Integrating Applications into UCAR Authentication and People DB Services Utilizing REST APIs

Bruce Sun
Web Engineering Group
Topics for Today

• Brief introduction to RESTful Web Services
• Auth REST APIs
• People DB
• People REST APIs
What is REST API?

- REST API a.k.a RESTful Web Services
- In REST API
  - URLs are used to identify the object for the operation
  - Standard HTTP methods such as GET, PUT, DELETE are used to represent the operations
  - Params in URLs are used as parameters for the method on the server
  - Data are transferred directly on HTTP body
Auth REST API - Architecture

- Apps (C/C++, Java, #C/.net, Perl, Ruby, Python, php, etc)
- Auth REST API
- Kerberos Auth Server
- UCAS Auth Server
- Radius Server
- ....
Auth REST API - Benefits

• REST API essentially acts as a proxy to the auth servers
• It hides the complexity to access the auth servers from the application
• When auth servers change, none or minimum changes will be needed by the applications
  – Example, When UCAS auth server is replaced by Keberos, no change was needed by the apps.
How to Use Auth REST API

- **URL**
  https://auth.api.ucar.edu/authenticator/rrh/authentication

- **Credentials**
  Login and Password

- **HTTP Method**
  HTTP Put

- **HTTP Body**
  ```
  {
    "authType":"password",
    "username":"bsun",
    "password":"bsun"
  }
  ```
How to Use REST API - Continued

- **HTTP Response Code**
  
  200 - Ok response
  
  401 – invalid credentials
  
  403 – invalid URL path
  
  500 – Server Error. Error message will be contained in HTTP Body

- **HTTP Response body**

  ```json
  {
    "username":<User Login>,
    "valid":"true" or "false"
    "firstName":<User First Name>,
    "lastName":<User Last Name>
  }
  ```
def authenticate(url, authType, username, password):

    http = httplib2.Http()
    authentication = {'authType': authType, 'username': username, 'password': password}
    http.add_credentials('admin', 'admin')
    headers = {'Content-type': 'application/json'}

    response, content = http.request(url, 'PUT', headers=headers, body=demjson.encode(authentication))

    status = response.status
    if status == 200:
        authentication = demjson.decode(content)
        return authentication['valid']
    if status == 401:
        print 'Invalid credentials for Auth Web Service Server'
    if status == 500:
        print 'Server error'
    print content
Auth REST API Example - Java

• Write your own client
• Use WEG provided client

```
try {
    Authentication auth=AuthClient.getAuthentication
        ("token", <username>, <yubikey or crypal card token>);

    if (auth.isValid()) {
        // do things for successful authentication
    } else
        // do things for failed authentication

} catch (AuthClientException ex) {
    // handle exception
}
```
Auth REST API – Integration with 3rd apps

• WEG provides libs to integrate with 3rd party apps such as Jira and Confluence.
Auth REST API – Docs & sample codes

• https://wiki.ucar.edu/display/weg/Auth+REST+API
Auth REST API

• Questions?
People DB 2.0

• Bring together UCAR master metadata about people/groups/orgs into a federated service
  – Staff, visitors, collaborators
  – UCAR organizations
  – External organizations
  – Groups
  – Alias, mailbox, mailman list
  – Locations (ISO countries, ISO states, Buildings)
• Honor who is the authoritative source for the data
• Role based permission model to control who can change what
• Use Synchronizer to push data changes to down stream data consumers at real time
People DB 2.0 - Architecture

- Admin Scripts
- HR
- NETS
- Teragrid Client
- People Search

REST API

App Server

PDB

Synchronizer

Plugin

MASTER LDAP

Plugin

HR

(SQL Server)

Plugin

NETS

(My SQL)

Plugin

ACC8

(Oracle)
4 ways to access PDB data

• Use master LDAP
  – read only

• Subscribe to Synchronizer to have data pushed to your own data store in real time
  – read only

• Direct access to People DB (MySQL) using SQL
  – read only, not recommended

• Use People REST APIs
  – read and write
People REST APIs - WADL

- WADL (Web Application Description Language)
  - like WSDL in SOAP
  - simply GET the resource /application.wadl at the base URI
  - https://tmerapi:8203/api/application.wadl
People REST APIs - public

