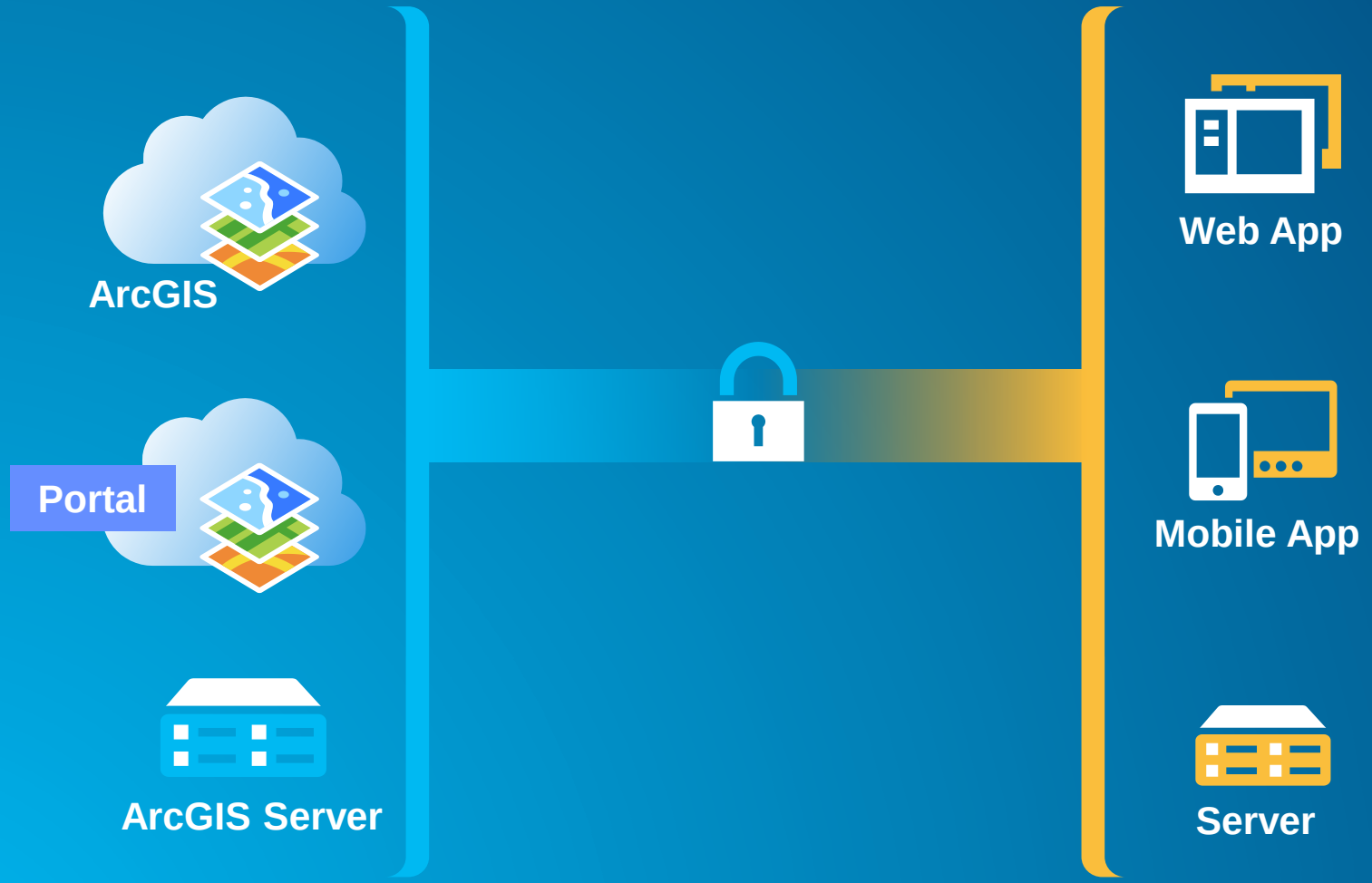
DevSummit DC

February 11, 2015 | Washington, DC

Building Secure Web Applications

James Tedrick

What We're Covering Today: Accessing ArcGIS Resources



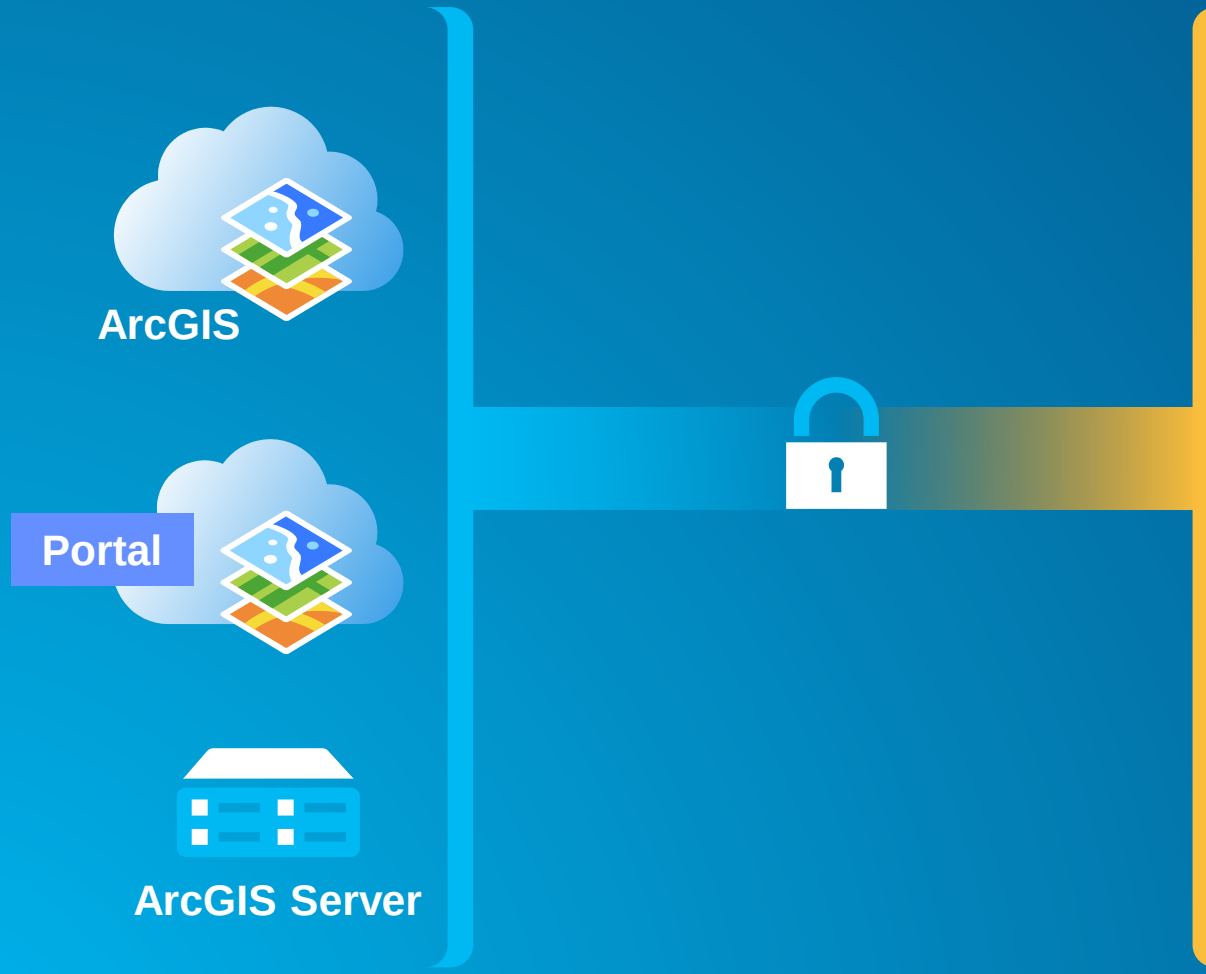
Topics covered:

- Security Configuration
- Classic Authentication
- Enterprise Logins (SAML 2.0 / OAuth)
  - User login
