The Importance of Quality Assurance Testing for the Internet of Things

Andrei Babulevich, Ron Mader, Dan Myers, Sudha Sundaresan
# The Importance of Quality Assurance Testing for the Internet of Things

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Test Requirements</td>
<td>3</td>
</tr>
<tr>
<td>Device-Level Testing</td>
<td>4</td>
</tr>
<tr>
<td>Target Test Areas</td>
<td>6</td>
</tr>
<tr>
<td>Cloud-Level Testing</td>
<td>7</td>
</tr>
<tr>
<td>Performance, Reliability and Scalability Testing</td>
<td>7</td>
</tr>
<tr>
<td>Security Testing</td>
<td>7</td>
</tr>
<tr>
<td>Data Governance</td>
<td>8</td>
</tr>
<tr>
<td>Data Privacy Testing</td>
<td>8</td>
</tr>
<tr>
<td>Mobile-Level Testing</td>
<td>9</td>
</tr>
<tr>
<td>Backward Compatibility Testing</td>
<td>10</td>
</tr>
<tr>
<td>CRUD Testing</td>
<td>11</td>
</tr>
<tr>
<td>Lifecycle Testing</td>
<td>11</td>
</tr>
<tr>
<td>End-to-End Testing</td>
<td>11</td>
</tr>
<tr>
<td>Field Trials</td>
<td>11</td>
</tr>
<tr>
<td>Summary</td>
<td>11</td>
</tr>
</tbody>
</table>
Overview

While connected products can offer tremendous benefits to both end-users and manufacturers, adding connectivity to devices is no easy task. The market has already witnessed some high profile failures of Internet of Things (IoT) devices, illustrating the importance of proper testing.

It is essential to implement a robust regression testing process that covers not only the device but the interaction with the cloud and mobile apps as well, to ensure the product is functioning as intended. As an example of why this is important, many IoT platforms feature over-the-air (OTA) updates, which enable manufacturers to upgrade firmware in devices remotely. If this feature isn’t properly tested, upgrades can disable devices and interrupt service for end-users, causing dissatisfaction, affecting customer loyalty and, ultimately, impact revenue.

The reality is that many manufacturers are not prepared to address all of the complexities involved with adding connectivity to their products. Selecting a mature IoT platform can help manufacturers address these challenges, lowering development costs and reducing time to market.

Test Requirements

The IoT is a system that consists of several components, which are often developed in parallel. It’s critical to have an infrastructure that can support the required tools to properly test individual components during the integration process, identifying and addressing potential problems.

With all of these different components, manual processes are insufficient for testing. For example, here at Ayla Networks, we currently support 17 different connectivity modules and two serial interfaces for each, which makes 34 various combinations. Each connectivity module has to be validated by about 1,000 test cases, which means 34,000 tests for a full release. Without automation, this task would be impossible. Facing such staggering numbers, it’s critical that members of the QA team be developer-grade programmers, capable of writing complicated and reliable test scripts, and enabling test hooks in the software.

The Open Web Application Security Project (OWASP) is considered to be the benchmark for security and web applications. Its Application Security Verification Standard (ASVS) contains a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, and even consumers to define what a secure application is. At a minimum, your cloud service provider should strive to meet this standard.
The typical IoT platform consists of three main system components:

1. Transport: the connectivity module
2. Service: the cloud
3. Application: the mobile app for end users to interact with the device

A proper QA process implemented by the IoT platform provider ensures the characteristics of a high-quality product: reliability, performance and visibility. However, to enable QA testing, the IoT platform should provide visibility into the network traffic at all system levels. Platform providers that implement comprehensive QA testing including this multi-level traffic, and use analysis, can provide visibility into potential product issues, allowing easy maintenance and painless device updates.

In this paper, we will examine key testing requirements for the three major components of a high-quality, reliable IoT system as well as how the IoT platform vendor’s QA testing processes can enable thorough testing of your implementation.

Device-Level Testing

Logging is important for device-level testing because it provides visibility into time-stamped events and shows the alarms generated by the system. At the device level, the connectivity module should support logging, because logging is usually separated from the microcontroller app by a serial interface (UART or SPI). On the cloud side, the server’s logging could be obtained by a number of third-party tools, such as Papertrail by AirBlade Software, for instance.

