Overview of the APIs and Data Structures and the Entity Registry Working Groups

TIER Vision and Overview

- Help education and research organizations solve the Identity and Access Management (IAM) challenges they encounter
    - By providing open source implementations of key IAM capabilities and assuring their long-term sustainability
    - By standardizing
        - How applications (whether local, federated or SaaS) *integrate* with IAM infrastructure
        - How existing institutional IAM infrastructure can *interoperate* with TIER components to provide a full IAM service suite

TIER Entity Registry and Data Structures and APIs Working Groups

- The TIER Entity Registry Working Group and the TIER Data Structures and APIs Working Group share the following key goals
    - To define integration and interoperability strategies and models
    - To help charter development projects that address specific gaps in existing open source IAM packages
    - To develop a comprehensive functional model of IAM
    - To define and adopt specifications for the resource schema and interfaces needed to deliver identity and access management (IAM) services
        - Between the various TIER IAM components
        - Between TIER components and the rest of the institutional IT landscape, both on premise and in the cloud
    - Provide guidance on building IAM infrastructure and processes that accord with the TIER model

Standards, Tools and Guidelines set out in TIER Release 1

- Expose IAM capabilities at RESTful endpoints
    - ...Where it makes sense: LDAP, SAML, etc. still have their well-earned place, TIER will take full advantage of such common protocols and interfaces. OAuth 2, OpenID Connect and UMA are also coming into play.
    - *REST*-ness in the TIER context means: HTTP verbs operate on Resources (groups, users,....); RPCish idioms should only be used when nothing else will do what needs to be done.
    - The model for interoperating with existing institutional IAM services is to provide the TIER components with connectors that know how to interact with both back end legacy systems as well as the growing number of contracted-out SaaS and PaaS services
    - An API-first design helps us achieve and maintain a level of abstraction from specific implementation choices. This gives TIER adopter sites the option to wrap their favorite legacy IAM service in a TIER API knowing that it will integrate well with other TIER or TIER-compliant packages.
- Adopt the many useful conventions specified in the new IETF standard, *SCIM 2.0* ,
    - around the design choices that would otherwise tend to provoke endless working group debates on matters such as pagination, metadata schema, data formats, etc.
    - the choice to leverage SCIM, as much as anything else, made the decision to support *JSON* easier. Support for XML can be provided if and where it's needed.

API Specifications:

- The canonical specification language for HTTP-oriented APIs in TIER is *Swagger 2.0*
- Why Swagger and not *RAML* or *API Blueprint*? (see this recent comparison on dzone)
    - In the move from version 1 to version 2, Swagger incorporated a lot of RAML's best features (around reusable definitions, etc.)
    - Swagger 2 has been adopted as the basis for further development by the industry-launched *Open API Initiative* (http://openapis.org, more on github here) and that should strengthen the already thriving Swagger developer and adopter community