  - App login

Other session topics *not covered*:

- Securing Services
- Designing Enterprise Security Strategy
- AGO / Cloud Security

What We're Covering Today: Accessing ArcGIS Resources



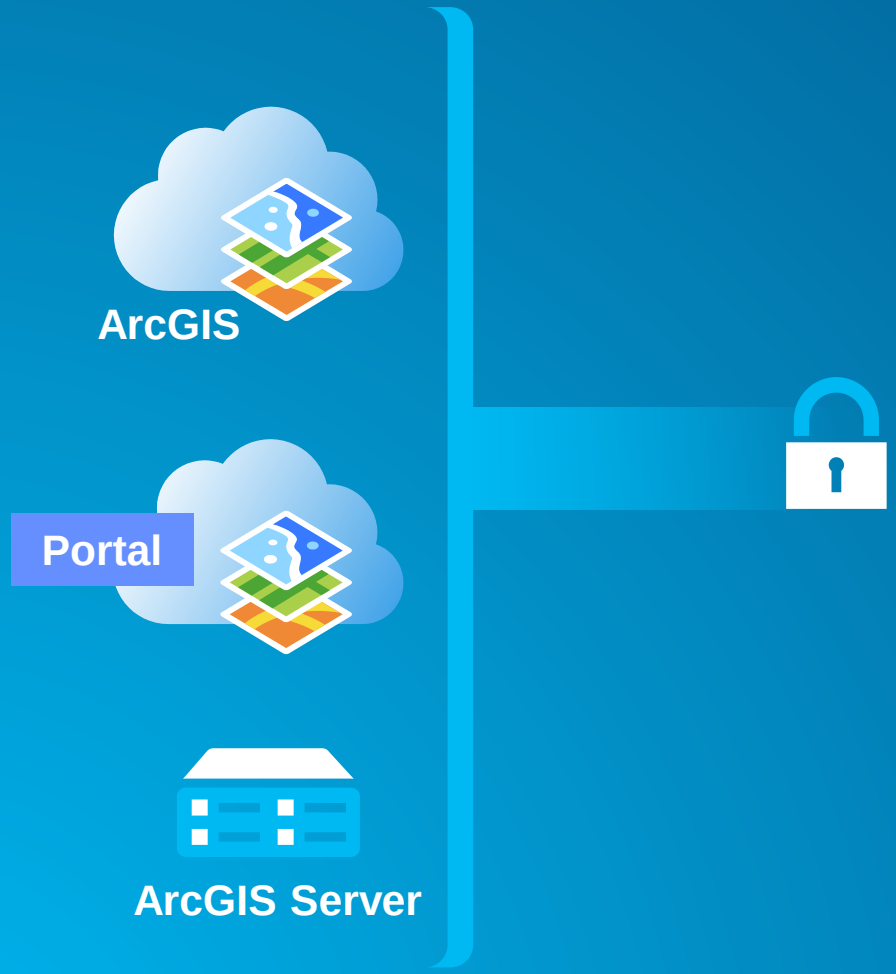
Classic Authentication

- Portal for ArcGIS 10.2
- ArcGIS Server
- ArcGIS Online

Enterprise Logins (SAML 2.0 / OAuth)