A proper device-level testing process involves a test bench which exercises the system. This can involve using a dashboard environment, which allows for the management of devices and supports an automated approach. Using the dashboard, a test bench sends commands to a device under test (DUT). The DUT passes information to the cloud via the communication module, while the test bench simulates the microcontroller application. Those packets are traced by the device and cloud logging mechanisms, and the resulting data can be validated using the dashboard or automatically via the test bench.

The cloud will also send back preformatted test data to the device under test which it will receive and send to the test bench. A test agent may be used in the DUT for this purpose of testing the system beyond the normal microcontroller application.

Automated tests are implemented in scripting languages, allowing ample flexibility and simple interfacing with external tools, such as Curl and Expect, and test management applications, such as TestRail, for reporting results.
Key Components of an IoT Platform

Figure 1. Key components of an IoT platform
Target Test Areas

There are six target test areas at the device level, they include:

1. **Device Properties:** Device properties, which represent the data model of the device, presented by various data types, are saved to and fetched from the cloud, for validation of their integrity. Ayla supports different data types, including numeric (boolean, integer, decimal, etc.) and text (strings, files, etc.). The tests validate the data as well as ensure that all of the data types are saved properly and retrieved from the cloud.

2. **Serial Protocol:** The serial protocol, which runs across SPI or UART and supports the reliable data exchange between the microcontroller and the module, is tested according to the specification. Data is sent on packets with the sequence numbers and control sum, and also provides a confirmation mechanism for the module. The test verifies data is transmitted in sequence and properly acknowledged. This is important, because the module sends alarms regarding the state of connectivity to the host; if it doesn’t work properly, data may be lost between the module and the host.

3. **Schedules:** Schedules control the device’s properties. They usually reside on the device, but can be configured in the cloud. It’s important to validate that multiple schedules are not interfering with each other or with the other services, such as OTA updates. Also, the provider should validate that when the time on the device is updated, schedules behave as desired. For example, if the time advanced by five minutes, the events scheduled for this interval should be executed in a sequence or dropped depending on the requirement.

4. **Power Modes:** Power consumption is essential for a wide range of embedded devices, so the communication modules usually support a low-power mode of operation. Thorough testing of switching between the power modes, as well as validation of the connectivity, are essential to ensure connectivity isn’t degraded when switching to the low-power modes.

5. **Connectivity:** The connectivity test validates the wireless configuration, the ability to switch between AP and station modes, and the ability to connect to the various WLANs with different security settings. In the field, the devices are connected to various Wi-Fi access points from different vendors. Wi-Fi chipmakers already test their devices for compliance with standards; however it's important to validate that evolving standards don’t introduce new problems.

6. **OTA:** The Over-the-Air test verifies that the host (customer) and module images can be successfully deployed from the cloud to the device. Because customers may be using different releases of the firmware, all upgrade-downgrade paths should be tested. OTAs should be tested against a compatibility matrix; to ensure all covered firmware releases are upgradable to the new release. Additionally, they must be tested to ensure that they can be downgraded successfully to the original release, if needed.

Cloud-Level Testing

For cloud-level testing, all functional, integration, system, and API tests should be automated and run on a nightly basis, using continuous integration tools, like Jenkins, so teams can identify and fix issues quickly and efficiently. Scalability and performance testing are also critical to ensure the cloud infrastructure is scalable and performing at the highest possible level.

Performance, Reliability, and Scalability Testing

A critical aspect of the cloud is how it scales to accommodate user demand. This involves understanding how people and devices are interacting with the cloud service to get a sense for the user experience. If more units are supported by the service, will it degrade performance? Two of the most common load tests for scalability are the
“Christmas Day” load where large volumes of new devices are coming online for the first time and the “predictable peak” where a certain event occurs at a predictable time (e.g. turning on a light at dusk). An example would be to simulate a million users accessing the service at once and observing how the service responds. Does it meet the service level agreement expected by the manufacturer? What’s the maximum level of latency that occurs if you scale from a thousand to a million units? Are you still within the stated latency parameters? If the servers are resilient enough, the customer should experience at most a “glitch,” but should still be able to continue using the connected device.

