**Live Assist QA Mocha functional regression test integration with AnthillPro, Perforce & Rally**

1. ***About AnthillPro, Perforce and Rally***
2. The AnthillPro is an automation tool for building a product codes and testing which can be triggered by DEV, release engineer and QA.

Role: As a central QA test automation tool used to deploy the new build, copy the test set to test server, triggered the test, collect the test result, generate report and send out result (such as email)

1. The Perforce is a sort of central [database](http://en.wikipedia.org/wiki/Database) and a master [repository](http://en.wiktionary.org/wiki/repository) of [file](http://en.wikipedia.org/wiki/Computer_file) versions for product implementation codes from developer and also codes for testing from QA.

Role:  As QA test code source depot, store all the QA test automation scripts and test cases.

1. The Rally is an agile management tool for development team and monitoring project progress and it also can be used for test result report.

Role: As QA Agile work progress tool, collect information and provide to user like: feature user stories, quality/defects report, QA test set/test cases, test result and report

1. ***Basic intro and work flow***

6

5

Live Assist Server

7

Rally

2

3

4

1

Perforce

QA Test Machine (2)

AnthillPro

Figure 1.1 Basic test integration work flow and tools relationship

The QA Mocha functional regression test running on two test machines:

* AHP Live Assist server: New Live Assist build deployed on this server (CentOS6.4 + JDK7)
* AHP QA api/gui test machine: Live Assist QA Mocha functional regression test set (Windows 2003 server + JDK7 + NodeJS + Mocha + QA test set)
* AHP QA jump server for Live Assist gui test using protractor (windows 2003 server + QA test/task trigger batch files)

The work flow and relationship for the Live Assist test integration with AnthillPro, Rally and Perforce:

S1 (Step 1): Deploy & setup new build to QA Live Assist test server via AnthillPro (RPM or ZIP artifact)

S2: Request/setup Perforce to get the latest QA test set from Perforce QA depot

S3: Copy latest QA test set to QA test machine

S4: Trigger/start the QA regression test

S5: Regression test run between Live Assist server with the new build and QA test machine with the latest QA test set

S6: Test results send to AnthillPro in Junit format

S7: Post the test result to Rally via QA tools (Junit format -> CSV sheet -> Rally), and reported by Rally Test summary

Note (\*): The ‘QA Test Machine’ setup here for Live Assist project has two type regression tests:

1. API regression test -> one regression test machine
2. API + GUI using protractor/selenium regression test -> one AHP jump server/slave server + one regression test machine
3. ***The basic steps for integration with AnthillPro***
4. Install AnthillPro agent on test server/machine, and add them to AnthillPro:

AHP agent installation and upgrade guide: <https://nuance.jiveon.com/docs/DOC-34337>

1. QA test server setup for AHP:

* Live Assist server:
  + CentOS6.4
  + JDK 7 (not JRE 7 requested by Live Assist server)
* Live Assist QA test machine
  + Windows 2003 server
  + Install/setup nodejs, mocha and all necessary modules
  + Install/setup browsers: Firefox (>=26), Chrome

1. Assign specific access or task permissions from AnthillPro administrator for specific project, permissions like:

* create/edit workflow, Job, Work Dir, notification/email and resources
* Recipient Generator under 'System', create new recipeent group
* Notification Schemes under 'System' tab, create new email notifcation scheme with specific message template, such as 'Simple Email Template with JUnit Summary'

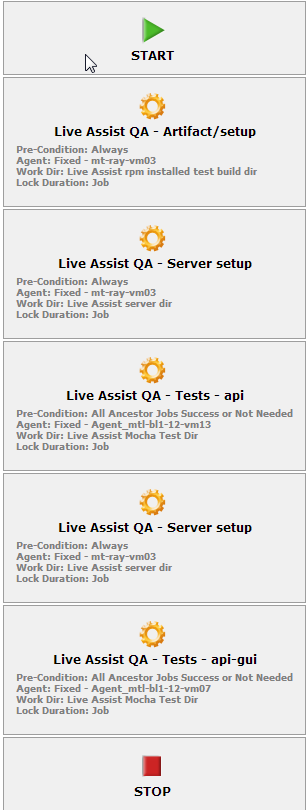
1. Create new Live Assist test workflow under Live Assist project in AHP: Administration -> Nina -> Live Assist
2. Create and add job steps under the new created AHP workflow
3. Test with new project build to verify if above AnthillPro setup works.

