## Faults:

* Standard format



* How is this implemented?
  + <Proxy\_Folder>
    - apiproxy
      * proxies
        + policies -> default.xml?

# Notes:

Don’t use whats in apigee (fault wise).

Should follow this standard (provided by Scott Blasi – Solution architecture sharepoint).

Occurs whenever something “Bad” happens ©Hunsinger 2016   
any JS thrown errors should be handled here.

(apigees fault handeling documentation.

Fault rule section

## Policies:

* Standard format
* Really whatever the format needs to be
  + Tag on everything called a display name and that can be different than the name of the file
    - This isn’t cared about so much as the name of the file so REMOVE it
    - Naming schemas should be consist StrongCamelCase
      * VASFraud
      * ELE
      * FileNamingSchema





* What can they be used for?
  + These are limited to what they provide
* Best Practices:
  + Try not to use javascript
    - Slower
  + Use what is provided (assign variable so on)
* What shouldn’t we use them for?
  + Complement of the previous question

How do we implement the verify API license key (Yes one more time)?

* Starts with extract license key alias
  + Grabs the license token
  + Extracts the signing key
  + Uses key for next policy (Get license public key)
    - This key needs to exist or fire ensues
  + Executes java sign verify license (Java callout)
    - Basic validation (fields are correct, gets encryption and hashing then magic happens)
    - Takes public key and algorithms it to decode the hash (last $$#$$ ignored and killed)
    - Decodes the hased portion makes sure it matches initial and yay
    - Assigns it to header and basic 64 encodes it.
    - Then the API key gets verified (baked in but not part of the lincense)
      * Via flow fragments not policies

# Notes:

## Logging:

* Standard format



* AssignVariable
  + SubSystem
    - **Name:** Vantiv.logging.subSystemId
    - **Value:** VAS
  + ComponentID
    - **Name:** Vantiv.logging.subComponentId
    - **Value:** VAS<ComponentID> (ex. VASFraudServices for the VAS-Fraud endpoint)

# Notes:

## NodeJS:

* Standard formatting

# Server.js



# controller.js



* + Is this the most up to date way of doing this?

# Swagger 2.0



* + This the standard format of VDP using swagger?
  + Should all API’s that VAS Writes follow this model?



* Maybe recheck?

# Notes: