UG Central SSO

Proposal for new Apache Usergrid

- Goals
- Architecture
- API Login Flow for Admin User
- Browser Login Flow for Admin User
  - Browser Login Flow Sequence Diagram
- Admin User Registration Flow
- New Usergrid Endpoints
- Identity Manager Requirements

Goals

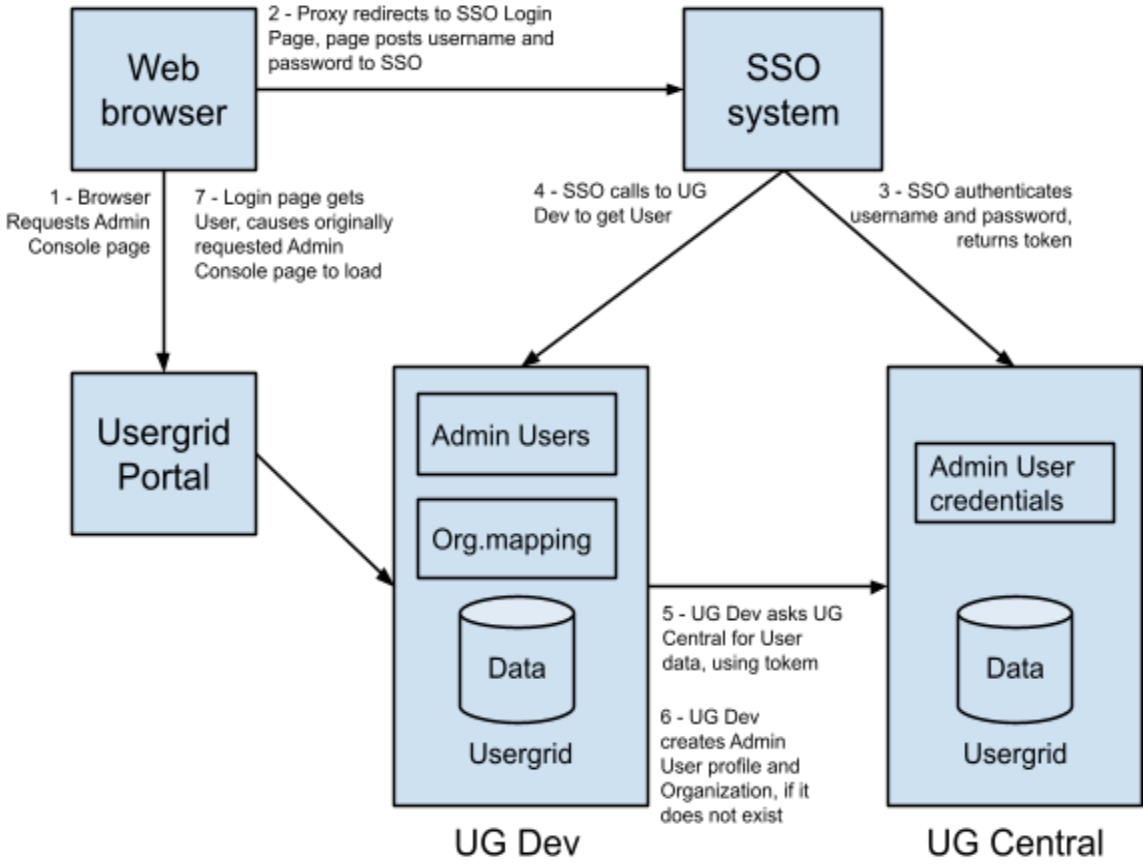
Make it possible for multiple Usergrid systems to use one central Usergrid system for Admin User authentication.

Architecture

There are four systems in play:

- **SSO**: This is an SSO system that presents user registration page, login page and calls UG Central to register and authenticate Usergrid Admin Users. This is not part of Usergrid and could be custom software or some 3rd party SSO system.
- **Proxy**: A proxy that will redirect Usergrid Admin Console requests without an access token to the SSO for login or registration.
- **UG Central**: the central Usergrid cluster that holds all user data. SSO calls UG Central to store and retrieve user credentials.
- **UG Dev**: a Usergrid cluster that uses UG Central to authenticate Admin User access tokens, and create Admin User and Organizations for users known in UG Central.

Note that the SSO and Proxy systems coordinate things, but Usergrid does not call or depend on them. SSO calls Usergrid APIs but Usergrid never calls SSO.



API Login Flow for Admin User

Here's the flow for Admin User login via API.