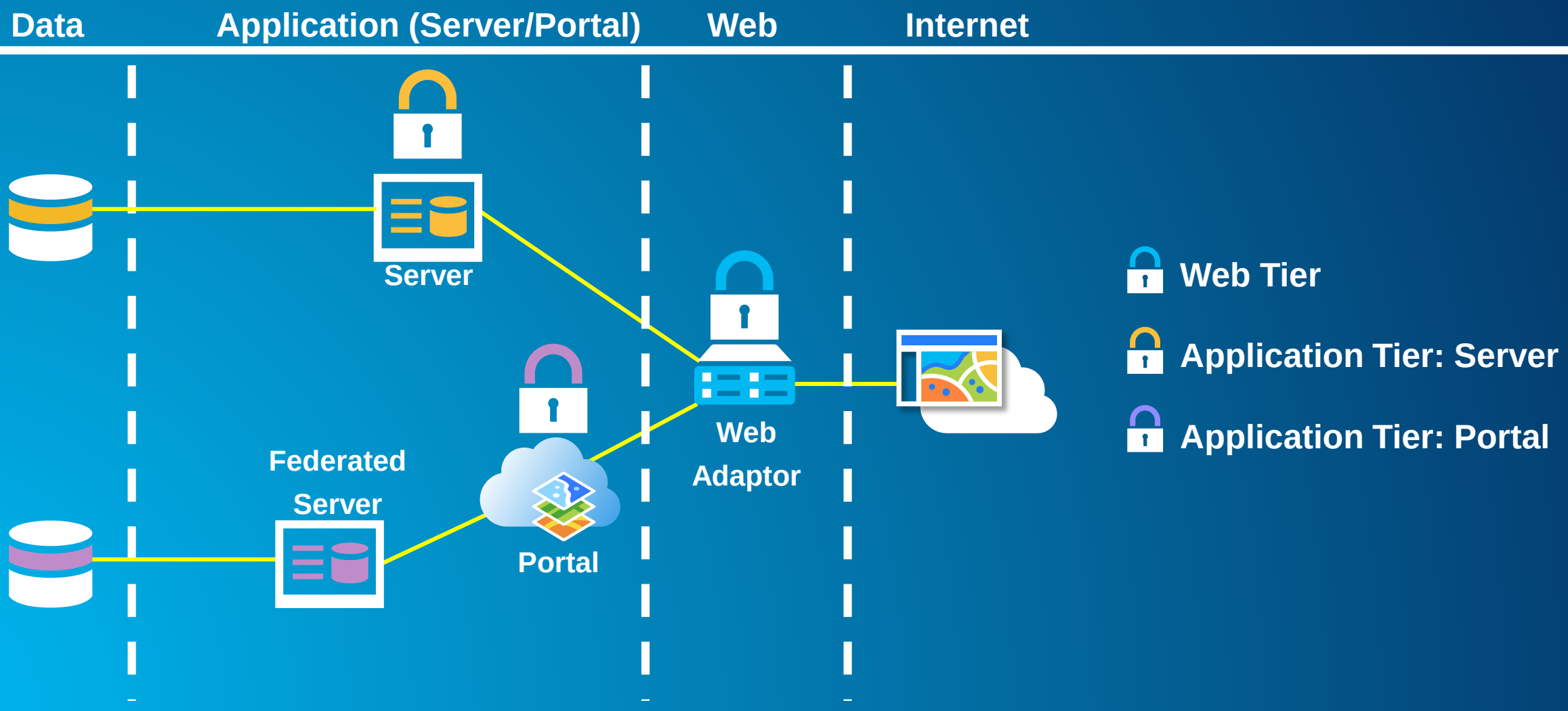
- ArcGIS Online
- Portal for ArcGIS 10.3
- User login
- App login
- Different workflows depending on application

Security Configuration



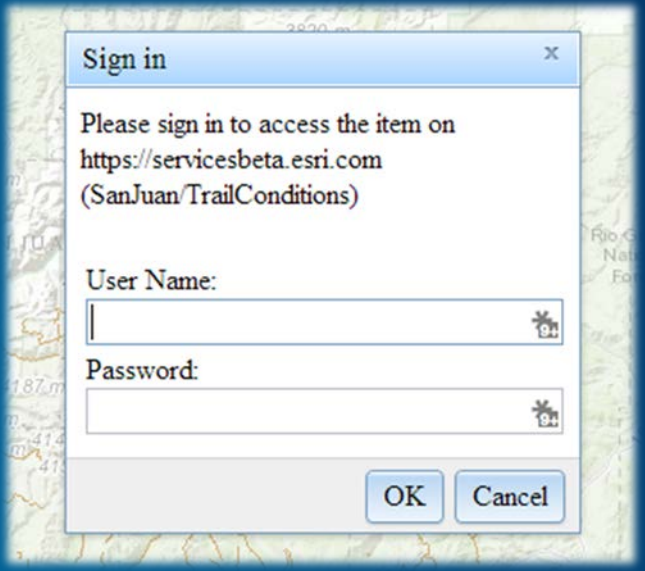
- Outside direct *developer* control
  - Configured by GIS Admin
  - Specific to a given GIS site
- Occurs at differing levels
  - Application (ArcGIS Server, Portal)
  - Web Server (IIS, Apache)
- Verifies against a user store
  - Application
    - ArcGIS Server, Portal, AGO internal store
    - External store via SAML
    - ArcGIS Server: LDAP (AD) via direct lookup
  - Web Tier
    - Active Directory / LDAP via direct connection
  - Groups/roles can be stored elsewhere

ArcGIS Platform Security Locations



IdentityManager

- Uniform class across web APIs for logging in
- Automatically handles login process for all secured services
- Updates tokens to keep access current



