Few things before we start:

Naming agreement will be as follows <TBD>

Homepage: http://teams.infovantiv.com/sites/ODP/FraudVAS/SitePages/Home.aspx

Team Agreement: <TBD>

Integration steps for the Maven and Subversion setup:

# Section 1: First and foremost…

1. Peter Hoesly needs to be contacted and a ticket needs to be opened for access to the Teamforge server.
2. An email needs to be sent to Sunil Roy for access to the apigee edge repository after access to the Teamforge server has been granted.
3. Tortoise SVN needs to be installed off proxy (on mps-exec).
4. On network: <https://sl1floapsubvr01.infoftps.com/svn/repos/svn_cloud_deveco_repo/ApigeeEdge/trunk/proxies>
   1. This repository needs to be pulled into a file where ever you would like it to be.
5. <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
   1. needs to be downloaded and stored in C:\Program Files\Java
6. Acquire a webstorm license from either Peter Hoesly or Amanda Masters and download it from: <https://www.jetbrains.com/webstorm/>
7. Follow the below documentation for installing maven and integrating it with java.

Setting up java and maven to work together with SVN

***Java Setup***

1. Modify the JAVA\_HOME environment variable (or add it if it doesn’t exist) by opening system properties (  
   + Pause), selecting the "Advanced" tab, and clicking the "Environment Variables" button. It can be in either user variables or system variables. Set it to C:\Development64\java. Also make sure that %JAVA\_HOME%\bin is in your Path environment variable. If not, append it.

***Maven Install (Note to make this easy store maven in C:\Tools by making the Tools folder under the C drive)***

Most projects use Maven as a build utility. Follow these instructions to install it.

* Copy the Maven distribution archive (i.e. i.e. apache-maven-3.0.4) from [T:\ftps\_public\IntSvcs\Tools\Onboarding\Java\Build\Maven\Installation\Current](file:///T:\ftps_public\IntSvcs\Tools\Onboarding\Java\Build\Maven\Installation\Current)
* Unzip the distribution archive to C:\Tools. The subdirectory apache-maven-X.X.X (i.e. i.e. apache-maven-3.0.4) will be created from the archive.
* Add the M2\_HOME environment variable by opening up the system properties (WinKey + Pause), selecting the "Advanced" tab, and the "Environment Variables" button, then adding the *M2\_HOME* variable in the user variables with the value C:\Tools\apache-maven-X.X.X (i.e. apache-maven-3.0.4).
* In the same dialog, add the M2 environment variable in the user variables with the value %M2\_HOME%\bin.
* In the same dialog, update/create the Path environment variable in the user variables and append the value %M2%.
* Copy the .m2 folder from [T:\ftps\_public\IntSvcs\Tools\Onboarding\Java\Build\Maven\Settings\Current](file:///T:\ftps_public\IntSvcs\Tools\Onboarding\Java\Build\Maven\Settings\Current) and place it at C:\Documents and Settings\<userid>\ (e.g. C:\Documents and Settings\n123456\.m2).
* Open a new command prompt (WinKey + R then type cmd) and run mvn --version to verify that it is correctly installed.

How to use Rally 101

## Section 2: Rally use and navigation.

* First access rally: <https://rally1.rallydev.com/slm/login.op>
  + At the top right of the screen click the your profile and click “My Settings”
  + At the top right of this screen click “Edit Profile…”
    - Change “Default Workspace / Project:” To “Vantiv Enterprise: API – Value Added Services”
  + Click “Save & Close”
  + At the top right of the screen click the box with an arrow pointing to the left:
  + From here we left click “Track” -> “Iteration Status”
    - Here can see all the current stories the teams working on.
* User stories are a representation of what you are currently working on
  + For us we must have a release – Designated by Amanda Masters
  + An Iteration – Current work week the team is on
  + And a milestone (for example 20160316 VAS release 1 CERT)
  + The project must be set to API – Value Added Services
  + A detailed and defined description
    - “As a <blank> I want <describe the goal you are attempting to reach>”
  + Acceptance Criteria example
    - **Acceptance Criteria**:
      * API meets current logging standards set by Blazy Scott.
      * API properly parses information and rejects calls with missing information with a readable error
      * Proxy properly logs information
      * Proxy properly forwards to Forter
* Point systems are designated by the team via the Rally Cheat Sheet
  + Located: <TBD>
  + This is also part of the “Team Agreement” so we all know what kind of work we can handle
    - Full time is 8 points per sprint
    - Part time is 4 points per sprint
* Notes are used at your digression for things that have changed or need to be noted.
* Likewise for tasks they can be added to let the team and your scrum master know where you are currently in your work.