1. Client POSTs Admin User Credentials to the SSO system, in the same format as it would have posted to the Usergrid /management/token endpoint.

```
https://SSO.example.com/auth

   { "username": "joe",
     "password": "foobar",
     "grant_type": "password" }
```

2. SSO POSTs to UG Central with the same data, gets back access token.

3. SSO POSTs to UG Dev, ~~authenticating as a superuser~~, passing access token to UG Central user endpoint.

```
   POST http://ugcentral.example.com/management/externaltoken

   { "ext_access_token": "<user access token from UG Central>" }
```

4. UG Dev does a GET on UG Central to get user profile data, or an error.

```
   GET http://ugcentral.example.com/management/me?access_token=<token>
```

If profile already exists in UG Dev: Returns access token for accessing DUG as that admin (or calls import token)

If profile does not exist in UG Dev, then create user and user's personal organization (named same as username) and add to organization.

5. UG Dev stores the token locally as in its Token Service and returns user access token to SSO in response to SSO's post from step #3

6. SSO returns access token to client

Browser Login Flow for Admin User

Here's the flow for Admin User login via web browser.

1. User browses to https://example.com/usergrid/ without an access token and the Proxy redirects them to SSO

```
https://SSO.example.com/accounts/sign_in?callback=https%3A%2F%2Fexample.com%2Fusergrid%2F%23!%2Forg-overview
```

2. SSO shows the Login page, user fills in username and password, clicks Login button.
   Login page POSTs credentials to SSO:

```
https://SSO.example.com/accounts/sign_in?callback=https%3A%2F%2Fexample.com%2Fusergrid%2F%23!%2Forg-overview

   { "username": "joe",
     "password": "foobar",
     "grant_type": "password" }
```

3. SSO POSTs to UG Central with the same data, gets back access token.

4. SSO POST to UG Dev, ~~authenticating as a superuser~~, passing access token to UG Central user endpoint.

```
   POST http://ugcentral.example.com/management/externaltoken

   { "ext_access_token": "<user access token from UG Central>" }
```

5. UG Dev does a GET on UG Central to get user profile data, or an error.

```
   GET http://ugcentral.example.com/management/me?access_token=<access token from UG Central>
```

- If profile already exists in UG Dev: Returns access token for accessing DUG as that admin (or calls import token)
- If profile does not exist in UG Dev, then create user and user's personal organization (named same as username) and add to organization.

6. UG Dev stores the token locally as in its Token Service and returns user access token to SSO in response to SSO's post from step #4