1. ***Live Assist QA test jobs and work flow setup detail in the AnthillPro including integration with Perforce & Rally***

The Live Assist QA Anthill pro automation regression test setup has 3 test machines setup with two 4 type of jobs:

1. Live Assist QA – Artifact/setup job (Cent OS6)
2. Live Assist QA – Server Setup (CentOS6)
3. Live Assist QA – Tests – api (win2k3)
4. Live Assist QA – Tests –api-gui Tests (Win2k3)

The jobs workflow and invoking sequence like this:



1. Live Assist QA – Artifact/setup job (Cent OS6, Live Assist server: mt-ray-vm03)

* ‘Set WorkDir’ – setup working directory on QA Live Assist server (/usr/local/Nuance)
* ‘Cleanup Live Assist sever’ – customize with AHP shell script:
  + ‘pkill’ command to kill specific Live Assist Java processes (if running)
  + ‘rpm’ command to uninstall Live Assist old build (if has)
  + ‘rm’ command to remove old Live Assist builds/files (if has)

'pkill' -f 'java .\*liveassist-audioserver'

'pkill' -f 'java .\*liveassist-server'

'sleep' 2

'rpm' -e liveassist

'sleep' 5

'rm' -fr liveassist/

'rm' -fr liveassist\*.rpm

* ‘Live Assist Artifact –rpm package’ deployment
  + Using AHP ‘Resolve My Artifact’ to copy the Live Assist new build to specific QA Live Assist test server from the AHP build system. The ‘Artifact Set’ select as ‘APP-LINUX-64’
* ‘Install rpm package’
  + Uinsg AHP shell script with rpm command to install Live Assist new rpm package and re-setup database (in case if there is any changes in the new build) on the AHP QA Live Assist test server:

'rpm' -U liveassist\*.rpm

'mysql' -u root -e "DROP DATABASE liveassist;"

'mysql' -u root -e "DROP USER 'liveassist'@'localhost';"

'mysql' < /usr/local/Nuance/liveassist/liveassist-server/bin/create\_db.sql

'mysql' -u root -e "CREATE USER 'liveassist'@'localhost' IDENTIFIED BY 'liveassist';"

'mysql' -u root -e "GRANT ALL PRIVILEGES ON liveassist.\* TO 'liveassist'@'localhost' IDENTIFIED BY 'liveassist';"

'mysql' -u root -e "FLUSH PRIVILEGES;"

1. Live Assist QA – Server setup (CentOS6, Live Assist server: mt-ray-vm03)

* ‘Set WorkDir’ – setup working directory on QA Live Assist server (/usr/local/Nuance/liveassist)
* ‘Stop and cleanup Live Assist sever’ – AHP shell script:

'service' nuance-wd stop

'sleep' 5

'pkill' -f 'java .\*liveassist-server'

'pkill' -f 'java .\*liveassist-audioserver'

* ‘Start Live Assist server’ – AHP shell script:

'su' -s /bin/bash nuance --session-command liveassist

* ‘Start Live Assist audio server’ – AHP shell script:

'su' -s /bin/bash nuance --session-command liveassistaudio

1. Live Assist QA – Tests - api (Win2k3, regression test server: mtl-bl1-12-vm13)

* ‘Set WorkDir’ – setup working directory on QA Live Assist test machine (C:\LiveAssist\_test\test)
* Cleanup old test set –AHP shell script:

del c:\LiveAssist\_test\test\QA\_TEST\\*.\* /f/q

del c:\LiveAssist\_test\test\QA\_TEST2\\*.\* /f/q

del c:\LiveAssist\_test\test\reports\\*.\* /f/q

del c:\LiveAssist\_test\test\\*.bat /f/q

* Get/Sync-up Test set from Perforce –AHP shell scripts:

p4 set P4PORT=10.3.4.77:6826

p4 set P4USER=qamgr

set P4CLIENT=c:\temp\_p4v

mkdir C:\temp\_p4v

cd C:\temp\_p4v

p4 client -o -S //entrd\_liveassistqa/main | p4 client -i

p4 sync -f

copy c:\temp\_p4v\LiveAssist\_test\test\QA\_TEST\\*.\* c:\LiveAssist\_test\test\QA\_TEST /Y

copy c:\temp\_p4v\LiveAssist\_test\test\QA\_TEST2\\*.\* c:\LiveAssist\_test\test\QA\_TEST2 /Y

copy c:\temp\_p4v\LiveAssist\_test\test\reports\\*.\* c:\LiveAssist\_test\test\reports /Y

copy c:\temp\_p4v\LiveAssist\_test\test\\*.bat c:\LiveAssist\_test\test /Y

* ‘Run special API Mocha regression test batch script for AHP’ – AHP script:

call run\_AHP\_api-test.bat

The detail of the API regression test batch file:

setx XUNIT\_FILE "c:\LiveAssist\_test\test\reports\TEST-xunit.xml"

setx LOG\_XUNIT "true"

start "Run Mocha test" /HIGH /min cmd /k mocha --reporter xunit-file --ui bdd QA\_TEST --timeout 3600s --slow 250000

REM -----------check and wait for the test report XML generated then exit---------------

@echo off

:CHECK

for %%a in (.\reports\TEST-xunit.xml) do (

if %%~za equ 0 (

REM %%~na is empty - wait

sleep 10

goto CHECK

) else (

sleep 2

@echo on

echo %%~na is not empty

ECHO Test done, copy report, exit and cleanup

move c:\LiveAssist\_test\test\reports\TEST-xunit.xml c:\LiveAssist\_test\test\reports\TEST-xunit-api.xml

@echo off

taskkill /F /IM "cmd.exe"

exit 0

)

)

1. Live Assist QA – Tests – api - gui (Win2k3, Jump server: mtl-bl1-12-vm07 -> test server: mtl-bl1-12-vm13)

* ‘Set WorkDir’ – setup working directory on QA Live Assist test machine (C:\LiveAssist\_test\test)
* ‘Cleanup old test result’ – AHP shell script:

del c:\LiveAssist\_test\test\test\_outputs\1\*.log /f/q

del c:\LiveAssist\_test\test\reports\TEST-xunit-api-gui.xml /f/q

del c:\LiveAssist\_test\test\reports\TEST-xunit.xml /f/q

* ‘Run speical Mocha batch script for AHP’ – AHP shell script:

call start\_ahp\_api-gui-test.bat

The detail of the batch file which triggered test from jump server to Mocha test server for gui regression test:

1. start\_ahp\_api-gui-test.bat

REM ----- get local date/time as task name ----------

@echo on

For /f "tokens=2-4 delims=/ " %%a in ('date /t') do (set mydate=%%c-%%a-%%b)

For /f "tokens=1-2 delims=/:" %%a in ("%TIME%") do (set mytime=%%a%%b)

echo %mytime%

REM -------- remove mapped folder/driver (if has) before test run ----------------

net use z: /delete

REM ------- map folder from QA test AHP agent machine --------

net use z: \\mtl-bl1-12-vm13\c$\LiveAssist\_test\test\reports L0cus1 /user:root

move z:TEST-xunit.xml z:TEST-xunit.bak

REM --------------start schedule task -------------

start "Start remote AHP api gui regression test" /min cmd /k remote\_ahp\_api-gui-test.bat %mytime%

REM -----------check and wait for the test report XML generated then exit---------------

@echo off

:CHECK

for %%a in (z:TEST-xunit.xml) do (

if %%~za equ 0 (

REM %%~na is empty - wait

sleep 10

goto CHECK

) else (

sleep 2

@echo on

echo %%~na is not empty

echo Test done, copy report, exit and cleanup

copy z:TEST-xunit.xml c:\liveassist\_test\test\reports\TEST-xunit-api-gui.xml

move z:TEST-xunit.xml z:TEST-xunit-api-gui.xml

copy z:TEST-xunit-api.xml c:\liveassist\_test\test\reports\TEST-xunit-api.xml

sleep 1

@echo on

net use z: /delete

taskkill /F /IM "java.exe"

taskkill /F /IM "cmd.exe"

exit 0

)

