, for our RC buttons created above, we could have a map file such as this:

```
{
  "StartApp": "START-IOS: K34TYV8SD2.com.accenture--debug",
  "StopApp": "STOP-IOS",
  "Left": "ORIENTATION:left",
  "Right": "ORIENTATION:right",
}
```

The first entry will instruct the MPS to start up the specified iOS app on the attached device. The second entry will exist from the running app. The third and fourth entries rotate the screen orientation left or right.

To add a button that will tap on an area of the screen use something like this. The two numbers are the x and y location on the device screen.

```
"watch": "TAP:285:430",
```



To emulate a swipe, you use the command SWIPE. In this case, you pass the x,y of the start of the swipe, the x,y of the end of the swipe and a count parameter to indicate how many 'fingers' are used (1 or 2). For example:

```
"scroll_up": "SWIPE:299:252:299:709:1",
```

If you wish to use UI elements instead of co-ordinates, then you need to pass in a RAW command to the MPS. To tap on a UI element in an iOS app, you would use a command such as this, which will click on the 'Downloads' button on the main window of the app:

```
"menu5":
"RAW:UIATarget.localTarget().frontMostApp().mainWindow().buttons()[\"Downloads\"].tap()",
```

To get the list of all the available UI elements in the app, you should ask the app developers to generate the list of elements for you.

It is also possible to define one button for more than one element. For example, there may be a 'back' button on two or more separate screens and instead of defining separate buttons in the map file, we can piggyback the javascript for both of these:

```
"back":
"RAW:UIATarget.localTarget().frontMostApp().mainWindow().elements()[\"View1\"].backButton().tap();UIATarget.localTarget().frontMostApp().mainWindow().elements()[\"View2\"].backButton().tap()",
```

Please note that both of these commands are called in series and the one relevant to the current window will be executed. It's important to order these commands to make sure that both don't get inadvertently executed (unless that's the plan).

### 6.1.3.2 Copy Map File

The map file **must** have the same name as the RC file created earlier. So, in the example earlier, we set the device name in the .txt file to *myapp*. This means that the RC file on each server is *myapp.txt*. We therefore need a map file called *myapp.map*.

This map file then needs to be copied to the StormTest server into the C:\Program Files (x86)\StormTest Server\ir\_databases directory.

## 6.2 Navigator

StormTest Navigators can be easily created for testing an app. The links between the navigator screens correspond to commands to the app and the screens in the navigator can be used to encapsulate the validation criteria for each screen in the app.

Since the apps can look very different when they are switched from portrait to landscape mode, it is recommended that two navigators are created with the same structure and the same screen names, one for portrait and one for landscape. This can be done by creating the first navigator in one orientation, creating a copy of the file and then recapturing each screen with the app in the other orientation.

## 6.3 Remote Control Skin

By default, the remote control interface will show the standard generic remote control skin. All the new buttons defined will show up as custom buttons. There is no requirement to define a remote control skin, but if you will be manually controlling the app remotely, it may make that process easier.

If you wish to create a skin, remember that there is only one 'view' that can be shown in the skin. If the structure of your app is quite well defined (for example all controls are arranged in a grid) then that won't be an issue. However, if your app is very dynamic, then it may not be possible to create a useful skin for your app.

We recommend that you have a conversation with the StormTest support team and they will advise on whether a skin can be created. They can also help to create and install that skin.

## 7 Troubleshooting

### 7.1 FAQ

#### 7.1.1 Why don't I see my app in the remote control video?

When you initially power on the MPS and connect to the video stream from that slot, you will not see the device or app that you are testing. Instead, you will see the desktop of the Mac mini, which will display a status message from the MPS software.

This is deliberate, as it makes it easier to verify that everything is running before testing commences. Once you've pressed the 'StartApp' button once, you will see the output from the device you are testing.

#### 7.1.2 When I open the video to the slot I see a 'dongle not found' error

If you see an image like this when you view the video on the slot with the MPS, then it means that you have not inserted the license dongle, or the dongle is not inserted correctly.



Remove and reinsert the dongle and restart the MPS. You should then see green messages indicating that the status is now ok.

#### 7.1.3 Can I connect two devices at once to a slot?

You can physically connect two devices to the MPS at once IF one is an iOS device and the other an Android device.

You cannot connect two devices with the same OS type at the same time.

However, while two devices can be connected only one can be active and tested at a time. If two devices are connected, then the RC buttons will need to include two 'StartApp' buttons, one to launch an app on the iOS device and another to launch the app on the Android device.

Before switching between devices, you must call Stop on the current app and then Start on the new app.

#### **7.1.4 Can I test more than one app?**

Yes, absolutely. You will need to define two 'StartApp' buttons so you can choose which app to start and you can only have one app started at any one time. But there is no reason why you can interact with one app, stop it and start a second app, as long as both have been appropriately signed (for iOS devices)

#### **7.1.5 Does the MPS have to be physically in the slot?**

Strictly speaking it doesn't. However as it needs an HDMI connection to the slot, it is recommended that both the MPS and the tablet device are placed in the slot.

#### **7.1.6 Can I test audio and video in my app?**

Yes! The audio and video from the device is captured by the MPS and transferred to StormTest over the HDMI cable. As the video is captured as screenshots, it is not full frame rate video, but it is sufficient to determine if video motion is present or not.