Ship Apps with Confidence
Solving the Mobile Quality Problem
## Table of contents

01  The state of mobile quality

02  The diverse device landscape

04  Mobile quality approaches

06  Xamarin Test Cloud

07  Xamarin’s model for mobile quality

09  Troubleshoot quickly with screenshots and graphical reports

10  Flexible automation

11  The world's largest collection of iOS and Android devices

11  Continuous quality

14  **Accelerators for going mobile**

  Enterprise-class support and services

  Enterprise support

  Consulting partners

  Managed testing services
The mobile app market continues to grow at a staggering rate. There are 224 million active monthly app users in the U.S. alone, and the amount of time people spend using apps increased 21% from 2013 to 2014. The market is expected to continue to grow for the foreseeable future, as the worldwide penetration rates of both smartphones and tablets increases, and as the app model expands to a variety of new connected devices such as smartwatches and next-generation cars, homes and business facilities.

While mobile users are more in love with their mobile apps than ever, they’re also less willing to tolerate a subpar user experience. One in five apps is opened only once, and surveys have consistently found that users are quick to give up on apps that are slow or buggy.

The challenge for businesses, then, is to create apps that perform at a high level on multiple operating systems and for an ever-increasing number of devices that must be constantly tested and monitored to ensure they continue to run optimally across all devices and configurations.

How will your business guarantee high-quality apps for millions of users on thousands of different devices?
In the US, companies must test on **134** devices to cover **75%** of devices in use.
The diverse device landscape

The mobile computing landscape is far more fluid and fragmented than it was during the PC era, and is changing much more rapidly as well. From 2009 to 2014, Android vaulted from a 4% smartphone market share to a whopping 76.6%. But, while Android dominates device volume, Apple generates more sales and higher revenue.

Nearly 20,000 different devices are currently in use running Android, the world’s dominant mobile operating system, and those gadgets have various sizes, form factors and manufacturer customizations. Additionally, Apple’s iOS runs on approximately two dozen configurations on the iPhone and iPad. Even if an organization commits to a single platform – which is a strategy that leaves wide swaths of the market unaddressed – device fragmentation makes it extremely difficult to guarantee app quality and consistency on all form factors.

Xamarin’s own device market share data illustrates exactly how challenging the fragmentation problem is – testing on 134 devices only covers 75% of device ownership in the US. This does not include the different operating systems which can triple or quadruple the number of device configurations required to ensure complete coverage. Most companies today still test on only a handful of devices, putting the experience of the majority of users at risk.
Popular apps like Instagram, Dropbox, and Facebook Messenger have shaped user expectations for all apps: they must be beautiful, fast, and frequently updated. User sessions often last just seconds, so a high-performance app that delivers information quickly can mark the difference between a highly-rated app that attracts new users, and a poorly rated app that drives away audiences and generates negative word of mouth. Employee-facing apps are judged by these same experience standards; employees will abandon ugly, slow apps, breaking the productivity promise of a mobile workforce.

The huge variety of form factors, OS versions, and OEM configurations makes guaranteeing app quality and consistency across a wide range of devices extremely difficult with standard methods for addressing mobile quality. Automatically testing apps on real devices, and monitoring apps and user behavior once released are the only sustainable ways to solve the fragmentation problem.
Simulator-only testing

By testing an app on a simulator, developers can execute their apps in a runtime environment without leaving their development environment. This approach is convenient and reduces costs and setup time because it doesn’t require actual mobile hardware. This is the most basic way to spot check functionality.

However, basing a production mobile quality strategy on just using simulators has significant downsides, whether using them in a manual or automated testing environment:

• Unrealistic CPU, memory, and performance caused by virtualization.
• WiFi, GPS, camera, sensors, etc. are simulated so error conditions are often not exercised.
• OEM and carrier customizations missing from simulators affect app behavior in unforeseen ways.

Simulators may appear to lower expenses for testing apps, but they provide a false sense of security because they fail to take into account how apps perform and behave on physical devices, making simulator-only testing a dead-end for guaranteeing real-world quality.

Manual testing

Many mobile teams rely on manual testing, usually on a handful of devices with test coverage focused on new features, not on identifying regressions. Manual testing runs the gamut from developers testing features as they build them, to a dedicated QA team following testing scripts and logging the results. Teams often assume that users have the same devices they’re testing, but apps are downloaded on hundreds of different models and OS versions, so even rigorous manual testing has pitfalls:

• Poor feature and device coverage, as tests are biased toward new features on popular devices.
• Slow result turnaround (often weeks).
• Security vulnerabilities because apps and data are in uncontrolled environments.

