Open API Documentation

Including

Methods and Role Definitions

Version 1.2

Drafted by: Matt Meng

3/28/2014

Open API Project Summary

The object of Open API project is to provide the capability for the third party venders to access Open HMIS Data. The “Client Intake” module in our compass system can be the prototype of the Open API project.

The Architecture of Open API

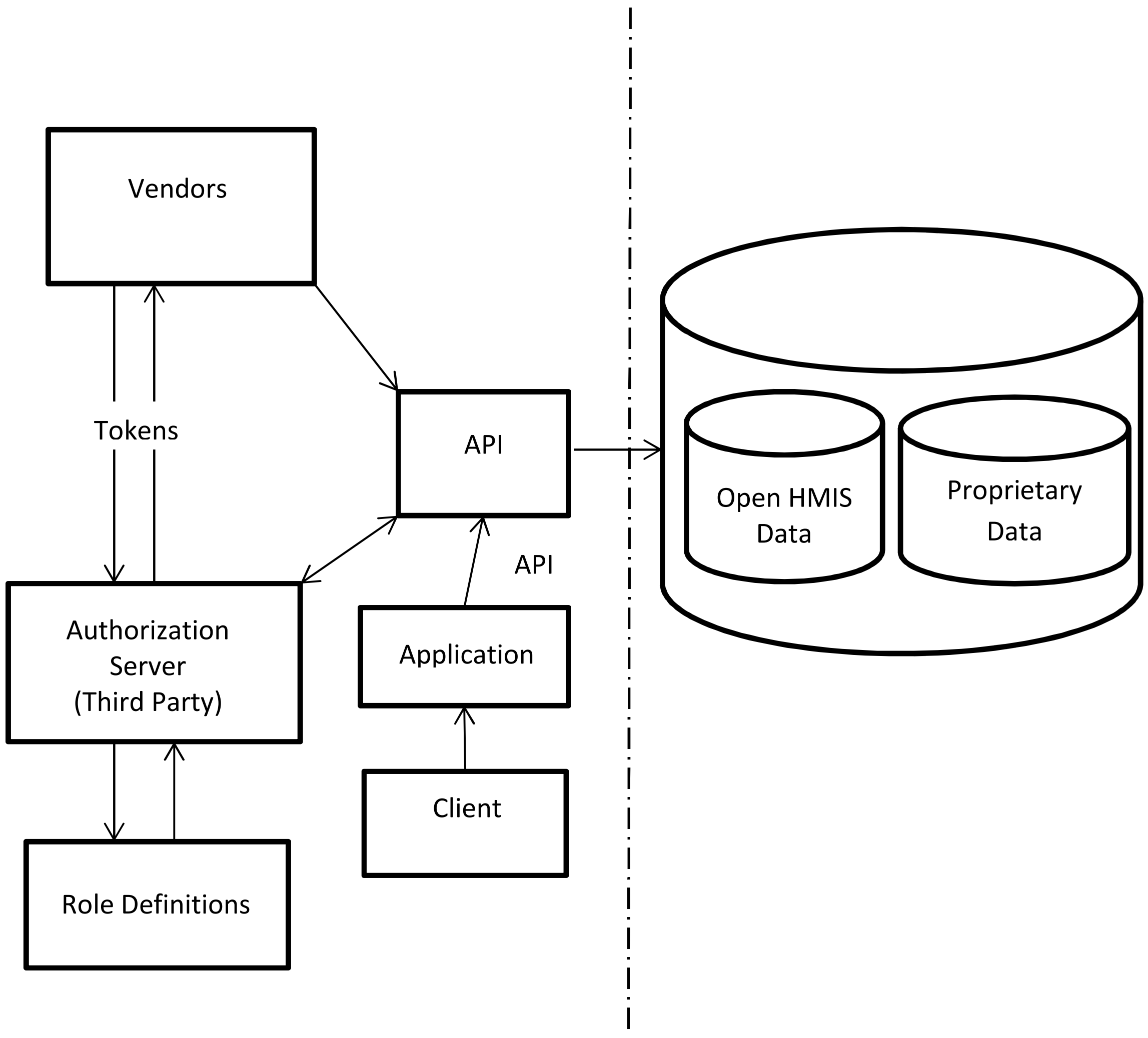

Figure 1. The Architecture of Open API project

As Figure 1 above shown, the venders will need to connect to OAuth server to validate their tokens in order to access API. If it is validated the application will be granted to access to the database through API. Venders will need to hold their extra data in their own schema. The API will be published with open source license.