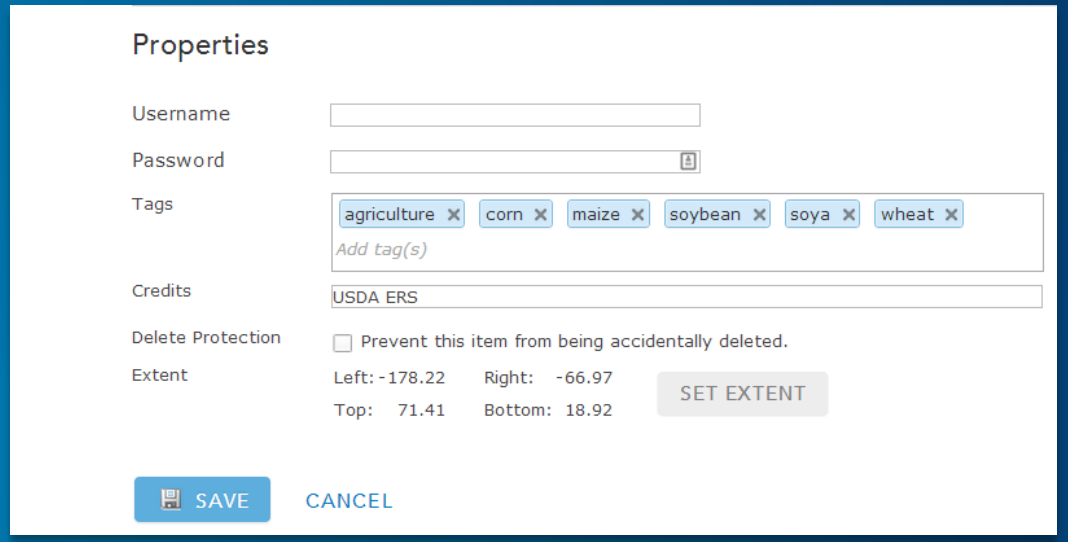
Single Sign On (SSO) & PKI

- Integrated Windows Authentication (IWA)
  - Sign in once to Windows (i.e., login to your computer)
  - Supporting apps supplied with Windows credentials
- ArcGIS Server (no federation) – enable Web Tier authentication with a web adaptor on IIS
- ArcGIS Online / Portal – Configure SAML landing page to use Single Sign On

- PKI - Federal identity standard
  - 2 Factor authentication (Card & PIN)
  - CAC card contains certificate
  - User supplies PIN that is matched against card, certificate is forwarded to application

Tokens

- Instead of using username & passwords for each request, we authenticate once & get a token (text that has encrypted authorization)
- Token lifespan – can be set during request & renewed
- Storing Tokens
  - Embedded within proxy – expose secured services
  - ArcGIS Online & Portal can store credentials and proxy on demand



Classic Authentication

Classic Authentication

- After username / password submission, receive token string
- Tokens have a set lifetime (default/requested length)
- Token Generation URLs:
  - Portal Tokens: ArcGIS Online, Portal for ArcGIS, ***Federated*** ArcGIS for Server <PORTAL URL>/sharing/generateToken (e.g., http://myportal.mycompany.com/portal/sharing/generateToken)
  - Server Tokens: ArcGIS for Server <SERVER URL>/tokens (e.g. http://myportal.mycompany.com/server/tokens)

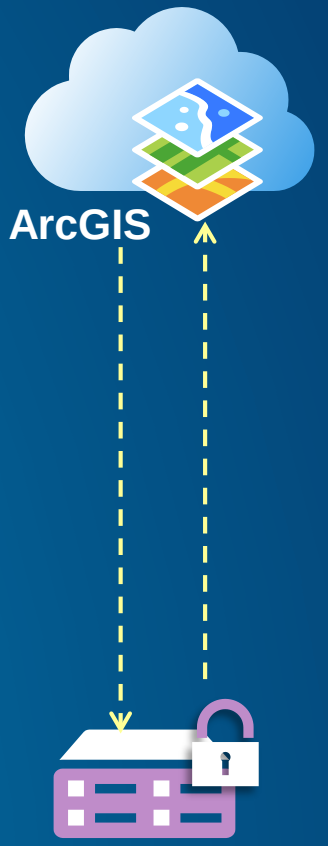
Classic authentication issues

- Application has access to user’s password
- Application is responsible for full login process
- Does not support enterprise logins
- Developer must manually support app usage tracking

Enterprise (SAML/OAuth) Logins

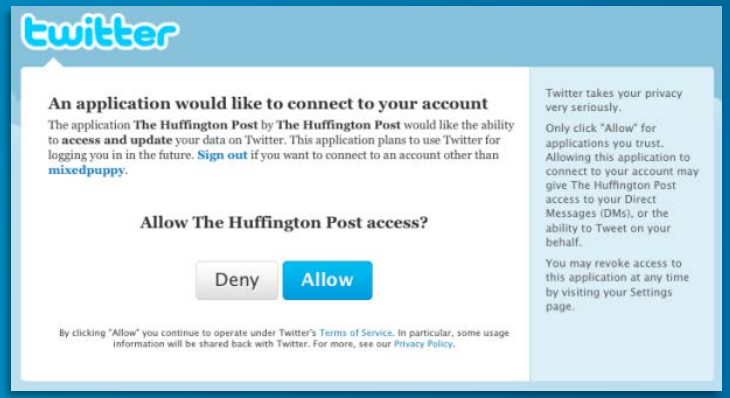
SAML - Security Assertion Markup Language