7. SSO returns access token to login page, in response to Login page's post from step #2

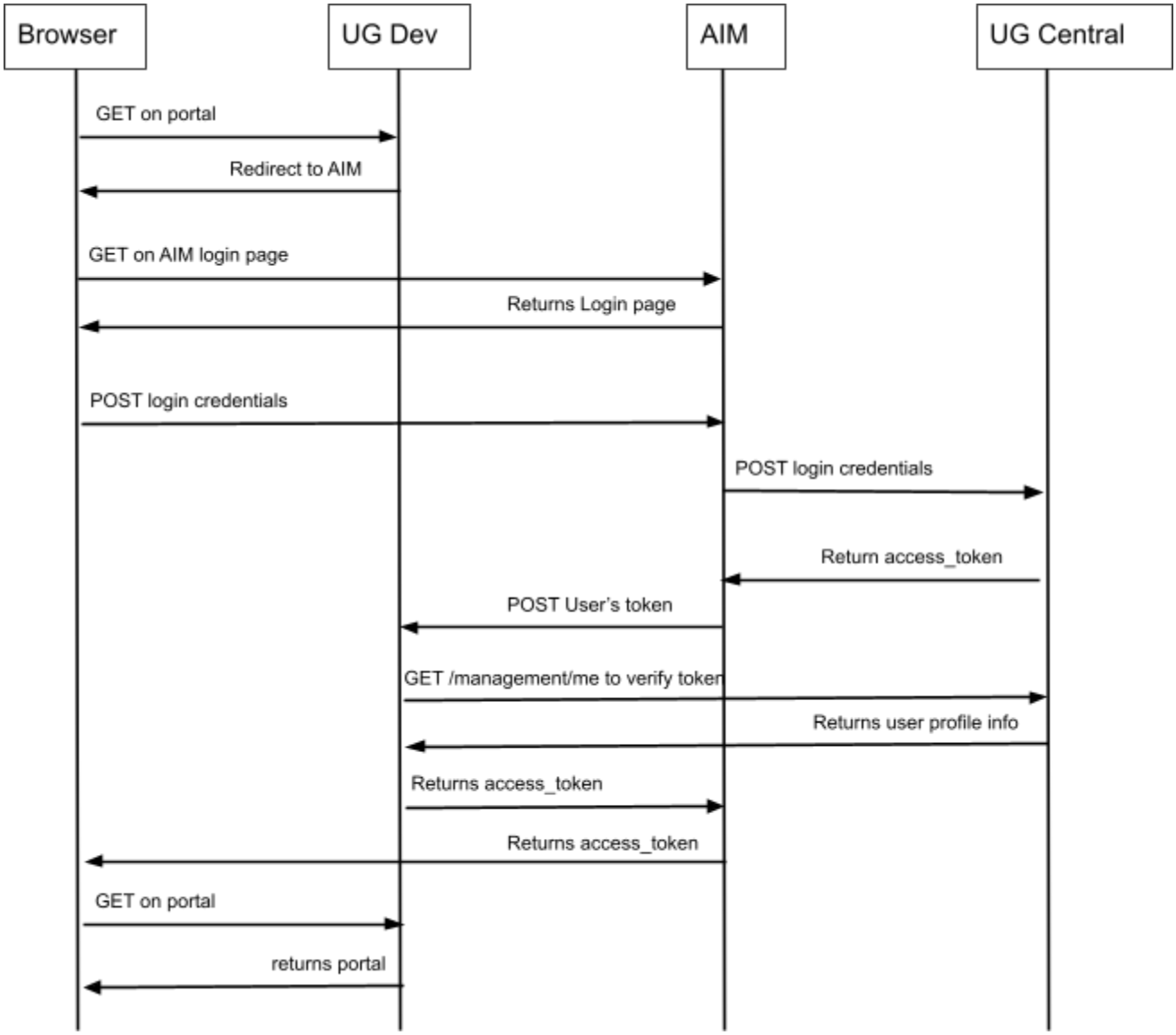
8. Login page causes browser to load originally requested page, but with token:

```
https://example.com/usergrid/#!/org-overview?access_token=YWMtIotUOt03EeSsQevRu
ZL-3AAAAUy4fZrGcdntEep8&uuid=e80dbf4a-0750-11e3-b3b9-bd25e55ac4a6&admin_email=j
oe@example.com
```

9. User is logged-in and sees API BaaS portal.

Browser Login Flow Sequence Diagram

Illustrates the order of interactions between the main components of UG Central architecture.



Admin User Registration Flow

New user creation is handled by the SSO system. Admin Users registers with SSO server and must specify username, password and email address. Using existing Usergrid APIs, the SSO server posts user credentials to UG Central, where they are stored.

Later, when a new Admin User logs in for the first time (see the Admin User Login Flow above), UG Dev will create its own local copy of the User, create the user's personal Organization and add the user to it.

New Usergrid Endpoints

Here are the new Usergrid endpoints we will need to support UG Central SSO.

/management/auth/usergrid

```
/**
 * <p>
 * Validates access token from other or "external" Usergrid system.
 * Calls other system's /management/me endpoint to get the User
 * associated with the access token. If user does not exist locally,
 * then user and organizations will be created. If no user is returned
 * from the other cluster, then this endpoint will return 401.
 * </p>
 *
 * @param ui              Information about calling URI.
 * @param extAccessToken Access token from external Usergrid system.
 * @param ttl             Time to live for token.
 * @param callback        For JSONP support.
 * @return                Returns JSON object with access_token field.
 * @throws Exception      Returns 401 if access token cannot be validated
 */
@GET
@Path( "/externaltoken" )
public Response validateExternalToken(
   @Context UriInfo ui,
   @QueryParam( "ext_access_token" ) String extAccessToken,
   @QueryParam( "ttl" ) @DefaultValue("-1") long ttl,
   @QueryParam( "callback" ) @DefaultValue( "" ) String callback )
   throws Exception {

  return validateExternalToken( extAccessToken, ttl, callback );
}
```

SSO Requirements

This proposal assumes an SSO system exists that can be configured to interact with UG Central and UG Dev via the Usergrid REST API, and to perform these functions:

- Present a Admin User Registration page does these things:
    - Register new Admin User with the UG Central Usergrid

- Present a Login page that does these things:
    - Send Admin User authentication requests a central Usergrid system
    - Validate central Usergrid system tokens with Usergrid system requested
    - Upon successful authentication, redirect browser to originally requested page

- Accept Usergrid authentication requests and forward them to Usergrid