- Search External Persons (Collaborators)
- Get External Person (Collaborator) Detail Data
- Search Internal persons (UCAR Staff and Visitors)
- Get Internal Person Detail 2.0
- Search Persons 2.0
- Get Person Data

- Search Organizations 2.0
- Get Organization Detail 2.0
- Get Organization Hierarchy 2.0
- Get Sub Organizations 2.0
People REST APIs - Restricted

- Search Groups
- Add or Update Group
- Delete Group
- Add or Update Group Admins
- Get Group Detail Data
- Check if a person or a group or an org has permission for a group
- Get Group Admins

- Add Or Update External Organization
- Get External Organization Data
- Delete External Organization

- Add Or Update Mailbox
- Get Mailbox Detail Data
- Delete Mailbox

- Add Or Update Mailman List
- Get Mailman List Detail Data
- Delete Mailman List

- Add or Update UCAR Organization Admins
- Check if a person is an admin for another person
- Check if a person is an admin for a UCAR organization

- Get Building Data
People REST APIs — Public

- `<base URL>/<collection name>?param1=value1&param2=value2`
  - HTTP Method: GET
  - Used for Search
  - Example: https://tmerapi:8203/api/persons?name=mark

- `<base URL>/<collection name>/<id>`
  - HTTP Method: GET
  - Used for Getting detail data for an object
  - Example: https://tmerapi:8203/api/persons/bsun

- Docs & sample codes
  - https://wiki.ucar.edu/display/weg/People+REST+API+2.0+core+services
People REST APIs – Restricted

- `<base URL>/protected/admin/<collection name>?param1=value1&param2=value2`
  - HTTP Method: GET
  - Used for Search
  - Example: `https://tmerapi:8203/api/protected/admin/groups?name=cisl`

- `<base URL>/protected/admin/<collection name>/<id>`
  - HTTP Method: GET, PUT, DELETE
  - Used for Retrieving Data for an object, Add/Update an object, and Delete an object
  - Example: `https://tmerapi:8203/api/protected/admin/groups/bruceGroup`
• For each resource, usually there are four APIs:
  – Search
    • HTTP GET
    • https://tmerapi:8203/api/protected/admin/groups?name=cisl
  – Add or Update:
    • HTTP PUT
    • https://tmerapi:8203/api/protected/admin/groups/bruceGroup
  – Get Detail Data
    • HTTP GET
    • https://tmerapi:8203/api/protected/admin/groups/bruceGroup
  – Delete Data
    • HTTP DELETE
    • https://tmerapi:8203/api/protected/admin/groups/bruceGroup

• Docs & Sample codes
  https://wiki.ucar.edu/display/weg/People+REST+APIs+for+admins
People REST APIs – Search People

• **Search internal persons (staff & visitors)**
  – Backward compatible with PDB 1.0
  – E.g. https://tmerapi:8203/api/internalPersons?name=sun

• **Search external persons (collaborators)**

• **Search persons (staff, visitors & collaborators)**
  – Provide the same search functionality as in People Search Interface
  – Supports a lot of parameters
  – Best way to construct URL is to build search in the interface, then copy/paste everything after ? In URL to Person search api: https://tmerapi:8203/api/persons?
People REST APIs – Groups

• **Group Members**
  – Individuals
  – Sub groups
  – Orgs
  – SQLs
  – REST APIs (in 2.1d)

• **Group can be used as email alias**
  – Mailboxes
  – External emails

• **Group data will be sync-ed to LDAP for authorization at real time.**
Demo

• Search group
• Add or update group
• Set group permissions
• Group members will be pushed to LDAP immediately
• Delete a group
Possible Uses of People DB

• Use LDAP for authorization
• Create & Manage global GIDs in People Search
• Keep your people data in sync with master metadata
• Refactor your People related scripts to use People REST API or LDAP as your trusted data source
• Add your external users to PDB collaborators
• Integrate your site/service user registration with PDB
• Automatic UNIX user and group creation using PADL nss_ldap plugin