Decreasing server build time

- Nuances of dealing with Spring contexts in tests - starting application context with too generic definitions is a very big contributor to slow builds. I have a few specific ideas here and if you agree on this item, I can elaborate later. Another benefit, this will help in creation of context files for new tests, which is not an easy thing currently

- Collectively identify all existing IT having as goal clear separation between Unit Tests and Integration Tests

- Parameterize execution of IT, so that we can choose when to run them. I think we have already all provisions to do this

- Having done items above, we can agree to trigger UT enabled verification builds automatically with every 10 commits into integration branch (for example), while IT enabled builds will run only a few times a day

Integration Testing improvements

- Move all ITs into the separate module (pcap-integration-tests). It means extracting them from individual modules where they are mixed in currently with main code. This new module will have all our “main” artifacts available only as maven dependencies. As such, we can easy choose version of artifacts to Test in Integration mode. This separation will give true meaning to the name Integration Test

- Enable triggering of those IT by Devs or QA on demand - as a suite, or individually. Interface to invoke them could be a simple internal-only REST of Struts controller hosted in dedicated Tomcat instance. Controller will wire all required “Services” or “Managers” available for testing, and will invoke them based on the input it receives from caller-test-driver. In this scenario, we will gain true flexibility in choosing method to invoke tests - from curl command on command line to Python or Groovy scripting, having Java as default. Choosing scripting languages (that are by definition more natural and easier to use) will allow us to train QA to write those tests more efficiently than in Java, and have interns contribute faster. All they need - scripting environment and number of jars (including IT jar) deployed to Tomcat harness. Results will be posted into Qmetry

Executing and baselining existing SQL in the test mode

Have a simple (few classes) framework that will execute DAO methods or individual native SQLs, capture execution time as well as Explain Plan. Searching through the string with Explain Plan for the “red flag” tokens like Full Table Scan, etc. Produce report of suspects for review (FTS is not bad sometimes). Elapsed times will

be logged in a spreadsheet and compared with baseline. Database for running test like this should be known and stable db (like QAStaging for example)