)

1. remot\_ahp\_api-gui-test.bat

@echo on

REM -------- script started to run remote machine tasks/scripts -------------

set machine=mtl-bl1-12-vm13

set file=\\mtl-bl1-12-vm13\c$\LiveAssist\_test\test\run\_AHP\_api-gui.bat

set user=root

set userPwd=L0cus1

set executeAs=%user%

set executeAsPwd=%userPwd%

REM ---- clean up task (if have any) before new test -----

schtasks /delete /tn \* /f /S %machine% /U %user% /P %userPwd%

sleep 3

REM --- Run the batch file as scheduled task on remote machine and accept the command line argument %1 as task name! -----------

schtasks /Create /S %machine% /U %user% /P %userPwd% /RU %executeAs% /RP %executeAsPwd% /SC ONCE /TN %1 /TR %file% /ST 00:00:00

schtasks /Run /S %machine% /U %user% /P %userPwd% /TN %1

1. run\_AHP\_api-gui.bat (from Mocha test server, mtl-bl1-12-vm13):

ECHO start selenium standalone server for web test driver

start "Start selenium server for protractor on api gui test" /min cmd /k c:\LiveAssist\_test\test\start\_selenium.bat

sleep 5

ECHO Mocha test run

setx XUNIT\_FILE "c:\LiveAssist\_test\test\reports\TEST-xunit.xml"

setx LOG\_XUNIT "true"

start "Run Mocha test" /min cmd /k mocha --reporter xunit-file --ui bdd QA\_TEST2 --timeout 3600s --slow 250000

sleep 5

ECHO Check Test done

REM start "Check Test Done script" /LOW /WAIT /min cmd /k chk\_gui\_test\_done.bat

REM sleep 1

REM -----------check and wait for the test report XML generated then exit------------

@echo off

:CHECK

for %%a in (.\reports\TEST-xunit.xml) do (

if %%~za equ 0 (

REM %%~na is empty - wait

sleep 5

goto CHECK

) else (

sleep 1

@echo on

echo %%~na is not empty

ECHO Test done, copy report, exit and cleanup

copy c:\LiveAssist\_test\test\reports\TEST-xunit.xml c:\LiveAssist\_test\test\reports\TEST-xunit-api-gui.xml

sleep 1

@echo off

taskkill /F /IM "java.exe"

taskkill /F /IM "cmd.exe"

exit 0

)

)

* ‘Get the Live Assist test build #’ - using AHP ‘Resolve My Artifacts’ to copy ‘/log’ folder from Live Assist build system to local QA test machine (contain version.txt file)
* ‘Update Rally config’ – using Perl script to parse the version # and added to rally properties config file (for posting AHP test result to Rally)

perl Buildname-to-Rally-Config.pl

The perl script detail:

use FileHandle;

#use strict;

my $build\_loc = "c:\\LiveAssist\_test\\test";

my $version\_filename = "version.txt";

my $rally\_filename = "rally.properties";

my $version\_file = $build\_loc."\\".$version\_filename;

my $rally\_file = $build\_loc."\\reports\\".$rally\_filename;

my $build\_version = "";

my $rally\_temp = "";

print "\nVersion file: $version\_file\nRally config file: $rally\_file\n";

#get the build # from version file

open(IN1, "+<$version\_file") || die "Cann't open $version\_file: $! \n";

if(IN1 != -1){

$build\_version = <IN1>;

chomp($build\_version);

}else{

$build\_version = "Unknown-build";

}

close(IN1);

print "\nGot build name: $build\_version\n";

#read content from rally.properties file to temp

open(IN2, "+<$rally\_file") || die "Cann't open $rally\_file: $!\n";

while(<IN2>){

$rally\_temp=$rally\_temp.$\_;

}

close(IN2);

print "\nRally properties file content:\n $rally\_temp \n";

#add build version

my $search\_key = "build=.+";

my $replace\_key = "build=".$build\_version;

if($rally\_temp =~ s/$search\_key/$replace\_key/){

print "Replace build line:\n$rally\_temp\n";

#write temp content to rally.properties

open(IN3, ">$rally\_file") || die "Cann't open $rally\_file: $! \n";

print IN3 $rally\_temp;

close(IN3);