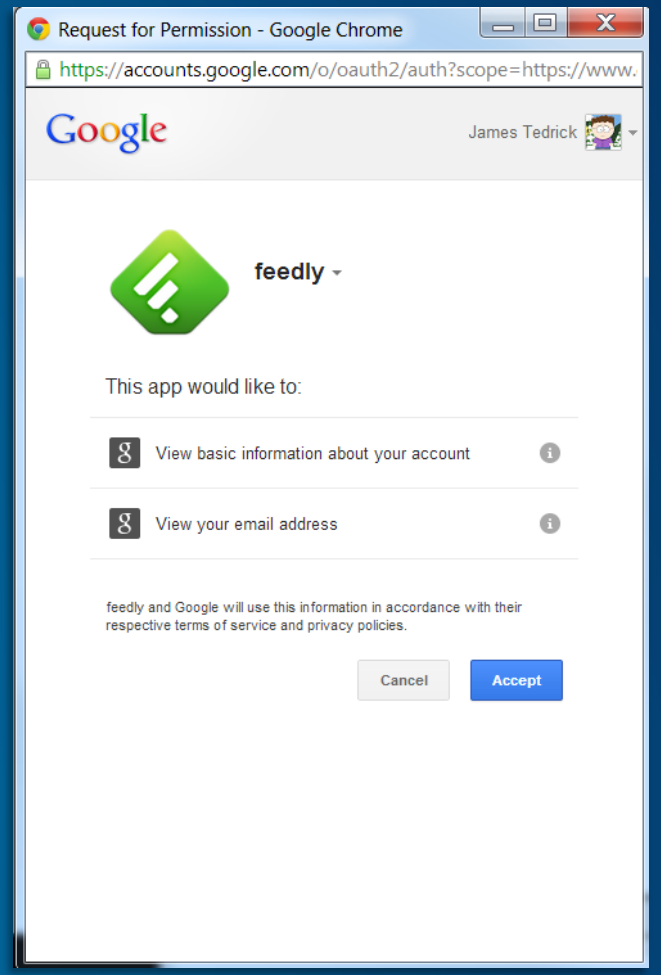
- Separates Authentication (login) from Resources (AGO)
- ArcGIS Online can use an organization's login information (i.e., Active Directory)
- Set up:
  - ArcGIS Online (AGO org admin)
  - Login provider (enterprise admin)
- Enables Single Sign-On into ArcGIS Online
- From app developer perspective, process is uniform- you interact with ArcGIS Online, not the identity provider



OAuth

- Differentiates between application server, authentication server
- Authentication server logs user in, checks for user acceptance of application
- Application server *does not* see username/password as they are entered
- Application does get access token after authorization





OAuth logins workflows

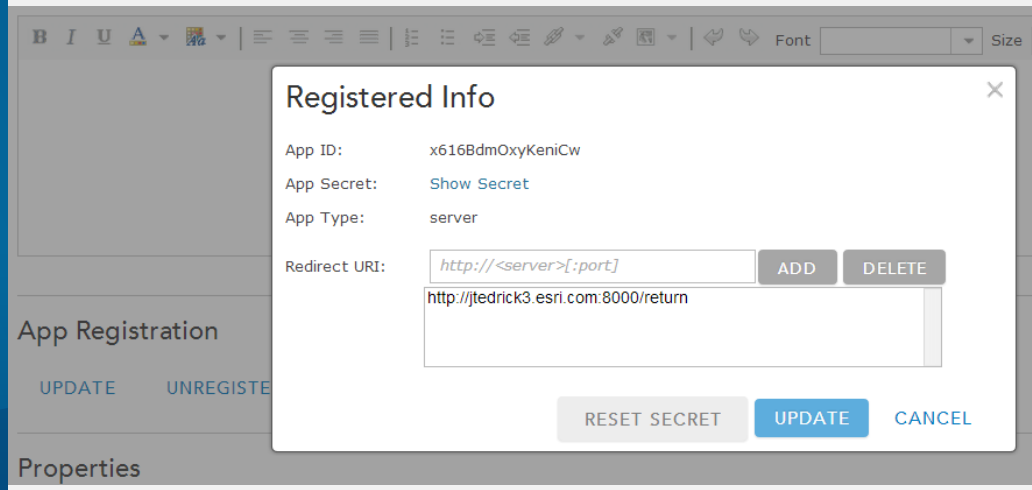
- **User login – User needs their *own* an ArcGIS Online account**
- **Application login – Users uses *your* ArcGIS Online account**
- **2 endpoints used in processes**
  - **https://www.arcgis.com/sharing/oauth2/authorize**
  - **https://www.arcgis.com/sharing/outh2/token**

OAuth login key properties

- Set up in ArcGIS Online's Item Content or Developer Dashboard
  - `redirect_uri` – resource to load when presenting new credentials
  - `appId` – unique ID of application in ArcGIS Online
  - `appSecret` – secret key used with appId (appId's 'password')
- appSecret should ***never*** be exposed to user
  - Including embedded in mobile application
- appId & appSecret can be reset by application owner

Demo

Registering an app

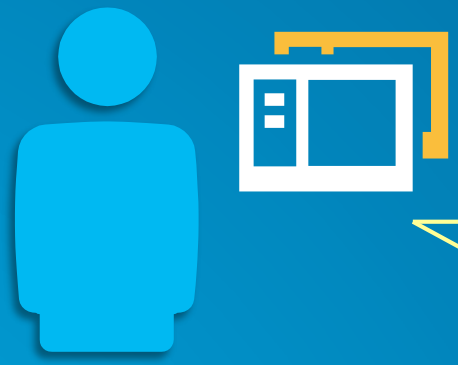


OAuth logins – User logins

- Access user's data & maps
- Tasks consume user's credits
- Logins can be either ArcGIS or Enterprise (i.e., Active Directory, LDAP)
- Process by application type:
  - Login in HTML/JS – 1 step (implicit grant)
    1. Access /authorize, load resulting redirect containing token
  - Login at application (iOS/Android) or web server (.Net/PHP) – 2 step (authorization grant)
    1. Access /authorize, load resulting redirect with authorization code
    2. Access /token with code, receive token

