Vermillion Automation Approach

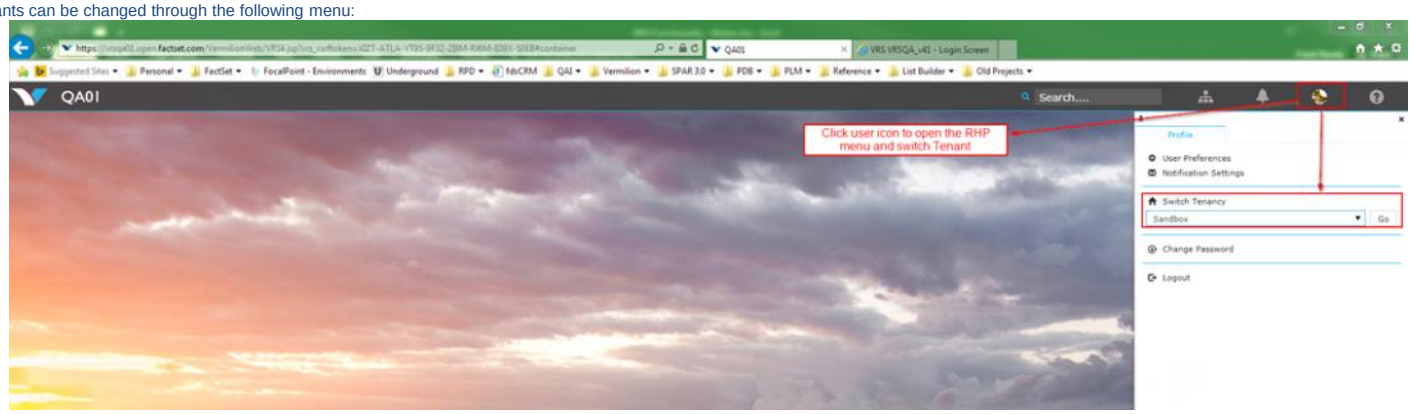
Wednesday, August 02, 2017 3:46 PM

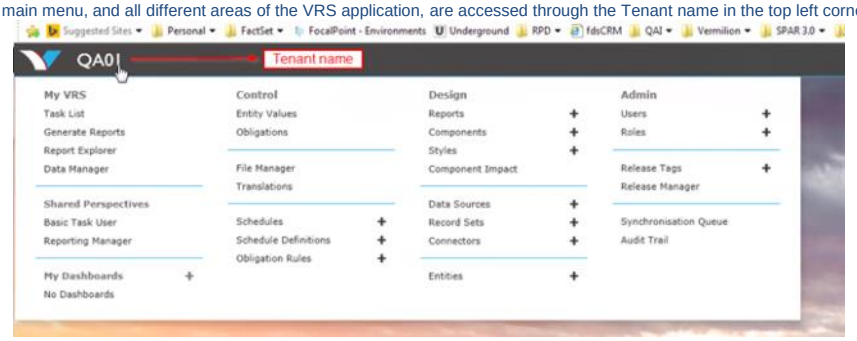
#1. Application Logon information

URL: https://vrsqa01.open.factset.com/VermilionWeb/vrs4gui/core/LoginVRS.jsp
Proxy UN: vrs-asp-user
Proxy PW: 22Redcat56
Login UN: rajul.shrivastava
Login PW: vrs123

About Application

A couple high level things:

- Our environment is further broken down into sections called “Tenants”, which contain isolated users/data/reports. We have 2 main tenants:
  - QA01 – used for test cases, data should be locked down
  - Sandbox – used for adhoc testing/exploring the application
- Tenants can be changed through the following menu:
  - 

- The main menu, and all different areas of the VRS application, are accessed through the Tenant name in the top left corner (dicking or hovering)
  - 

- VRS is a reporting application - to see these outputs, you can go to Design > Reports from the main menu. There are a couple example reports inside that you can generate

Basic Test Plans

- http://is.factset.com/justifier/TestManager/#/p3645/p5116/p5114/82771
- http://is.factset.com/justifier/TestManager/#/p3645/p5116/p5114/82770
- http://is.factset.com/justifier/TestManager/#/p3645/p5116/p5114/82826
- http://is.factset.com/justifier/TestManager/#/p3645/p5116/p5114/82825

#2. Automation Approach:

Current:

- Currently Vermillion QA team use to write component level test cases which would validate each widget of the application.
- They considered these Unit Level test cases. These test cases where written with Java Web-driver with selenium as driving tool
- These test cases were integrated with Ci process to run as part of build.
- Locators used CSS class which as per Engineer remained static through course of years.