Security Testing

Another critical consideration is security. How secure is the cloud service? This is particularly important with connected devices that collect and store sensitive information. Consumers need to feel confident that hackers won’t be downloading personal information from their connected appliances, home security systems, thermostats, and so on. The purpose of security testing, then, is to ensure data flowing through the cloud and dispersing to all interfaces and endpoints is protected.

In testing for security, understanding the data flow is important. You cannot anticipate where data may be coming in or out of the cloud, or who’s accessing it. Here are some questions to consider when testing security:

- **Are all interfaces to the cloud secure?**
- **Is the cloud following security best practices?**
- **If I perform an injection attack on the cloud, how does the system handle it?**

One way to test cloud security is through penetration testing. This involves locating vulnerabilities in the infrastructure to ensure that the firewalls, and other security measures, are set up correctly and doing their job. There may be unforeseen vulnerabilities in the infrastructure, making it susceptible to distributed denial-of-service (DDoS) attacks.
**Data Governance**

As data flows in and out of the cloud, it should be subject to a consistent set of policies that stretch across the entire IoT system; from the application interface, to the cloud service, and to the end device. There are numerous ways in which consumers and OEMs are in contact with the cloud service, and you’re only as secure as your weakest link.

Some providers will engage in regular third-party audits to ensure compliance with data governance standards.

**Data Privacy Testing**

Data privacy goes hand-in-hand with security and data governance; but it’s not just good practice. In many places such as Europe, it’s considered a human right. So, this form of testing is especially important for connected device manufacturers operating globally. In Europe, the customer can legally pursue the service provider for any compromise in data privacy, and serious ramifications can ensue. It’s important to look at the cloud holistically and make sure all aspects comply with a minimum set of data privacy standards. Of course, doing so requires extensive knowledge about how data is flowing through the IoT system, the infrastructure, and any related legal requirements. All of these factors should feed into the testing and verification for a truly trusted cloud service.

**OWASP**

The Open Web Application Security Project (OWASP) is considered to be the benchmark for security and web applications. Its Application Security Verification Standard (ASVS) contains a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, and even consumers to define what a secure application is. At a minimum, your cloud service provider should strive to meet this standard.
Mobile-Level Testing

It’s not uncommon for many manufacturers entering the IoT market to be new to mobile technology. Often, these companies specialize in building devices—many of which have historically had no Internet connectivity. So, when they initially develop these products, mobility isn’t a consideration. The refrigerator sits in your kitchen; the thermostat is affixed to your wall. Such market players are experts in product design and implementation, but when it comes to connecting those products to the web, expertise may be lacking.

However, mobile capabilities bind everything together in an IoT environment, because it’s through mobile access that consumers interact with their connected devices. User experience is therefore a critical aspect of testing as it represents the manufacturer’s brand. It’s important to note that some bugs that appear on the mobile app can actually be attributed to problems on the device or cloud, so its critical to investigate the root cause of any errors that may appear on the mobile app.

Mobile-based testing aims to find and identify problems and inconsistencies within and between the mobile app user interfaces, and back-end systems, as well as problems and inconsistencies in their interactions with the cloud service and hardware components. With a focus on usability, mobile testing combines a mixture of testing methods to address the complex, often subjective nature of the mobile app user experience, which begs an exploratory “gray box” approach to test case design and implementation.

Some vendors use a multitude of mobile devices and emulators across multiple platforms to test edge and corner cases of mobile hardware compatibility, GUI rendering, and the performance of custom application builds on all major hardware/OS combinations before any release.

Backward Compatibility Testing

Apple and Google are constantly revving their operating systems, and you need to think about how your SDKs accommodate those changes. They should be backward compatible and forward-looking at the same time. Because it’s impossible to support all the versions that have ever been released, you should be considering which versions they’ll support.

Monitoring Tools

Some cloud providers use application performance monitoring tools that go beyond basic monitoring to track the performance and behavior of the CPU, memory and other aspects of the IoT system to ensure optimal performance and responsiveness.
Testing and compatibility of features may become compromised if you try to support all of them. A best practice is to eliminate the bottom 10 percent of OS versions. Testing to support that bottom 10 percent can be expensive, and you’ll be compromising features and spending too much time trying to make it work. In fact Apple actively encourages customers to continually update their iOS to the latest version and its common for iOS developers to support only the current and one previous version of the OS.