Understanding subversion (svn) and apigee deployment via maven!

## Section 3: Maven deployment and file structures and swagger 2.0.

* Since we release our API’s via Apigee we will be using apigees swagger 2.0 development studio.
  + <http://apistudio.io/> <- Read the tutorials and get started app to understand how this works.
* Under C:\Users\<your UID\Desktop\<where ever you placed the SVN file>\apigee\proxies\<your file> Upon copying a file structure from Fraud-VAS we will look at “Fraud-VAS\apiproxy\resources\node”
* Server.js will hold you server code it should be copied/left alone unless you need to edit it
* Package.json will hold your required node modules needed for your server
  + Under dependencies – make sure you add any that you use followed by its correct version
* Your controller should be placed under controllers/controller.js
  + A path is accessed by placing module.exports.<swaggers operationId name>
* This folders file structure is what we will follow

Main Folder

Pom.xml

* <artifactId>Fraud-VAS</artifactId> and <name>Fraud-VAS</name> should both be changed to your project name

apigeedDeploy.bat

* Deploy script that will automatically deploy your proxy

Apiproxy Folder

<service\_name>.xml

* Inside this file we add and remove policies and resources via <Policy></Policy> tags and <Resource></Resource>.
* Under “proxies\Fraud-VAS\apiproxy\proxies” There is a file default.xml
  + Under here we add and remove policies via #<policy\_name># in the request and response flows
  + Under the “<HTTPProxyConnection>” tag we have
    - <BasePath>/Fraud-VAS/v1</BasePath>
  + Here we use <BasePath>/<VAS\_service\_name>/v1</BasePath>
    - This ensures we don’t have conflicts in any other proxy pathing on apigee
* Under “proxies\Fraud-VAS\apiproxy\targets” we have another file named default.xml
  + Again under the <Request> tags we place #<policy\_name># to add policies to the flows in apigee
  + Under “<ScriptTarget>” we have “<ResourceURL>node://server.js</ResourceURL>”
    - This is what Apigee looks for to run a node server

Adding common policies to our proxies

* Here we will discuss how to add common policies to our proxies.
  + Under the “proxies\Fraud-VAS” folder there is a “pom.xml” file
  + This file will help name and deploy our proxy along with also adding plugins to it.
    - Under this file we have a <build> tag this tag lets the deployment process know where to look for policies to replace (remember the #<policy\_name>#?)
    - Here we enter proxy references using the <proxyRef> tag
  + By default we should have this:



* + Notice “../Components” and “../common-policies”
    - This allows the deployment to look in these two specified folders for rules on what to replace inside the #’s based on file names.

Understanding how to use sign verify

* Upon any API call to Vantiv we use something called an “Authorization” header
* This header is accessed by using the apigee-access node module
  + The apigee-access.getVariable(req, “<variable name>”); command is how we access ours in javascript code.
  + At Vantiv most variables are accessed by this format for a variable name
    - “vantiv.license.base64” in this case for the license id in a 64 bit string
  + When we receive a call we get the license as a string
    - What we do this this string is then rotate it to base 64
    - We attach it in a soap base 64 string BinarySecurityString header
    - And then send this soap XML package with the header to Web-Methods endpoint
    - Web-Methods then sends this to data-power to make sure the license is valid (we do this a lot)
    - If it is we process the request
      * If it's not then we deny the call and send back a proper error response

Deploying your first proxy

* After writing your proxy in Webstorm and testing it locally we want to deploy it to Apigee.
  + This is assuming we have written a node server to deploy
* First we will open a command prompt and navigate to the folder our proxy is contained in.
* Once there we type:
  + “.\apigeeDeploy.bat nonprod-dev <Apigee login username> <Apigee login Password>
* And the batch process beings!
  + From here maven begins to deploy your proxy automatically to Apigee.