UnderstandingAppCard

App-Card.

App-Card data structure.

App-Card is a kind of M-Card app-cards are defined on the Higgins wiki here. The data "pointed to" by an app-card is described on Higgins wiki here.

[PAUL: for the Mydex project we need something added to the App-Cards described above. I have started today to sketch this out. See the optional template attribute that has just been added to an AppCard instance. The next thing I need to do is document the format of this template file. I plan to define it as a combination of an AppData context + MappingContext + some access control metadata (to indicate which attributes can be edited by the user) + the type of context (e.g. Jena, LDAP, etc.).

Why? because when the user imports a Mydex app-card the active client needs to dynamically create a context for the app-card to use (the contexts don't already exist when the card is imported).]

It will be distributed by using crd format. The App-Card specific data will be exposed by using extension. The following I-Card properties will be re-used:

- Card Name
- Card Type
- Card Image
- Issuer
- Policy
- SupportedEmail
- SupportedSite
- Expired
- Version

The following App-Card specific properties will be exposed with App-Card extension:

Required:

- Application Id (please, see http://wiki.azigo.net/display/Azigo3/ABX+2.0)
- Developer Id (please, see http://wiki.azigo.net/display/Azigo3/ABX+2.0)
- Card description (please, see http://wiki.azigo.net/display/mydex/Mydex+Verified+Address+App)
- Sites - some cards can be design to work only on specific web sites. we can re-use chrome extension match pattern by adding "dashboard" schema http://code.google.com/chrome/extensions/match_patterns.html
- Permission list - Azigo platform provides few operation types like "get/set app card data", "request data from other app card", "share data to the other app card", "request access token", "request suggestions" etc, so we have to define supported permission list. The App-Card developer has to include permissions which used by his App-Card. The Azigo will show accept dialog to user before importing App-Card (please, see http://wiki.azigo.net/display/mydex/Mydex+Verified+Address+App)

- [PAUL: I agree that the Azigo PDS Client supports get/setExAttributes() (aka "get/set app card data") and "request suggestions". I don't see the need to request and share data between app cards OTHER THAN as defined by the "p:source" persona-linking attribute. I DO agree that we need a permissions list (on a per-attribute basis). I'm going to define this in the app-card "template" file that I mention in my comment above].

Optional:

- Dependency - the list of dependency App-Card. if App-Card depends on other App-Card, it should define depended App-Card "Card Name", "Application Id", "Developer Id", "Min Version" and "Max Version", "Download URL" (from Azigo market place) or "Download site" (third party site). Azigo has to check dependency list before importing App-Card and propose to download install depended App-Cards. bq.Should Azigo install/enable App-Card if dependencies are not satisfied ?
  - [PAUL: the JS for an app-card should just have to call getExAttributes()--the PAUL component does all of the matching, etc. and figures out the source context. I don't see the need for an explicity dependency list. getExAttributes allows you to optionally specify the card/context issuer that you trust as the source your attributes. I think that is all we need]
- AppUI JS - this JS will be running in the dashboard "Profile" tab. (please, see http://wiki.azigo.net/display/mydex/Mydex+Verified+Address+App). This JS can load external web page, but it also should support offline mode.
- WebPage JS - this JS will be injected in the browser web page if web page matches "Sites" pattern.
- OnChangeCardStateJS - this JS may define JS functions (onInstalled, onUninstalled, onUpdated, onEnabled, onDisabled) which will be run by Azigo after installing/uninstalling/updating/enabling/disabling App-Card. This JS will be injected and executed in the "Profile" tab, so it can re-use "AppUI JS".
- Public JS - this JS will be available to the other App-Cards which required some data from current App-Card. For example, Brent Council App needs Mydex verified address.

