**Proposal for PowerMockito in RefApp\_Android**

@author – Ritesh Jha

@data: 4th Aug 2017

[**Feature 18596:**](http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/_workitems/edit/18596)**Deliver an easy to use integration between Apteligent SDK and AppInfra in RefApp**

While working with Apteligent, encountered the major part is handling its APIs, which is all “**static**” methods. Same time, we wanted to retain RobolectricRunner as well. Came across PowerMockito library which suits our intention very well.

Implemenated some of the test cases using PowerMock, which is interesting to see in file ApteligentBroadcastReceiverTest.java

**Benefits:**

1. The great benefit of Powermock is that it supports Mockito and Easymock APIs. That means that everything is possible about Mockito, is valid for Powermock.
2. Static methods can be tested. So existing all possible static methods can be automated very well.
3. Supports Mockito-style mocking.
4. Mocks constructors, private and final methods.
5. Singleton classes can also be mocked: <http://www.technovillage.org/?p=139>

**Implemenations:**

**Gradle dependencies:**  
testCompile **'org.powermock:powermock-core:1.6.6'**testCompile **"org.powermock:powermock-module-junit4:1.6.6"**testCompile **"org.powermock:powermock-module-junit4-rule:1.6.6"**testCompile **"org.powermock:powermock-module-junit4-rule-agent:1.6.6"**testCompile **"org.powermock:powermock-api-mockito:1.6.6"**testCompile **"org.powermock:powermock-classloading-xstream:1.6.6"**

**Mapping dependencies before test class**

@PowerMockIgnore({ **"org.mockito.\*"**, **"org.robolectric.\*"**, **"android.\*"** })

@PrepareForTest(Static class name .**class**)

Please check ApteligentBroadcastReceiverTest for more details.