TIER IAM BackBone Scenario

Simple Backbone Usage Scenario for which TIER will Demo Solutions

Narrative

The Backbone Usage Scenario is intended to illustrate the simplest possible end-to-end example of TIER in action. The story begins with a new hire or a new student arriving at a hypothetical university, having their basic ERP-based demographic and affiliation information incorporated into the core entity registry, passing through an identity match process to determine if the person is already represented in the registry.

Once in the registry, the new arrival should be provisioned with an account and credentials in a general purpose LDAP directory. Further automated processing of the ERP data ipopulates basic affiliation groups in Grouper (in this demo, the affiliations will be faculty or student). Finally, there is a Shibboleth-protected "Learning Management System" accessed by the faculty member or student through SAML-based Web-SSO. The LMS will display either a course design page (for faculty) or a course catalog (for students).

The steps in the scenario can be traced from the numbered items in the third column of the table below.

Identity Ecosystem Column	TIER Functional Service	Functionality in BackBone Scenario
Systems of Record	Registration and Enrollment from SoR (BennO, KeithH)	1.1) Provides periodic feed (most SoRs can do this today, daily, hourly, etc)
Filtering/Routing/Integration Rule Engine	Rule Engine Service (SteveC) Registration and Enrollment from SoR>Search /Match (KeithH) Credential Management (WarrenC) Rule-Driven Group Management (SteveC, ChrisH)	2.1) For each SoR person, use the IdMatch API to see if this person already exists in the Registry 2.2) For ADD operations 2.2.1) If IdMatch found, update the existing Registry entry (un-soft-delete if necessary) 2.2.2) If no match found, create a new Registry entry, and send “new user” event in the Lifecycle Management Engine (lizard brain or ?) which triggers replication to LDAP and Credential Store 2.2.2.1) LDAP and Kerberos for reference implementation? *The Shib plug-in for Web SSO authentication is the integration point for alternate AuthN methods* 2.2.3) Assignment of a one-time token good for a userid/password credential (or challenge questions? (both approaches have been discussed) 2.3.4) Notify user to visit “Activate” page. 2.2.5) Create group memberships based on affiliation attributes in the input feed. Drive this from a rule in the rudimentary provisioning engine, aka "lizard brain". ○ Translate the syntax/semantics of the local feed to standard TIER format ○ Invoke TIER utility to process that file ○ A Grouper loader (cron) would query to keep track of entry changes or track attribute changes ○ Brown feeding registry changes directly to Grouper ○ UW: Changes are fed to LDAP; Grouper as backend; many many sources of subjects for reference groups ○ BillT: Use loader or keep up to date with RT updater; Groups 2.2.5.1) If AFFILIATION from SoR = faculty then add person to Group “Instructor” 2.2.5.2) If AFFILIATION from SoR = student then add person to Group “Learner” 2.2.5.3) Grouper groups update LDAP groups; isMemberOf in Shib SAML assertions comes from LDAP 2.3) For DROP operations, TAG the user appropriately in the Registry (“soft delete”); this would cause updates in LDAP and Kerberos (expire account), and remove them from groups.

Repositories	Repository Components (BennO EthanD, KeithH)	3.1) Provide IdMatch API 3.2) Provide Add Person API 3.3) Provide DROP Person API 3.4) Provide add person X to group Y API 3.5) Provide remove person X from group Y API
Provisioning /Rules Engine	Provision/De-provisioning Person Entity (BillT with help from JonM) Rules Engine (SteveC)	4.1) When person is added replicate them to ldap 4.2) When person is added replicate them to kerberos 4.3) When person is dropped flaf as soft-deleted or remove them(?) from ldap 4.4) When person is dropped remove/disable them from kerberos 4.5) When person is added to group update linked ldap group 4.6) When person is dropped from a group update the linked ldap group
Capabilities of the backbone and associated application. Demo in this manner…		5.1) Configure the sample Service Provider application to use Shibboleth as its Web SSO mechanism 5.2) Configure requested attributes element on Shib session protected endpoints, ask for displayName and isMemberOf 5.3) Have registered users browse to the protected application 5.4) Users authenticate at their IDP 5.5) Depend on LDAP for the initial user authentication behind the IDP, Release attributes X, Y, Z to the SP application 5.6.1) If authenticated user isMemberOf the Instructor Group, show a hello {name} “Course Design” page 5.6.2) If authenticated user isMemberOf the Learner Group, show a hello {name} “Course Catalog” page

What minimal Lego-Block collection of functional interfaces are needed to actually build a proof of concept implementation?

- Minimal IDmatch API implementation for first round demonstration sandbox
- Evaluation of rules engine longer term will include a review of http://activiti.org and its business process (including workflow) and rules capabilities
- Suggestion to review more use cases, considering the human beings' role in the processes, as a guide to requirements on the UI
- Leverage a tool like PWM https://github.com/pwm-project/pwm for account and credential management. Evaluate its utility as a way to do all the messy work around account and credential management. It can do tokens, password recover questions, reset to personal email, SMS message to mobile, and more. Based on initial look at the documentation this looks like a good "lego" for the TIER package

Implementation Guidelines

1. COmanage should be the basis for one implementation in the demonstration sandbox.
2. A second implementation using midPoint for the Entity Registry and Provisioning functions is under assembly at https://midpoint.testbed.tier.internet2.edu.
3. One challenge: How do we successfully complete all steps of this story without implementing a full-fledged rules engine?
    a. The now famous ‘lizard brain’ (simple) rules engine is one answer. It would need to be barely smart enough to do the simple logic behind the (SoR feed)-to-(Registry user attribute)-to-(Grouper Group), and behind the process that adding to ‘faculty’ group should trigger provisioning to LDAP and Kerberos.
    b. midPoint provides this kind of rule-based provisioning functionality
4. The Entity Registry may simply inform a provisioning engine: “This Resource changed and here’s its current representation”. This can be accomplished via either a push over RESTful APIs or as an event message payload on a queue or pub-sub channel or both.
5. Beyond simple add/remove faculty and students; one individual tends to have multiple hats, each hat implies access to a particular set of services