Warning:
Simplified Diagrams Ahead

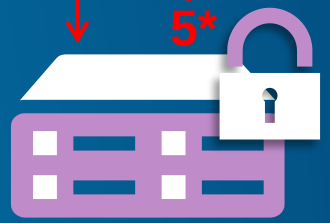
All communication between ArcGIS Online and Identity Management occurs via client through redirects URLs



- **ArcGIS Online does not contact Identity Provider directly**
- **Identity Provider only needs to be seen by the client, not the internet**

Your application server

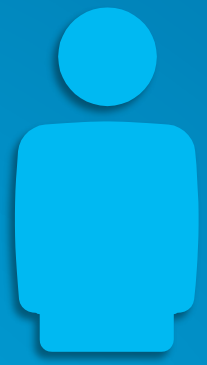
ArcGIS Online



Identity Provider

User Login – Web Applications

1. Application loads into client
2. Application requests authorization by opening https://www.arcgis.com/sharing/oauth2/authorize
3. ArcGIS Online redirects to organization login



Your application server

1

2

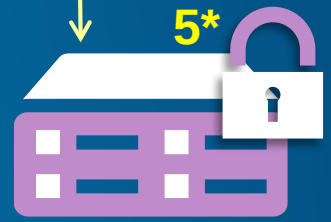
ArcGIS Online

6

4

3*

5*

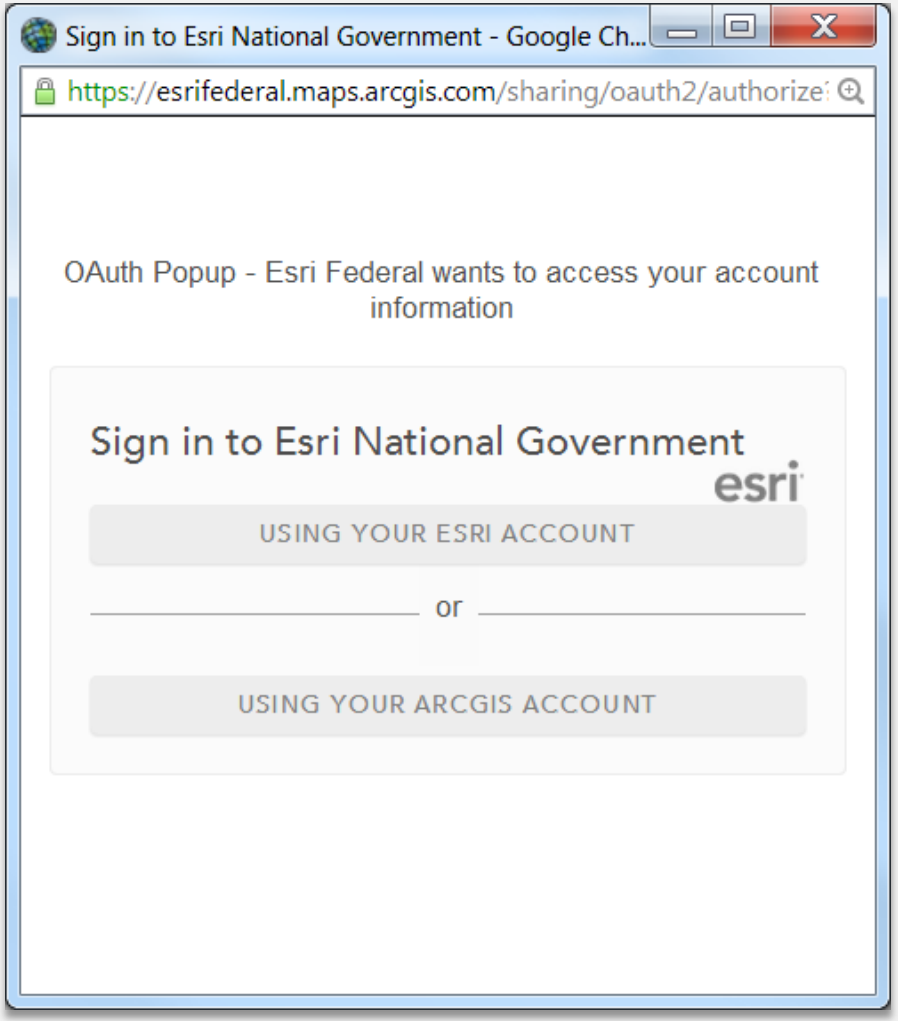


Identity Management

4. User logs in using login system
5. Login system authorizes user to AGO
6. ArcGIS Online gives application an access by loading redirect_uri with access token appended