Future State

In order to achieve a Test suite which have following attributes :

- Effective
- Fast
- Easy to maintain
- Scalable

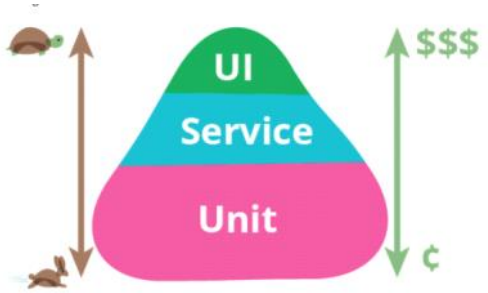
**** Decision Point:**
Due to Vermilion experience in terms of stability for CSS Locator at this point the need for Data-QA-ID is Not necessary

Ideal Automation State

The recommendation would be to have an testing Pyramid followed for Vermillion test suite <https://martinfowler.com/bliki/TestPyramid.html>

In the diagram for our references

- **UI =** UI End to end Automated + UI End to End Manual Test cases
- **Service Test cases =** Web Service Test cases



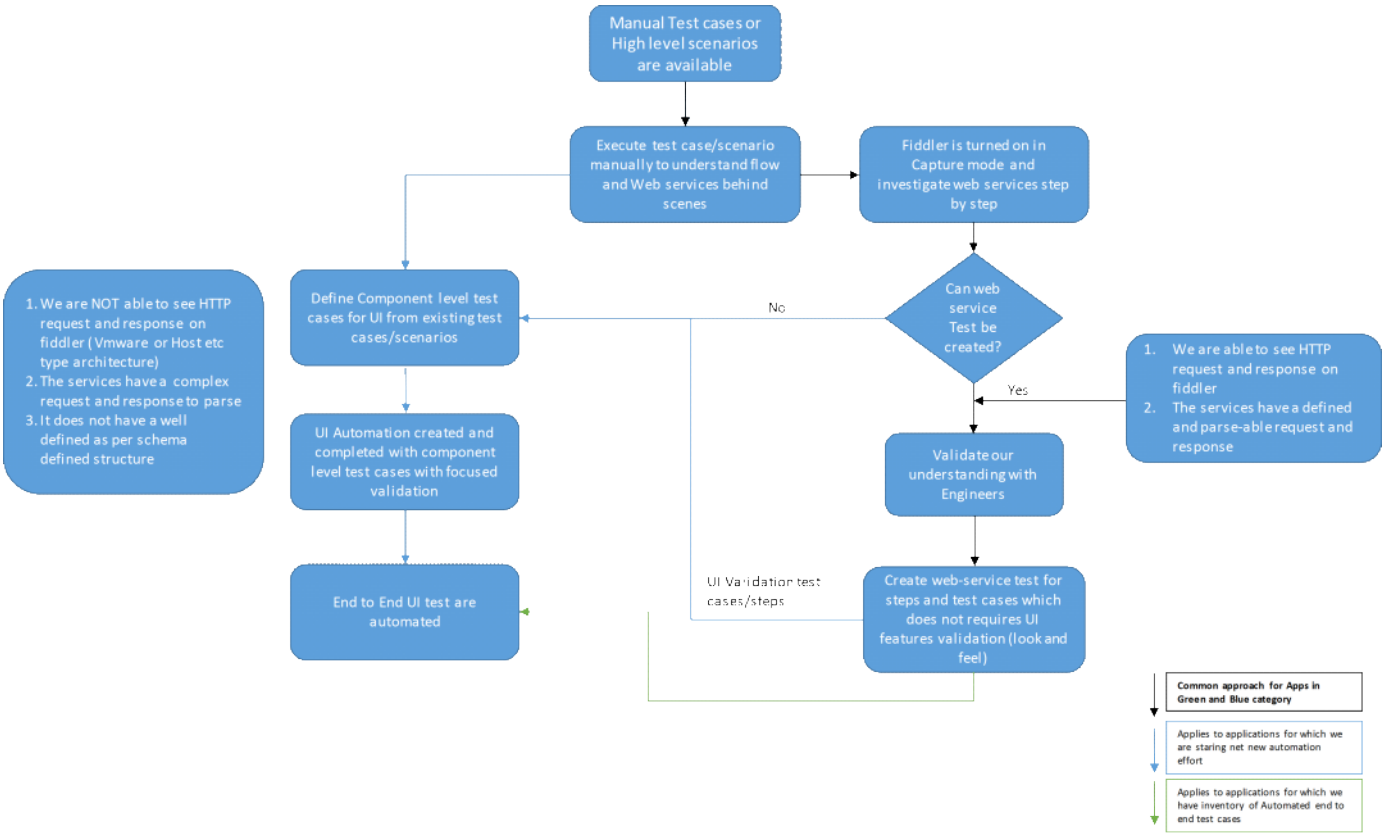
In the diagram for our references

- **UI =** UI End to end Automated + UI End to End Manual Test cases
- **Service Test cases** = Web Service Test cases
- **Unit Test cases** = Developer Unit test cases + Automated component Level Ui Test cases

