Installation Steps for the ADSI Prototype

Prerequisites:

1. The Docker container engine has previously been installed in the target environment(s). See Appendix A for instructions on installing Docker on Ubuntu Linux.

2. A traffic management / load balancing solution is available that can handle traffic from a client to a single virtual host and route it to the back-end Docker containers as appropriate. Apache web server works well for this purpose and an example configuration is shown in Appendix B. A more complicated example can be found in Appendix C.

3. The target environment has access to the public internet, specifically to DockerHub to obtain the container images and to https://open.fda.gov for access to ADSI's back-end data.

Installation:

1. Pull the two required Docker images from the DockerHub repository. The "ADSI API" image contains a RESTful service listening for http requests on port 8880. The "ADSI UI" image contains static content (HTML, JavaScript, CSS, images, etc) served by Apache web server listening on port 80.

```
> docker pull wtsintegration/api-wtsadsi
> docker pull wtsintegration/apache-wtsadsi
```

2. Start the Docker instances mapping host ports to the exposed Docker ports (The example commands use 8101 and 8102 for the two images respectively).

```
> docker run -d -p 8101:80 --name apache_wts_adsi -i -t wtsintegration/apache-wtsadsi
> docker run -d -p 8102:8880 --name api_wts_adsi -i -t wtsintegration/api-wtsadsi
```

3. Configure a load-balancing / traffic management solution to handle inbound requests and route them appropriately. The ADSI JavaScript has been configured to assume the REST API is available on the same host with "/api" added to the referring URL.

For example, assuming you wish to configure a virtual host to listen on http://10.10.10.10 to serve this application first configure it to route traffic to port 8101 on the server (or servers or virtuals etc) hosting the "UI" Docker instance by default. Then configure it to route any requests for resources starting with "/api" to port 8102 on the server (or servers, or virtuals, etc) hosting the "API" Docker instance. All requests for this application are stateless, so no "sticky sessions" are required.

The application should now be available on the configured virtual.

Appendix A: Installing Docker on Ubuntu Linux

Execute the following command:
> wget -qO- https://get.docker.com/ | sh

Appendix B: Installing and configuring Apache on Ubuntu Linux to serve as the load balancer, assuming the Docker images are running on *the same server* as Apache.

1. Install Apache
> sudo apt-get update
> sudo apt-get install apache2

2. Enable the proxy module
> a2enmod proxy_http

3. Modify the configuration (/etc/apache2/apache2.conf) by adding the following lines:

ProxyPass /api http://localhost:8102/api
ProxyPass / http://localhost:8101/

4. Start (or restart) apache2
> service apache2 start

Appendix C: Installing and configuring Apache on Ubuntu Linux to serve as the load balancer, assuming the Docker images are running *on a different server* than Apache. In this example 192.168.1.17 hosts 2 of each version of the Docker images.

1. Install apache
> sudo apt-get update
> sudo apt-get install apache2

2. Enable the proxy module, the proxy_balancer module, and the lbmethod_byrequests module
> a2enmod proxy_http
> a2enmod proxy_balancer
> a2enmod lbmethod_byrequests

3. Modify the configuration (/etc/apache2/apache2.conf) by adding the following lines:

<Proxy balancer://adsiapi>
 BalancerMember http://192.168.1.17:8102
 BalancerMember http://192.168.1.17:8104
</Proxy>

<Proxy balancer://adsiapache>
 BalancerMember http://192.168.1.17:8101
 BalancerMember http://192.168.1.17:8103
</Proxy>

ProxyPass /api balancer://adsiapi/api
ProxyPass / balancer://adsiapache/

4. Start (or restart) apache2
> service apache2 start