}else{

print "Replacing build version failed\n";

}

* ‘Send test result to Rally’ – AHP shell script:

call run\_win.bat

The batch file detail (convert xml result -> csv format -> export and send to Rally):

java -jar QA-REG-xml-to-csv.jar

java -jar QAExportToRally.jar

* ‘Test result report’ – using AHP report template with following patterns: “\*\*/TEST-\*.xml”

Note: The Mocha test result need to be reported as Junit format to AHP, we can use xunit-file module refer at: <https://github.com/peerigon/xunit-file>, and a patch QA customized to make it work with AHP: [\\mt-nasrops01\qa\LiveAssist\QA\_tech\xunit-file\_js\_patch](https://nuance.jiveon.com/docs/DOC-39883?sr=stream)

* Assign Status (success/fail) – using AHP ‘Assign Status’
* Email notification (optional) – using AHP ‘Email Publisher’ to send test result/report to specific recipients.

For the detail setup steps/workflows/scripts, please check at (Note: you may need to request the Live Assist project specific AHP permissions to visit from AHP admin): <http://mt-urban.nuance.com:8080/tasks/admin/project/workflow/WorkflowTasks/viewWorkflowDefinition?workflowId=1840>

Note: Most of above commands used in the job/workflow has impersonation settings which log in as ‘root’.

1. ***Test result report:***

The Live Assist Mocha functional test result can be reported by both AHP and Rally. The more info the better for user review the test results. Rally has better GUI with more info on the last 5 builds result comparison, and AHP provides concise report by email and has more info on success rate and test running time.

An example for this basic comparison can be checked from the following Live Assist Mocha functional test result report links (example on one of Live Assist iteration 5.4 build):

* AHP example on one of Live Assist iter5.4 build: <http://mt-urban:8080/jobs/artifacts/657/3183666/LiveAssist_QA_Functional_Test_Report/junitReport.html>

Test summary include: total, pass, fail, error report and also success rate and test running time

Test detail include: test case title/name, status, and extra details (optional), with each case running time

* Rally: <https://rally1.rallydev.com/#/14813812965ud/testfolders>, customized for Live Assist based on standard QA test report template, under ‘Live Assist’ -> ‘Quality’ -> ‘Live Assist – QA Test Report’, then select release -> ‘Live Assist 1.0 (POC), TestSet -> ‘QA-Regression’.

Test summary include: last 5 builds results/comparison, pass/fail criteria, build#

List of Test cases include: case ID, title/subject, description (optional), build # (last 5 builds), and results

For the QA test report setup in the Rally:

- ‘Quality’ -> ‘Add app’ -> ‘custom html’

- Copy the source HTML from perforce: //entrd\_qa/gang-test-dev/Rally\_tools/Rally\_Dynamic\_Reports/TestReport\_Last5Builds.html to the new app (HTML)

- Make title/name changes for Live Assist

- Save and refresh

- Send request to Rally project management to make this new QA test report page public

1. ***Reference and latest sample result links:***

* Live Assist work flow/Job steps setup for test automation with AnthillPro: <http://mt-urban.nuance.com:8080/tasks/admin/project/workflow/WorkflowTasks/viewWorkflowDefinition?workflowId=1840>
* Live Assist test results posted to Rally (select release -> ‘Live Assist 1.0 (POC), TestSet -> ‘QA-Regression’):  <https://rally1.rallydev.com/#/14813812965ud/custom/16358313963>
* Live Assist sample test plan/cases/result reported in the Rally:
  + Test cases: <https://rally1.rallydev.com/#/14813812965ud/testcases?tpsV=qv%3A0>
  + Test plan/test set: <https://rally1.rallydev.com/#/14813812965ud/testfolders>
* Live Assist QA test results posted to Rally shared QA tools (jar, configuration files and perl script): [\\mt-nasrops01\qa\LiveAssist\QA\_tech\AHP\_Rally\_integration](file:///\\mt-nasrops01\qa\LiveAssist\QA_tech\AHP_Rally_integration); [\\mt-nasrops01\qa\LiveAssist\QA\_Doc\AHP](file:///\\mt-nasrops01\qa\LiveAssist\QA_Doc\AHP)