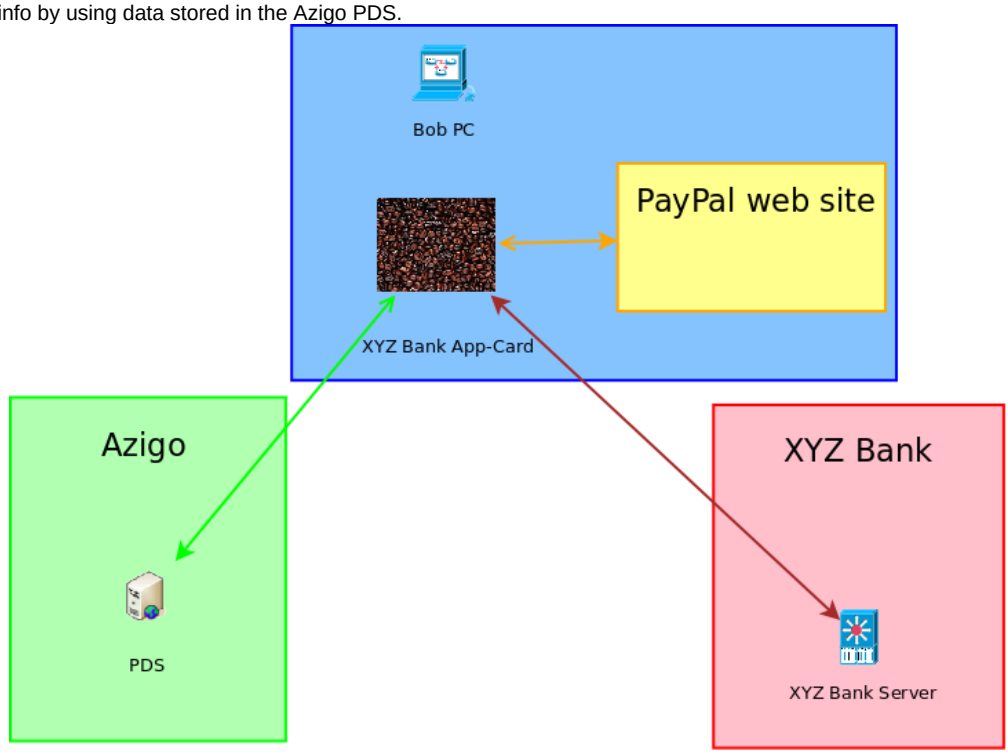
> *Let App-Card directly read data stored by other App-Card is not so good. it can not share just few stored attrs or if attribute will be renamed, it cause unexpected error, furthermore it will be more hard to implement security model. It will be more flexible to provide JS functions. Are you agree?*

- Public JS API - it can be help or example how to use provided "Public JS"
- Black list site - the list of site that user disable to this card. if user import/export App-Card Azigo may store that information. bq. I'm not sure that it should be exported in the crd file.

General App-Card Data Flow.

Let assume that Azigo is well know platform and "XYZ Bank" is developing App-Card. The "XYZ App-Card" should work on PayPal site (auto-filling billing info).

1. After installing XYZ App-Card Azigo invokes JS onInstalled() functions, the XYZ App-Card ask login/password for XYZ Bank user account, loading from XYZ Bank server billing info.
2. XYZ App-Card Azigo stores billing info in the Azigo PDS (card information was encrypted on the client side).
3. Than, user open browser and buy new bicycle on bicyclebuy.com. The bicyclebuy.com redirects to paypal site and "XYZ App-Card" fills billing info by using data stored in the Azigo PDS.



Keynotes:

- XYZ Bank is third party company that developed App-Card without Azigo by using well know technology (JS) and already existed bank service.
- The data was encrypted and stored by Azigo3 client app, so Azigo can't read user data.
- The "XYZ App-Card" "WebPage JS" read the data from Azigo PDS, it was more faster than read the same data from bank server and it didn't require ask login/password.
- The "XYZ App-Card" "AppUI JS " load web page from XYZ Bank server which refresh data stored in the Azigo PDS.

If we're going to use the same Data Flow with MyDex and Brent Council App-Cards, we need two web applications. MyDex Web App wrap communications with Expirian(or any other) WS. Brent Council Web App wrap communications with Brent Council server. bq. Are you agree?

Form-Filler App-Card.

Mydex App-Card require Form-Filler feature.