Manual testing requires significant investment and ongoing expense, with a TCO upwards of $1,400 per device. Human testers often make mistakes or skip steps. Logging each step and reporting bugs is a painstaking process that can limit the number of features and devices tested and results often take weeks to report. Ultimately, manual testing adds significant expense and overhead along with a loss in time, agility, and innovation.
In contrast to manual testing and simulator-only testing, Xamarin Test Cloud lets teams test every feature on more than a thousand devices and on every commit. Catching bugs before release shortens development cycles and allows more time for innovation.
Xamarin’s model for mobile quality

• **Complete test coverage.**
  Using automation, developers can test every new feature and perform comprehensive regression tests from the UI down to ensure high-quality releases every time.

• **Comprehensive device testing.**
  Run an app on the largest device cloud in the industry with more than 1,200 real devices. Select devices based on form factor, manufacturer, operating system, or even target market popularity.

• **Fast troubleshooting.**
  See screenshots and video for every step of every test, then receive performance data and compare reports against previous runs to find regressions and bottlenecks.

• **Accelerated cycles with continuous integration.**
  Xamarin Test Cloud integrates with TFS, Jenkins, TeamCity, or any Continuous Integration (CI) systems with custom post-build commands, enabling collection of test results in a CI system automatically.

• **Comprehensive support for all native and hybrid apps.**
  Xamarin Test Cloud tests all apps, including apps built in C# with Xamarin as well those built in Objective-C, Swift, Java, and other cross-platform frameworks.

“Xamarin Test Cloud is our path to a low-maintenance, high-quality, regression-free future. We’re excited about where this can take our team and product.”

Sean Beausoleil
Mailbox Engineering Lead at Dropbox
### Number of tests
- **15** in latest run

### App size
- **17.78 MB** in latest run

### Peak memory
- **159.70 MB** in latest run

<table>
<thead>
<tr>
<th>Test run</th>
<th>Version</th>
<th>Results</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug 23, 2015 10:03:59 PM</td>
<td>2.0 (1.0)</td>
<td>4/15 tests failed</td>
<td>10</td>
</tr>
<tr>
<td>Aug 22, 2015 10:03:34 PM</td>
<td>2.0 (1.0)</td>
<td>4/15 tests failed</td>
<td>10</td>
</tr>
<tr>
<td>Aug 21, 2015 10:03:20 PM</td>
<td>2.0 (1.0)</td>
<td>2/15 tests failed</td>
<td>10</td>
</tr>
<tr>
<td>Aug 21, 2015 4:27:35 PM</td>
<td>2.0 (1.0)</td>
<td>4/15 tests failed</td>
<td>10</td>
</tr>
<tr>
<td>Aug 21, 2015 10:05:17 AM</td>
<td>2.0 (1.0)</td>
<td>15/15 tests failed</td>
<td>30</td>
</tr>
<tr>
<td>Aug 20, 2015 12:00:28 PM</td>
<td>2.0 (1.0)</td>
<td>10/15 tests failed</td>
<td>10</td>
</tr>
</tbody>
</table>

### Devices
- **Apple iPhone 6**
  - Screen size: 4.7 in (11.9 cm)
  - Resolution: 750 x 1334 (166 ppi)
  - Release date: September 2014
  - Market share: 5.51%

### Test Log
- **Full Size**
- **Device Log**
- **Test Log**
Troubleshoot quickly with screenshots and graphical reports

The results dashboard compares the most recent test run to historical data, showing trends in app size and peak memory usage over time. Drilling into an individual test run shows overall patterns such as problems with a particular operating system, form factor, or manufacturer.

Drilling down further, Xamarin Test Cloud shows full-resolution screenshots and recordings of every interaction on every page of an app, providing visual comparisons among screen sizes, form factors, and operating system versions.

For functional failures, Xamarin Test Cloud provides precise device specifications, console logs, and CPU and memory metrics for every step leading up to the failure.
Flexible automation

Xamarin’s UIT est and Calabash frameworks enable powerful automated UI testing. Developers write tests that behave as users do, performing taps, swipes, rotations, and waiting for UI elements to appear. Xamarin Test Cloud results are available within minutes, and include new feature testing as well as comprehensive regression tests for complete coverage and peace-of-mind that apps will work once released.

Scripts are object-based and flexible to UI changes that may happen in design and development, unlike brittle optical character recognition (OCR) solutions that can break with just small changes to the app UI, or with rotation or form-factor size changes.

Tests are written in C# in Xamarin Studio or Visual Studio, or in Ruby using Calabash, the industry’s most powerful framework for automated testing of native and hybrid apps.