Backward compatibility testing ensures that when new libraries become available, the technology is compatible with major releases. Additionally, native OS testing over a range of versions, manufacturers, and products will ensure the apps work as OSs evolve.

**CRUD Testing**

Create, Read, Update, and Delete (CRUD) testing ensures your RESTful API works as expected over any network. Additionally, you must test a variety of mobile app features to ensure the success of your IoT platform:

- **Account management:** Creating and updating an account; updating, resending, and deleting passwords
- **Sign-in management:** Credentialing, username and password, single sign-on across multiple accounts
- **Data management:** Protocol type testing, create, read, update, and delete (CRUD) testing
- **ZigBee:** Data waves, nodes, groups, bindings, and themes
- **Share management:** Creating and deleting shares
- **Support schedules and timers per device**
- **Support contacts and notifications management**
- **Local area network-connect mode**

**Lifecycle Testing**

Lifecycle testing involves testing real-world usage scenarios—the way an app works from start to finish, when you send requests or move the app to the foreground or background of your device. Many apps that pass CRUD testing will crash during lifecycle testing when tested across a range of OS releases and versions.

**Here are some actions to test:**

- **Terminating Requests:** It’s important to consider terminating requests. For example, if you make a change to your thermostat and then move the app to the background, you want to be sure the temperature still changes, and the request doesn’t hang or terminate.

- **Timing Out Requests:** This needs to be tested in both WAN and LAN mode. Imagine you’ve made a request to raise the temperature on your thermostat, but for some unknown reason, the app cannot connect to the server or device. In many cases, if there is no response, the request will fail. This scenario can be extremely frustrating to the end user.

- **Real-world Scenarios:** You may have intended or predicted uses for your app, but consider how people may really use the app in practice. Test those scenarios, as well. Sometimes it is difficult to predict complex app use—try random use scenarios—does it cause a problem, is it a possible choice?

**End-to-End Testing**

Probably the most critical aspect of QA in IoT is end-to-end testing, during which QA teams run automated tests, orchestrating test cases involving all three components of the IoT Platform: device, cloud, and app.

Although it’s important to test each component—device, cloud, and mobile—it’s equally important to test how the three components work together.
Extensive testing for each separate component can uncover issues that must be addressed in the other components; a change to mobile will necessitate a change to the device, and so on. For example, changing the temperature on your thermostat from your mobile device initiates a change in the cloud service, which initiates a change in the device. Changes to any piece of the system propagate throughout the entire system. It’s a three-way conversation and endpoints can change dynamically. This interdependency is what makes IoT testing so unique, and so complex.

In addition to testing all of the components of an IoT solution, it is also critical to test the solution in multiple conditions. For example, different networks can behave differently, so it’s important to test the solution across different networks, routers and operators.

**Field Trials**

Generally the final step in proper QA and testing is to conduct field trials. Manufacturers should plan to leave time in the development schedule for field trials and testing. When you have a product you believe is fully functioning and ready for customers, test it among a small group of 10-20 users in each region you plan to deploy product. Each trial group can download the app, unpack, and install the product and do all the steps a regular customer will do. Once they’ve used the product for a number of days, say 3 to 5, collect feedback. Incorporate that feedback into your product, and then release the next version to a slightly larger group. Field-testing such as this takes 1-2 weeks per cycle but is absolutely essential to ensuring your product works as intended when it’s finally released to consumers.

**Summary**

Contemporary and agile IoT platforms are complex, scalable, and high-performance systems that require a sophisticated QA process, grounded in automated testing.

Make sure your IoT platform provider has invested substantial resources in QA, which is added value that they can pass on to you, their customer.
Ayla Networks provides the industry’s first Agile IoT Platform, accelerating development, support, and ongoing enhancements of connected products for the Internet of Things. Ayla’s software fabric runs across devices, cloud, and apps to create secure connectivity, data analytics, and feature-rich customer experiences. Offered as a cloud platform-as-a-service (PaaS), Ayla's flexibility and modularity enables rapid changes to practically any type of device, cloud, and app environment.