> *Form fields should be auto-filled from the user's root (formerly called meta) context, according to the same algorithm as described in Azigo3 Form Filler. The general idea is that: http://wiki.azigo.net/display/mydex/Mydex+Verified+Address+App*

But Mydex and Form-Filler should be different App-Cards. Form-Filler requires more permissions (getSuggestions), so we need Form-Filler App-Card which will work in dashboard.

ABX API

Requirements:

- security - App-Card identifies itself by using "Developer Id" and "Application Id" (so ABX should block ability to change those values), ABX has to execute "App-Card WebPage JS" in the "App-Card sandbox", which control access to ABX API, Cross App-Card messages API, ... by using the "App-Card Permission list".
- easy to use.

ABX Public API

> *We was going to commit Azigo3 to Higgins. Can I use ABX (Azigo Browser Extension) names in the API?*

JS function ABX_getABX()

Create new ABX instance. The "App-Card WebPage JS" must create ABX instance for using ABX methods.

JS Example:

```
var ABX = ABX_getABX();
```

Few App-Card WebPage JS's would be injected in the same web page, so please use some prefix for global variables, for example com_azigo_FormFiller_ABX.

ABX method isExtendedSelector()

Returns true if and only if Azigo is the current selector.

Note: **Ex**() methods (see below) are supported only by extended" selectors (Azigo).

JS Example:

```
var ext = ABX.isExtendedSelector();
```

ABX method getExAttributes(string rp, string audience, Attribute attributes, *optional* Where where, function responseCallback)

Parameters:

- **rp**: string identifier of the "next hop" attribute data sink. It is expressed in as detailed a form as possible.

        - If the ultimate attribute data consumer is a website then the string is at least a domain, possibly with an additional path:
            - May be the domain of an RP (e.g. "whitehouse.gov" --we're trying to login to the RP using IMI. In this case the "call" would have originated in HBX.
            - May be the domain+path (e.g. "staples.com/checkout") --the page of a website containing a form to be filled. In this case the "call" would have originated in an app (e.g. Form Filler) and then to ABX2 and finally to here. the "/checkout" portion adds a bit more framing (trying not to say context!) to the attributes.
    - **audience**: string. Must match either the agent or the rp parameter value or be nil (undefined). If not nil, then indicates whether to encrypt tokens for the agent or the rp. [wiki:Not implemented]
    - **attributes**: is an array of **Attribute** (please, see ABX Types)
    - **where**: is an array of **Where** (please, see ABX Types)
    - **tokenTypes**: is an array of token types that are understood by RP. Must be one of :
        - SAML 1.0 URI (from IMI spec)
        - SAML 1.1 URI (from IMI spec)
        - XDI document
    - **responseCallback**: function to process response like

```
function(Attribute attributes)) {...});
```

    .

JS Example:

```
var attrs = [
             new ABX.Attribute("http://higgins.eclipe.org#email_address","",true,""),
             new ABX.Attribute("http://higgins.eclipe.org#password","",true,"")
             ];

var where = new ABX.Where("http://higgins.eclipe.org#email_address","myn");


ABX.getExAttributes("http://higgins.eclipe.org", "", attrs, where,"", function (attributes){
 if (attributes && attributes.length > 0){
  //successfully
  for ( var i = 0; i < attributes.length; i++) {
   // attributes[i].attribute and attributes[i].value
  }
 }else{
  //process error
 }
});
```

ABX method setExAttributes(string rp, Attribute attributes, *optional* function responseCallback)

Parameters:

- **rp**: string identifier of the "next hop" attribute data sink. It is expressed in as detailed a form as possible.
    - If the ultimate attribute data consumer is a website then the string is at least a domain, possibly with an additional path:
        - May be the domain of an RP (e.g. "whitehouse.gov" --we're trying to login to the RP using IMI. In this case the "call" would have originated in HBX.
        - May be the domain+path (e.g. "staples.com/checkout") --the page of a website containing a form to be filled. In this case the "call" would have originated in an app (e.g. Form Filler) and then to ABX2 and finally to here. the "/checkout" portion adds a bit more framing (trying not to say context!) to the attributes.
- **attributes**: array of **Attribute** (please, see ABX Types)
- **responseCallback**: function like

```
function(int status)) {...});
```

    . if success the **status** equals 0.