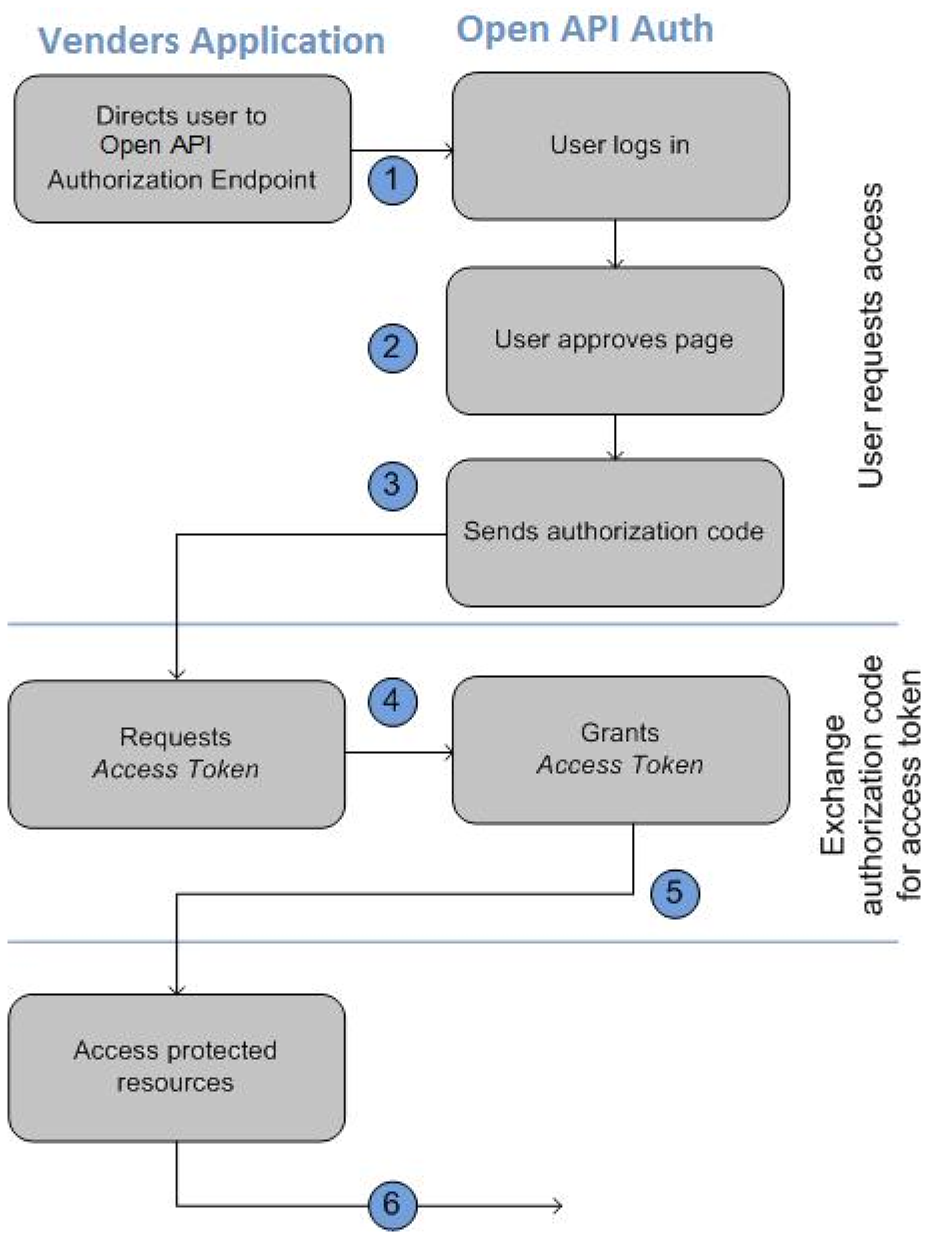

Figure 2. Web Server OAuth Authentication Flow

Major Technologies

Major technologies needed for the Open API project are proposed by system team including Java (1.6 JDK), Google APP engine, POJO, Hibernate, Struts, Log4J and OAuth(2.0).

1. OAuth(2.0): It is an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications. It is a simple way to publish and interact with protected data. The OAuth will be set up on a server for the third party vendors to validate their tokens.

   a. The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service.
   b. Third party role definitions would be needed
2. Google App Engine:
   a. Google App Engine is a Platform as a Service (PaaS) allowing users to build and run the applications on Google's infrastructure.
3. POJO: The Plain Old Java Object.
4. Struts:
   a. With Struts, an open-source MVC framework, developers can create elegant, modern java applications. It favors convention over configuration and ships with plugins to support REST, AJAX and JSON.
5. Hibernate:
   a. Hibernate enables developers to develop persistent classes for applications interacting with relational databases. The selling point of Hibernate is the notion of database portability.
6. Log4J: With log4j developers are able to debug the application without modifying the application binary at runtime.