This test suite will need to be designed in such a way that lower level test case successful completion will trigger next level of execution
Example: When Component level + Service test case pass UI end to end test trigger to run

Percentage distribution (Approx)
60 - 70 % Component level + Service level Automated test cases
20 - 30% UI end to end automated
10% - Manual end to end

Approach



Service Testing Analysis

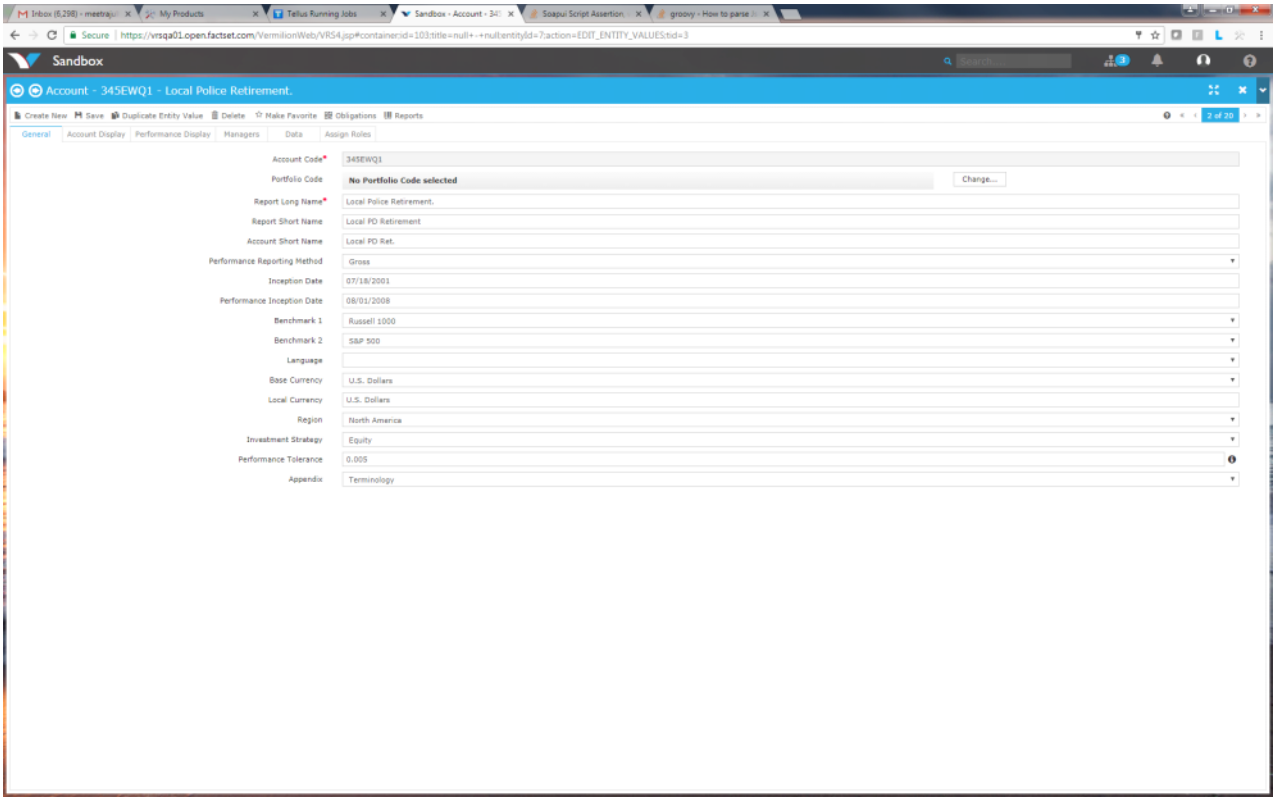
Analysis of test case - Modify-entity-values