JS Example:

```
var attrs = [
             new ABX.Attribute("http://higgins.eclipe.org#email_address","myname@example.com",true
,""),
             new ABX.Attribute("http://higgins.eclipe.org#password","my_password",true,"")
             ];

ABX.setExAttributes("http://higgins.eclipe.org", attrs, function (status){
 if (status && status===0){
  //successfully
 }else{
  //process error
 }
});
```

ABX method delExAttributeValue(Attribute attribute, *optional* function responseCallback)

Parameters:

- **attribute**: **Attribute** (please, see ABX Types), **attribute.value** is required;
- **responseCallback**: function like

```
function(int status)) {...});
```

. if success the **status** equals 0.

JS Example:

```
var attr = new ABX.Attribute("http://higgins.eclipe.org#email_address","myname@example.com",true,"");

ABX.delExAttributeValue(attr, function (status){
 if (status && status===0){
  //successfully
 }else{
  //process error
 }
});
```

ABX method delExAttribute(string attribute, *optional* function responseCallback)

Parameters:

- **attribute**: the URI of the attribute
- **responseCallback**: function like

```
function(int status)) {...});
```

. if success the **status** equals 0.

JS Example:

```
var attr = new ABX.Attribute("http://higgins.eclipe.org#email_address","",true,"");

ABX.delExAttribute(attr.attribute, function (status){
 if (status && status===0){
  //successfully
 }else{
  //process error
 }
});
```

ABX Types

Attribute

- **attribute** is a URI indicating the attribute type
- **value** is attribute value
- **optional** is a boolean (if true then this attribute is desired but not required)
- **authorities** is a list of domains that are considered by the caller as authoritative WRT this attribute and thus must be used as the source of the attribute, if this list is nil then self asserted values are acceptable. If authority == dev (where dev is the developer of an action-bearing r-card) then only the "host" card of that app will be allowed as the source of attributes.

JS Example:

```
var attr = new ABX.Attribute("http://higgins.eclipe.org#email_address","",true,"");
```

Where

- **attribute**: is the attribute URI
- **expression** (for now) May be an exact value (e.g. "Alice") or may be a regex that the value must match. For 3.0M2: the so-called "regex" is just a string of one or more characters followed by
  an asterisk (e.g. 2*, or 25*)--meaning that the value of the attribute must start with these characters (e.g. values "290" and "250" will match "2*" and "250" (only) will match "25*")

JS Example:

```
var where = new ABX.Where("http://higgins.eclipe.org#email_address","mye");
```

App

- **devId**: is the developer id
- **appId**: is the application id

JS Example:

```
var app = new ABX.App("com.azigo","FormFiller");
```

Cross App-Card messages

ABX method onRequest.addListener(function(any request, App sender, function sendResponse))

If App-Card developer would like to share some data to other App-Cards, the "App-Card Public JS" has to register event listener to handle the Cross App-Card messages.

Parameters:

- **request** is any JSON object or null('undefined') defined by App-Card developer,
- **sender** is App (please, see ABX Types),
- **sendResponse** is function that should be called to send response data.

JS Example:

```
```

ABX method sendRequest(App app, any request, function responseCallback)

If App-Card needs some data from other App-Card, it can use sendRequest method.

Parameters:

- **App** is App that share data (please, see ABX Types),
- **request** is any JSON object or null('undefined') defined by App-Card developer that share data,
- **responseCallback** is a function like

```
function(any response) {...});
```

.

JS Example:

```
```

ABX Private(Azigo only) API

ABX method getSuggestions(string attribute, function responseCallback)

Note: this method may only be called by UN-PW-Filler and Form-Filler, no others. (For privacy reasons).

Parameters:

- **attribute**: the attribute type string (HTML form element id)
- **responseCallback**: represents function like

```
function(any response) {...});
```

.

JS Example :

```
var onSuggestions = function (res){
//...
};
//...
ABX.getSuggestions(attribute, onSuggestions);
```