Data Elements for the 1st Stage

1. Name (First, Middle, Last, Suffix)
2. SSN (2 fields; numeric field and drop down definition)
3. DOB (2 fields; numeric field and drop down definition)
4. Race
5. Ethnicity
6. Gender
7. Veteran Status
8. Disabling Condition

Methods

1. **Client Form(Client Bean)**

   Getter() and setter() for following field:

   1) Name
   2) SSN
   3) DOB
   4) Race

5) EthnicityKey
6) GenderKey
7) Veteran
8) DisablingCondition

2. Session Manager (User role definition)

In our compass system, we have the Http Session and Servlet to manage the role. For OpenAPI we don’t need to worry about the presentation layer. It is not necessary to have an Http Session or Servlet Request for each request. We may use Oauth 2.0 session to manage the role.

If the salesforce is the first vender we will support, the following online support document for salesforce with Oauth 2 reference link might be useful:

http://www.salesforce.com/us/developer/docs/api_rest/Content/quickstart_oauth.htm

Roles:

1) StatewideUser
2) CocUser
3) RegionAdmin
4) AgencyAdmin

Attributes:

1) Agency_Key
2) Agency_Name
3) Region_State_Key
4) Region_State_Name
5) UserKey
6) UserID
7) HouseholdKey
8) ClientKey
9) RegionKey
10) IntakeKey

Methods:

1) Private static SharedSessionManager ssm=new SharedSessionManager();
2) Private HashMap sharedSessions=new HashMap();

3) Public SharedSessionManager getSharedSession(String sessionId){}

4) Public Object getAttribute(String sessionId, String name) {};

5) Public void setAttribute(String sessionId, String name, Object value) {};

6) Public void removeAttribute(String sessionId, String name) {};

7) Public void removeAll() {};

8) Public static boolean isSessionValid(SharedSession sess) {};

9) Public Boolean isSessionLoggedIn (String sessionId) {

   //check if session is null or valid, if not return false

   //else return sess.getAttribute(UserKey) !=null;

   };

10) Public synchronized void invalidateUser(SharedSession currSess, String userKey){

   // if the user is not valid, remove the key

   };

11) Public static void updateSharedSession(){

   //new client, new household case or the household/client has changed

   };

12) Public static void setUserKey(Request request, String userKey){

   //set UserKey with the session request

   };

13) Public static void setAgencyKey(Request request, String agencyKey){};

14) Public static void setRegionKey(Request request, String regionKey){};

15) Public static void setStateKey(Request request, String stateKey){};

16) Public static void setClientKey(Request request, String clientKey){};

17) Public static void setHouseholdKey(Request request, String hhkey){};

18) Public static String getClientKey(Request request) {};

19) Public static String getHouseHoldKey(Request request) {};

20) Public static String getAgencyKey(Request){};

21) Public static String getUserKey(Request request){};

22) Public static String getRegionKey(Request request) {};

23) Public static void setRegionAminFlag(Session sess, Boolean regionAdmin){};

24) Public static Boolean isRegionAdmin(Session sess){};

25) Public static void setMainstreamAdminLevel(Session sess, int msLevel){

//set up the admin level

};

26) Public static int getMainstreamAdminLevel(Session sess) {};
27) Public static void setAgencyAdminFlag(Session sess, Boolean agencyAdmin){};
28) Public static Boolean isAgencyAdmin(Session sess) {};
29) Public static void setStatewideMemberFlag(Session sess, Boolean statewide){};
30) Public static boolean isStatewideMember(Session sess) {};
31) Public static boolean isCocUser(Session sess) {};
32) Public static void setCocUserFlag(Session sess, boolean cocUser) {};

3. ClientInfoQuery

1) private String getRelationshipQuery(String householdKey, String clientKey){};
2) private String getClientQuery(String clientKey, String agencyKey){};
3) public String getRelationshipUpdateSQL(Form form, String householdKey, String clientKey){};
4) private String getClientInsertSQL() {};
5) private List getClientInsertParams(Form form, String newClientKey, String regionKey, String userKey)
6) public String getClientUpdateSQL(){};
7) private List getClientUpdateParams(Form form, String clientKey, String userKey){};