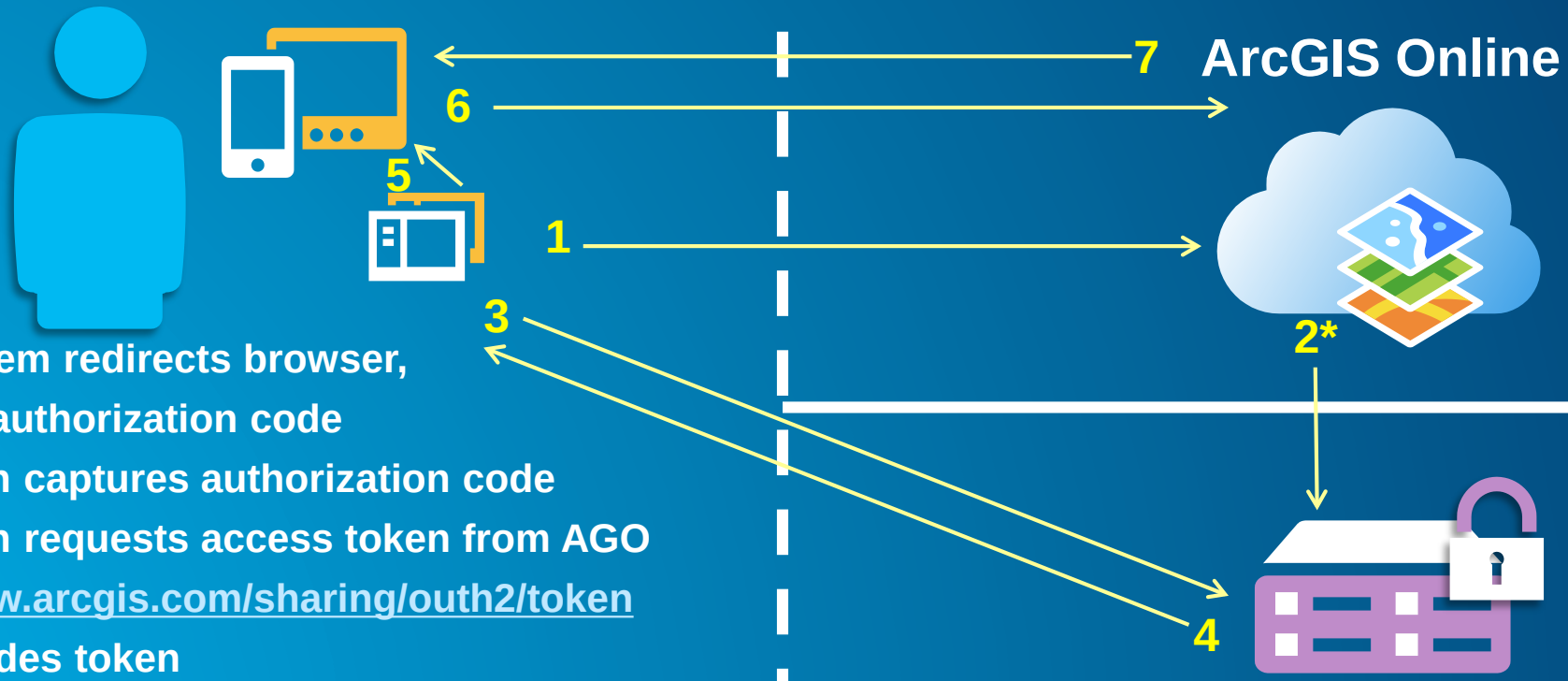
Demo

User Web Login



User Login – Desktop/Device App

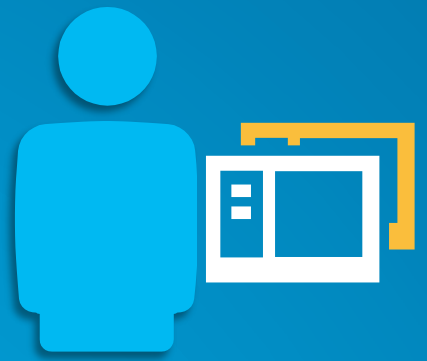
1. Through an embedded web browser, application requests authorization by opening https://www.arcgis.com/sharing/oauth2/authorize
2. ArcGIS Online redirects to organization login page
3. User logs in using login system

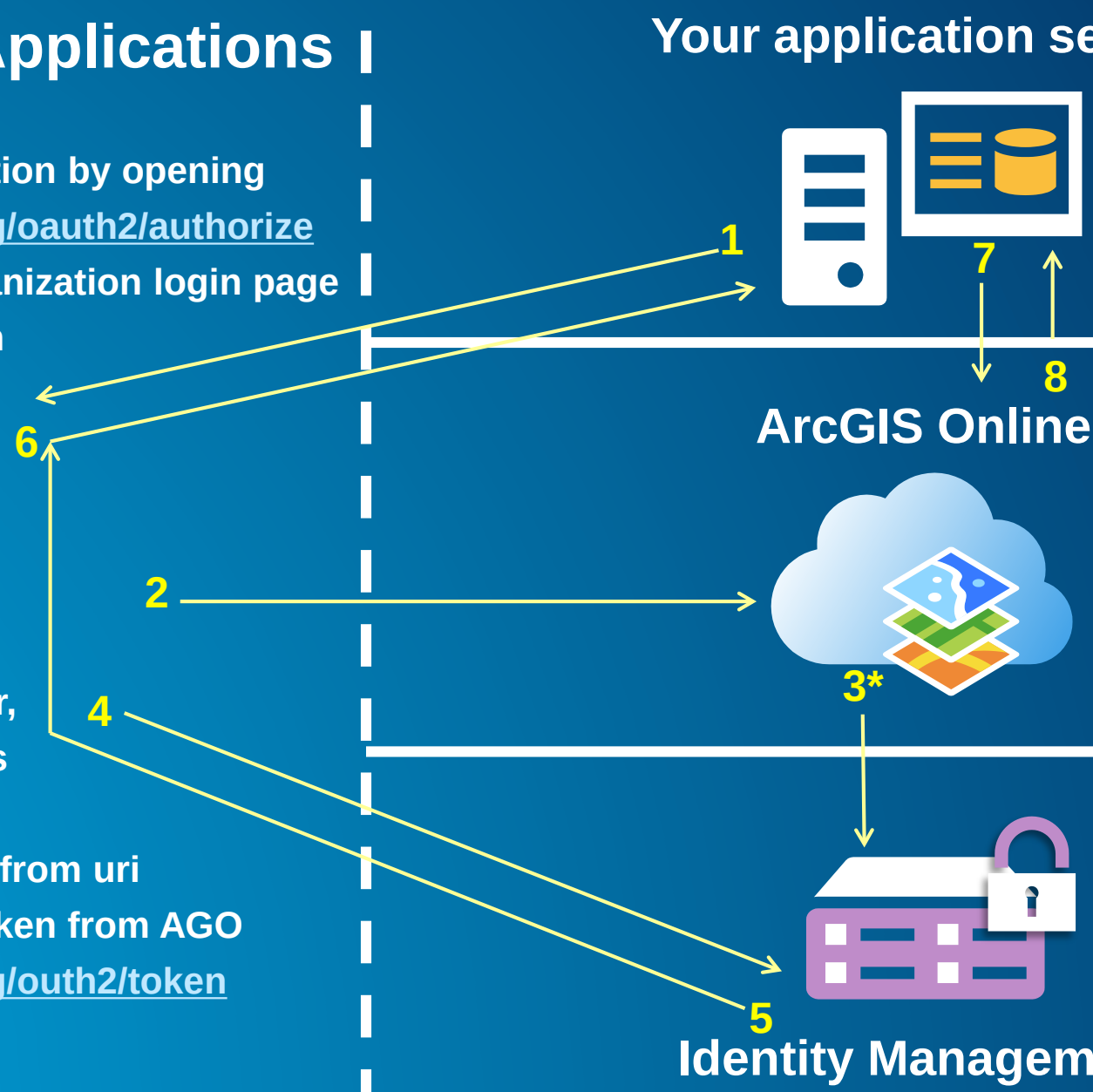


4. Login system redirects browser, providing authorization code
5. Application captures authorization code
6. Application requests access token from AGO https://www.arcgis.com/sharing/outh2/token
7. AGO provides token

User Login – Server Applications

1. Application loads into client
2. Application requests authorization by opening https://www.arcgis.com/sharing/oauth2/authorize
3. ArcGIS Online redirects to organization login page
4. User logs in using login system
5. Login system redirects browser, providing authorization code as uri parameter
6. Server gets authorization code from uri
7. Application requests access token from AGO https://www.arcgis.com/sharing/outh2/token
8. AGO provides token



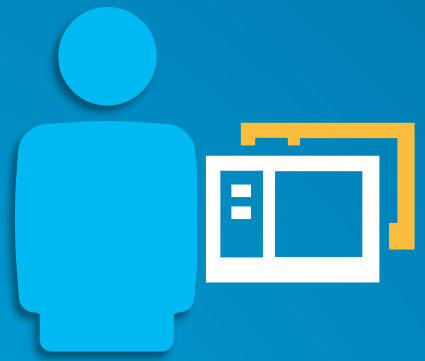


Application Login

- Uses appID, appSecret as application ‘username’ & ‘password’
- User is never aware of ArcGIS Online (aside from the documentation ☺)
- Developer’s responsibility to provide access controls
  - Otherwise, you’re offering your credits to everyone!
- What you can do:
  - Access AGO tasks (Geocoding, routing, etc.)
  - Access application owner’s private items stored in ArcGIS Online
  - Search public items in ArcGIS Online (NOT Organizational)

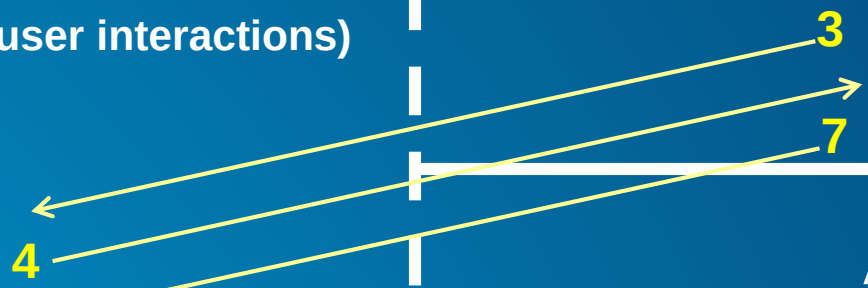
Application Login

1. Application requests authorization by opening https://www.arcgis.com/sharing/oauth2/token (normally done independent of user interactions)
2. ArcGIS Online provides a token
3. Application loads into client



4. Client requests an operation that makes use of AGO resources
5. Server requests resources with token
6. ArcGIS Online provides response
7. Possible further processing; response is delivered to client

Your application server



ArcGIS Online

Demo

Application Login

```
this._getToken = function () {
    // request parameters
    var post_data = qs.stringify({
        'client_id' : _self.APPID,
        'client_secret' : _self.APPSECRET,
        'grant_type' : 'client_credentials'
    }),
        // request setup
        post_options = {
            host: 'www.arcgis.com',
            port: '443',
            path: '/sharing/oauth2/token',
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-
                'Content-Length': post_data.length
            }
        },
        //Make the request
        post_req = https.request(post_options, funct
            postres.setEncoding('utf8');
            postres.on('data', function (chunk) {
                var tokenObj = JSON.parse(chunk);
                _self.token = tokenObj.access_token
                console.log(new Date());
                console.log(tokenObj.access_token);
                console.log('--------------------');
                //We'll set the refresh 30 seconds
                _self._refresh = setTimeout(_self._
            });
        });
    // post the data
    post_req.write(post_data);
    post_req.end();
};
```

Resources

- **Presentation Samples:**
  - **IdentityManager Information:** https://developers.arcgis.com/javascript/jsapi/identitymanager-amd.html
  - **Proxy Information:** https://developers.arcgis.com/javascript/jshelp/ags_proxy.html
  - **OAuth User Login:** https://developers.arcgis.com/en/javascript/jssamples/portal_oauth_inline.html
  - **Application Login demo:** https://github.com/tedrick/agoServerLoginExample

- **Developer Page:** https://developers.arcgis.com/authentication/
- **JS Application Boilerplate:** https://github.com/Esri/application-boilerplate-js
- **JS Sample: iOS Sample:** https://github.com/Esri/arcgis-runtime-samples-ios/tree/master/OAuth%20Login%20Sample
- **Developer Libraries (listed by OAuth group):** http://oauth.net/2/

Understanding our world.