Step : 713309
Below end point is a post which renders screen shown below.
https://vrsqa01.open.factset.com/VermilionWeb/dispatch/GetEntityFieldsAction?vrs_csrftoken=Y59E-CGAX-CCEV-UZDY-RK8K-5SQW-ZK9Q-YH79&gwt=y

Post Message:
7|0|8|https://vrsqa01.open.factset.com/VermilionWeb/VRS4/|798690A976015EA04C1165C0D7085BF1|com.gwtplatform.dispatch.rpc.shared.DispatchService|execute|java.lang.String/2004016611|com.gwtplatform.dispatch.rpc.shared.Action|com.vermilion.vrs.vrs4gwt.shared.core.action.GetEntityFieldsAction/3495938006|java.lang.Long/4227064769|1|2|3|4|2|5|6|0|7|8|H|8|Bn|1|

Headers that need to be passed are below < FOR ANY VERMILLION SERVICE TO WORK> :

Cookie
X-Fds-Chrome-Extension
X-Fds-Override-Name
X-Fds-Application-Client
X-GWT-Permutation
Content-Type



- The service generates a response which is a flat JSON without any hierarchy and it for all the tabs shown above like General, Account Display, Performance Display etc
- This response since being Flat do not have ***direct correlations between field name and value***
- Hence assert cannot be set by using Point and click provided by tool

Approach:
The response coming back will be have to be parsed by Json slurper code something like below (show validation on a metadata item shown in green)

```
import groovy.json.JsonSlurper

def response = context.expand( '${Accounts Code-Long Name#Response}' )
log.info (response)
slurperresponse = new JsonSlurper().parseText(response)
//log.info (slurperresponse.name)
log.info (slurperresponse[0])
//assert.slurperresponse[0] == "V2fmZ0X"
```

Vermilion-G
etFieldEnt...

Action Needed
QA to understand the response with Engineers for this end point, the correlation
Results: This test case can be automated using services for steps **713309, 713310,713313.** Rest are pure UI steps . But since the Web services are Flat structure, need to work with Engineers

**** Decision Point:** Need to work with engineers to understand the service Entity and Data correlations

To Do : Analyze services which renders PDF reports

Component level definition

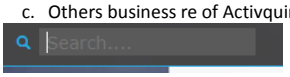
- Below are some illustrative examples of definition of component level test from UI perspective.
- These should be build such that they are independent of each of from validation stand point
- This will help write small and very focused test cases

Illustrative Examples only

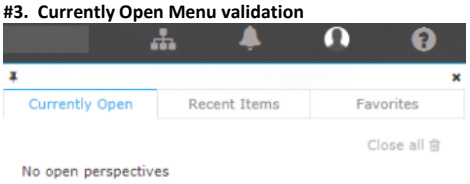
#1. Validation of appearance of below menu as user hovers over Vermillion brand image
   a. Validation of all sections like MY VRS, Control etc
   b. Validation of presence of Business functions within them
   c. Validation of result of action performed on them

#2. Search Function

a. Validation of presence
b. Validationity
c. Others business re of Activquirements if any



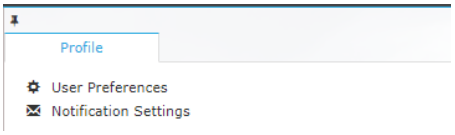
#3. Currently Open Menu validation



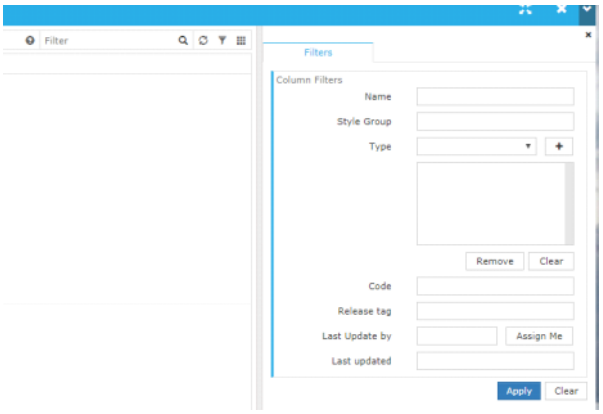
#4. Notification menu

#5. User Profile

a. User preference Test
b. Notification setting test



#6. RH Menu inside an Section Style Definition . Below menu is common across all the sections. Hence may be just 1 component level test by itself



#7 Control>> Entity Value>>

a. Validation of Load of screen
b. Click >> Account >> Edit
b.1 Click >> Account >> Edit >> Create
b.2 Click >> Account >> Edit>> Edit
b.3 Click >> Account >> Edit >> Delete