MACH Development Plan
20110916
v1

1. Registration
    1. Via registration message from the bus
    2. Via manual user entry
        1. filling out a mach register message and perhaps reading it in via the mach admin tool?
        2. Manually filling in db?
    3. Basic, based on list of component names, and topics read in on start up
2. appMgr integration
3. DB4O as datasource
4. Alerts
    1. Timing/parameters
    2. Adding new alerts
        1. Register/Unregister
        2. Startup/Shutdown of component
        3. ?
5. Component drop-ins
    1. Camel Processor/Bean
    2. Spring configuration for routes
6. Models and Diagrams

1. Registration
Finalize an initial method, and can leave room for other methods of registration as development continues and more components are mach-enabled. The most simple is a list mach can read in at start time, with the name of a component, and the topic they listen to. This won't necessarily allow for appMgr interaction, but at the least will allow for monitoring of their rabbitmq connection, and sending alerts. The second method is filling in a registration message, and having mach read in the file. It will be processed by the same code that would get it from the bus.

DUE: 23 Sep

2. appMgr Integration
Implement appMgr integration, a class that interacts with our appMgr shell scripts. Whether we want this for this milestone is up in the air. There could be a mechanism where if the component truly stops receiving its feed for some amount of time, mach would restart it via appMgr. Further, the restarting of components could also be exposed via JMX, so we could restart components with the mach-admin tool, which would allow us to tell mach to stop/start via appMgr from the command line. But at that point, why not just run the appMgr scripts. Soon we'll want to decide soon what we want to include.

DUE: TBD

3. DB4O

Finish implementing the DB4O manager class, which will handle CRUD operations, mostly dealing with storing NodeComponent objects, which will be the back-end data source to the registry.

DUE: ~ 30 Sep

4. Alerts
Some kinks need worked out with the timing of sending alerts, and allowing for some overlap to stop false positives from occurring. This may be as simple as lengthening the threshold the system is currently using.

DUE: ~ 17 OCT (will be done and tested and tweaked from now through the 17th)

5. Component Drop-ins
Implement a component drop-in. Currently this is happening in-line in the route, which is hard-coded. But the plan is to move the components over to spring, so we can dynamically define routes in XML, which can include going into a MACH processor. This is depends on the components that are to be mach-enabled being outfitted with the new spring configuration method.

DUE: TBD

6. Models and Diagrams
Use MagicDraw to generate models, etc.

DUE: 23 Sep