C# Example

[Test]
public void ShouldBeAbleToPlaceLimitOrders()
{
    app.EnterText(x => x.Id("Username"), "cathy@business.com");
    app.EnterText(x => x.Id("Password"), "xka13482733");
    app.Tap(x => x.Text("Log in"));
    app.WaitForElement(x => x.Text("Place Order"));
    Assert.IsNotNull(app.Query(x => x.Class("Button").Text("Limit")));
    app.Screenshot("Place limit order");
}

Calabash Example

Feature: Trades
Scenario: Selectable Trading Order Types
Given I am logged in as “Cathy”
When I go to the Trading Orders Screen
Then it should display the Trading Orders order by Id
Xamarin Test Recorder

As an alternative to writing your own scripts, Xamarin Test Recorder makes it easy to record automated tests for your app. Simply connect Test Recorder to your preferred iOS or Android device, and it will automatically generateUITest scripts by capturing your interactions. Once you finish recording and editing your test, you can export your test or upload it to Xamarin Test Cloud in one click.

The world’s largest collection of iOS and Android devices

Xamarin Test Cloud provides automated mobile testing on hundreds of real devices in a secure device cloud.

The only way to know the user experience is to test on the devices consumers use, not on simulators or emulators, which is why Xamarin Test Cloud features over 1,200 real devices that can be selected by market share, manufacturer, form factor, or OS version.

Continuous quality

Instead of running a test from start to finish on a device, Xamarin Test Cloud’s parallelization feature separates test runs and executes them across multiple identical devices simultaneously, significantly increasing testing speed and frequency.

Xamarin Test Cloud is designed to integrate with CI tools like Jenkins, Team City, Team Foundation Server, and Visual Studio Online, so a test run can kick off with each commit and test results and failures can be included in nightly reports. Issues that are found and fixed early in the development cycle significantly reduce risk, so teams can develop cohesive, high-quality software much faster.
## iOS Devices

<table>
<thead>
<tr>
<th>Phone Model</th>
<th>OS Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone 5S</td>
<td>iOS 7.0.4</td>
</tr>
<tr>
<td>iPhone 5C</td>
<td>iOS 8.0, 7.1.1, 7.1, 7.0.4, 7.0.2</td>
</tr>
<tr>
<td>iPhone 5</td>
<td>iOS 7.1.1, 7.0.4, 6.1.4</td>
</tr>
<tr>
<td>iPhone 4S</td>
<td>iOS 8.0, iOS 7.1.1, 7.1, 7.0.4, 6.1.3, 5.1.1</td>
</tr>
<tr>
<td>iPhone 4</td>
<td>iOS 7.1.1, 5.1.1</td>
</tr>
<tr>
<td>iPhone 3G</td>
<td>iOS 6.1.3</td>
</tr>
<tr>
<td>Other</td>
<td>iOS 6.1.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tablet Model</th>
<th>OS Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad Mini Retina</td>
<td>iOS 8.0, iOS 7.1.1, iOS 7.0.4, iOS 7.0.3</td>
</tr>
<tr>
<td>iPad Mini</td>
<td>iOS 8.0, iOS 7.1.1, iOS 7.0.4, 7.0.3, 6.1.3</td>
</tr>
<tr>
<td>iPad Air</td>
<td>iOS 8.0, iOS 7.1.1, 7.1, 7.0.4</td>
</tr>
<tr>
<td>iPad</td>
<td>iOS 7.1.1</td>
</tr>
<tr>
<td>iPad 4</td>
<td>iOS 8.0, iOS 7.0.4, 6.1.3</td>
</tr>
<tr>
<td>iPad 3</td>
<td>iOS 5.1.1</td>
</tr>
<tr>
<td>iPad 2</td>
<td>iOS 7.1.1, 7.1, 7.0.4, 7.0.3, 6.1.3</td>
</tr>
<tr>
<td>iPad</td>
<td>iOS 5.1.1</td>
</tr>
<tr>
<td>Other</td>
<td>iOS 7.1.1, 7.1, 7.0.4, 6.1.3, 6.0.1</td>
</tr>
<tr>
<td>iPod touch 5th generation</td>
<td>iOS 6.1.</td>
</tr>
<tr>
<td>iPod touch 4th generation</td>
<td>iOS 6.1.</td>
</tr>
</tbody>
</table>

## Android Devices

<table>
<thead>
<tr>
<th>Phone Models</th>
<th>OS Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acer Iconia Tab A1</td>
<td>Jelly Bean 4.2.2</td>
</tr>
<tr>
<td>Amazon Fire Phone</td>
<td>Jelly Bean 4.2.2</td>
</tr>
<tr>
<td>Amazon Kindle Fire</td>
<td>Ice Cream Sandwich 4.0.3</td>
</tr>
<tr>
<td>Amazon Kindle Fire</td>
<td>Ice Cream Sandwich 4.0.4</td>
</tr>
<tr>
<td>ASUS Memo Pad</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>ASUS Transformer</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>Google Nexus 7</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>HP Slate 7</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>HTC Desire</td>
<td>Froyo 2.2.2</td>
</tr>
<tr>
<td>HTC Desire C</td>
<td>Ice Cream Sandwich 4.0.3</td>
</tr>
<tr>
<td>HTC Desire HD</td>
<td>Gingerbread 2.3.5</td>
</tr>
<tr>
<td>HTC Desire V</td>
<td>Ice Cream Sandwich 4.0.3</td>
</tr>
<tr>
<td>HTC Incredible S</td>
<td>Gingerbread 2.3.5</td>
</tr>
<tr>
<td>HTC One</td>
<td>KitKat 4.4.2, Jelly Bean 4.3, 4.2.2, 4.1.2</td>
</tr>
<tr>
<td>HTC One (M8)</td>
<td>KitKat 4.4.3</td>
</tr>
<tr>
<td>HTC One Max</td>
<td>Jelly Bean 4.3</td>
</tr>
<tr>
<td>HTC One Mini</td>
<td>Jelly Bean 4.2.2</td>
</tr>
<tr>
<td>HTC One mini 2</td>
<td>KitKat 4.4.2</td>
</tr>
<tr>
<td>HTC One X</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>HTC One X+HTC Sensation XL</td>
<td>Jelly Bean 4.1.1</td>
</tr>
<tr>
<td>HTC Wildfire</td>
<td>Ice Cream Sandwich 4.0.3</td>
</tr>
<tr>
<td>Huawei Ascend Y</td>
<td>Froyo 2.2.1</td>
</tr>
<tr>
<td>Huawei Ascend Y300</td>
<td>Gingerbread 2.3.6</td>
</tr>
<tr>
<td>LG G2</td>
<td>Huawei Ascend Y300</td>
</tr>
<tr>
<td>LG G3</td>
<td>LG Optimus Y300</td>
</tr>
<tr>
<td>LG G Flex</td>
<td>LG G3</td>
</tr>
<tr>
<td>LG Nexus 4</td>
<td>LG G Flex</td>
</tr>
<tr>
<td>LG Nexus 5</td>
<td>LG Nexus 4</td>
</tr>
<tr>
<td>LG Optimus 3D</td>
<td>LG Nexus 5</td>
</tr>
<tr>
<td>LG Optimus L5</td>
<td>LG Optimus L3 E400</td>
</tr>
<tr>
<td>LG Optimus L9 P710</td>
<td>LG Optimus L5</td>
</tr>
<tr>
<td>LG Optimus L9 P768</td>
<td>LG Optimus L9 P710</td>
</tr>
<tr>
<td>LG Optimus One P500h</td>
<td>LG Optimus One P500h</td>
</tr>
<tr>
<td>Motorola ATRIX 2</td>
<td>LG Optimus One P500h</td>
</tr>
<tr>
<td>Motorola DROID Maxx</td>
<td>Motorola ATRIX 2</td>
</tr>
<tr>
<td>Motorola DROID Mini</td>
<td>Motorola DROID Maxx</td>
</tr>
<tr>
<td>Motorola Moto X</td>
<td>Motorola DROID Mini</td>
</tr>
<tr>
<td>Motorola Moto X715</td>
<td>Motorola Moto X</td>
</tr>
<tr>
<td>Motorola RAZR</td>
<td>Motorola Moto X715</td>
</tr>
<tr>
<td>Motorola XOOM</td>
<td>Motorola RAZR</td>
</tr>
<tr>
<td>Nokia X Dual Sim</td>
<td>Motorola XOOM</td>
</tr>
<tr>
<td>Oppo N1</td>
<td>Nokia X Dual Sim</td>
</tr>
<tr>
<td>Oppo R819</td>
<td>Oppo N1</td>
</tr>
<tr>
<td>Oppo R819</td>
<td>Oppo R819</td>
</tr>
<tr>
<td>Samsung Epic 4G</td>
<td>Samsung Epic 4G</td>
</tr>
<tr>
<td>Samsung Galaxy Ace</td>
<td>Samsung Galaxy Ace</td>
</tr>
<tr>
<td>Samsung Galaxy Ace</td>
<td>Samsung Galaxy Ace</td>
</tr>
</tbody>
</table>
Android Devices  (continued)

Samsung Galaxy Ace 2
Samsung Galaxy Ace Plus
Samsung Galaxy Beam
Samsung Galaxy Centura
Samsung Galaxy Core
Samsung Galaxy Express
Samsung Galaxy Grand Duos
Samsung Galaxy Mega 6.3
Samsung Galaxy Mini 2
Samsung Galaxy Nexus
Samsung Galaxy Note
Samsung Galaxy Note 10.1 3G & WiFi
Samsung Galaxy Note 10.1 WiFi
Samsung Galaxy Note 3 (Octo-Core)
Samsung Galaxy Note 3 (Quad-Core)
Samsung Galaxy Note 8.0
Samsung Galaxy Note II
Samsung Galaxy Note II LTE
Samsung Galaxy Pocket Neo
Samsung Galaxy Precedent
Samsung Galaxy S
Samsung Galaxy S4
Samsung Galaxy S4 Active
Samsung Galaxy S4 Mini
Samsung Galaxy S4 Mini Duos
Samsung Galaxy S4 (Octo-core)
Samsung Galaxy S4 Zoom
Samsung Galaxy S5
Samsung Galaxy S Advance
Samsung Galaxy S Duos
Samsung Galaxy S Duos 2
Samsung Galaxy S II
Samsung Galaxy S III
Samsung Galaxy S III LTE
Samsung Galaxy S III Mini
Samsung Galaxy S III Mini (NFC)
Samsung Galaxy S III (US Carrier)
Samsung Galaxy S II Plus
Samsung Galaxy Star
Samsung Galaxy Tab 2
Samsung Galaxy Tab 2 7.0
Samsung Galaxy Tab 3 10.1
Samsung Galaxy Tab 3 7.0
Samsung Galaxy Tab 3 8.0
Samsung Galaxy Win Duos
Samsung Galaxy Y
Samsung Galaxy Y Duos
Samsung Galaxy Young
Samsung Google Nexus 10
Samsung Google Nexus 5
Sony Ericsson Live With Walkman
Sony Ericsson Xperia Mini
Sony Ericsson Xperia PLAY
Sony Ericsson Xperia S
Sony Ericsson Xperia U
Sony Miro
Sony neo L
Sony P
Sony SP
Sony T
Sony Xperia Tablet Z Wifi
Sony Xperia Tipo
Sony Xperia V
Sony Xperia Z
Sony Xperia Z1
Sony Xperia Z2
Sony Xperia ZL
Sony Xperia ZR
Sony Xperia Z Ultra HSPA+
Sony Xperia Z Ultra LTE
ZTE Avid 4G

We're adding over 100 devices per month. Visit testcloud.xamarin.com/devices for the complete list.
Accelerators for going mobile

Xamarin offers the ideal platform for testing any native or hybrid app, as well as training and services to ensure success. Xamarin helps businesses at every step of the mobile journey—from training that jumpstarts the automation process, to running testing as a fully outsourced managed service.

Enterprise-class support and services

Enterprise support

Our enterprise support offerings include one day response SLAs, access to the latest hotfixes, technical training for getting started, and access to resources for troubleshooting.

Managed testing services

Xamarin’s expert automation engineers help teams automate the mobile quality process, from providing coaching, all the way to building and maintaining a complete test suite as a managed service—provide the app use case and test scenarios, and Xamarin will automate and execute tests on device and report results back.

Consulting partners

Customers can also tap into our Consulting Partners program that includes a worldwide network of over 300 Premier and Authorized Xamarin Consulting Partners who have first-hand experience helping clients execute a mobile quality strategy.
"As a QA Engineer, I am really enjoying Xamarin Test Cloud because it offers me a broad range of Android and iOS devices, which gives me the ability to have more test coverage and to discover issues sooner. And what’s particularly helpful is that Xamarin offers exceptional service and support that I can depend on every day."

Derrick Lam, QA Engineer at Flipboard

"Xamarin Test Cloud has helped us catch important bugs in challenging scenarios, and to ensure they never occur again."

Hamid Palo, Android Team Lead at Trello

"Xamarin Test Cloud helps us ensure that critical features of Expensify look and perform great on every Android device imaginable."

Tom Jacobs, Software Engineer at Expensify

"Xamarin Test Cloud helped us identify and fix critical device-specific issues in the latest release of Foursquare before we released."

Mike Singleton, Software Engineer at Foursquare
“Xamarin Test Cloud plays a fundamental role in our continued effort to maintain the highest possible quality of our products and also securing a short time-to-market.

Our test setup takes care of most of the tedious work of doing regression tests and gives our team quick feedback, while helping us deliver new features at a fast, predictive speed.”

Niels Frydenholm, eBay Denmark