Security and ArcGIS Web Development

Heather Gonzago and Jeremy Bartley

Agenda

- **Types of apps**
- Traditional token-based authentication
- OAuth2 authentication
- User login authentication
- Application authentication
- Resource proxy

Types of apps

First need to figure out…

- How are you interacting with the application?
- Are the end users known to the application?
- User-login
  - Application allows user to log in with valid credentials
- Are the end users unknown to the application?
- App-login
  - App provides credentials within the app itself

Agenda

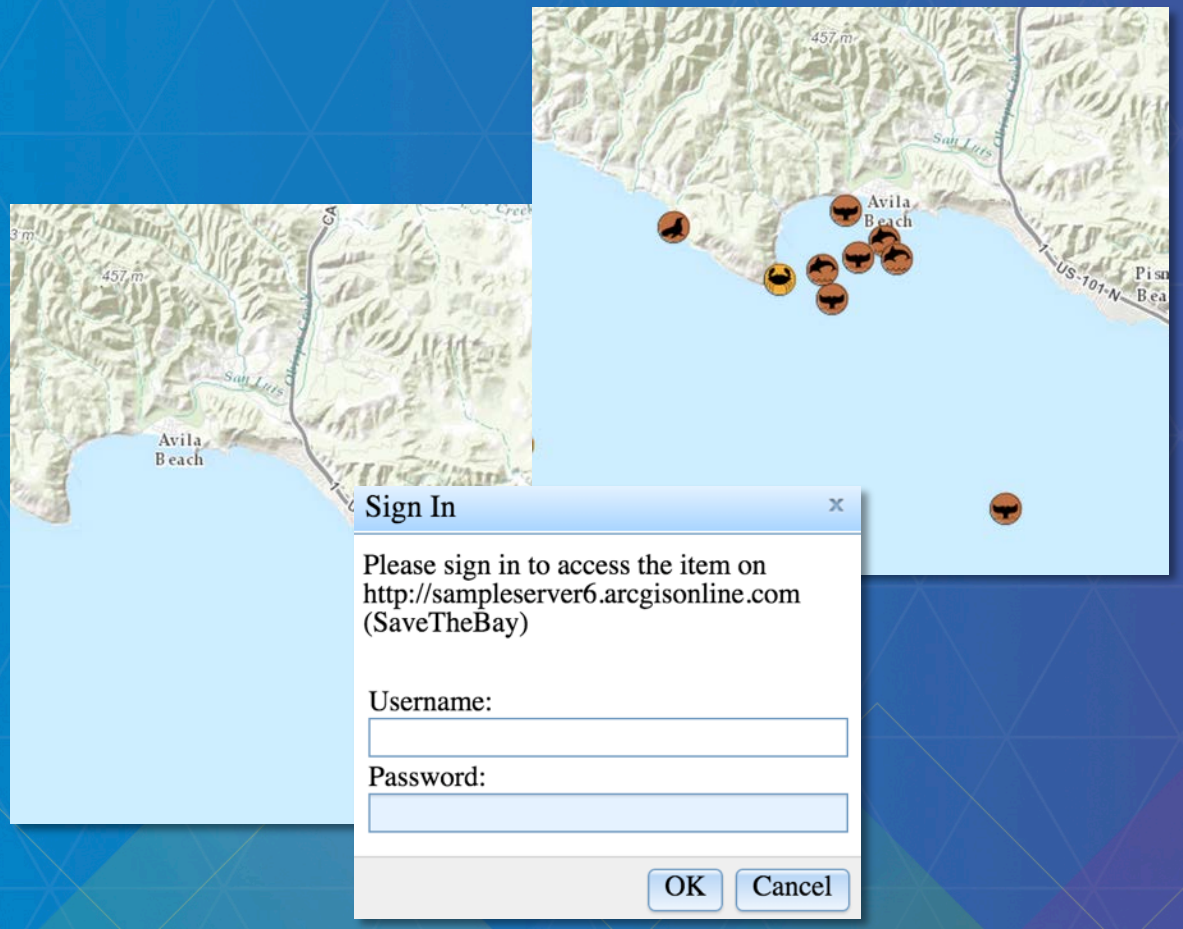
- Types of apps
- **Traditional server-based token authentication**
- OAuth2 authentication
- User login authentication
- Application authentication
- Resource proxy

Traditional server-based token authentication

- Classic way to access secured services
- Username and password sent over https
- `/generateToken` call
- Short-lived token response
- Handled via SDK’s Identity Manager component
- Everything handled on the client
- Web application responsible for keeping credentials secure

generateToken

- Access secured service with server-based token authentication
- Identity Manager takes care of most of this work
- Challenge for credentials
- `https://<server>/sharing/rest/generateToken` called when credentials are passed
- Token appended and used to unlock these services



Demo: generateToken

generateToken limitations

- The web application has access to credentials
- The application is responsible for recognizing when to prompt for login
- No enterprise logins
- Need to sign in every time calls made to a secure service
- Can’t track how the app is being used
- Can’t list in Marketplace

Agenda

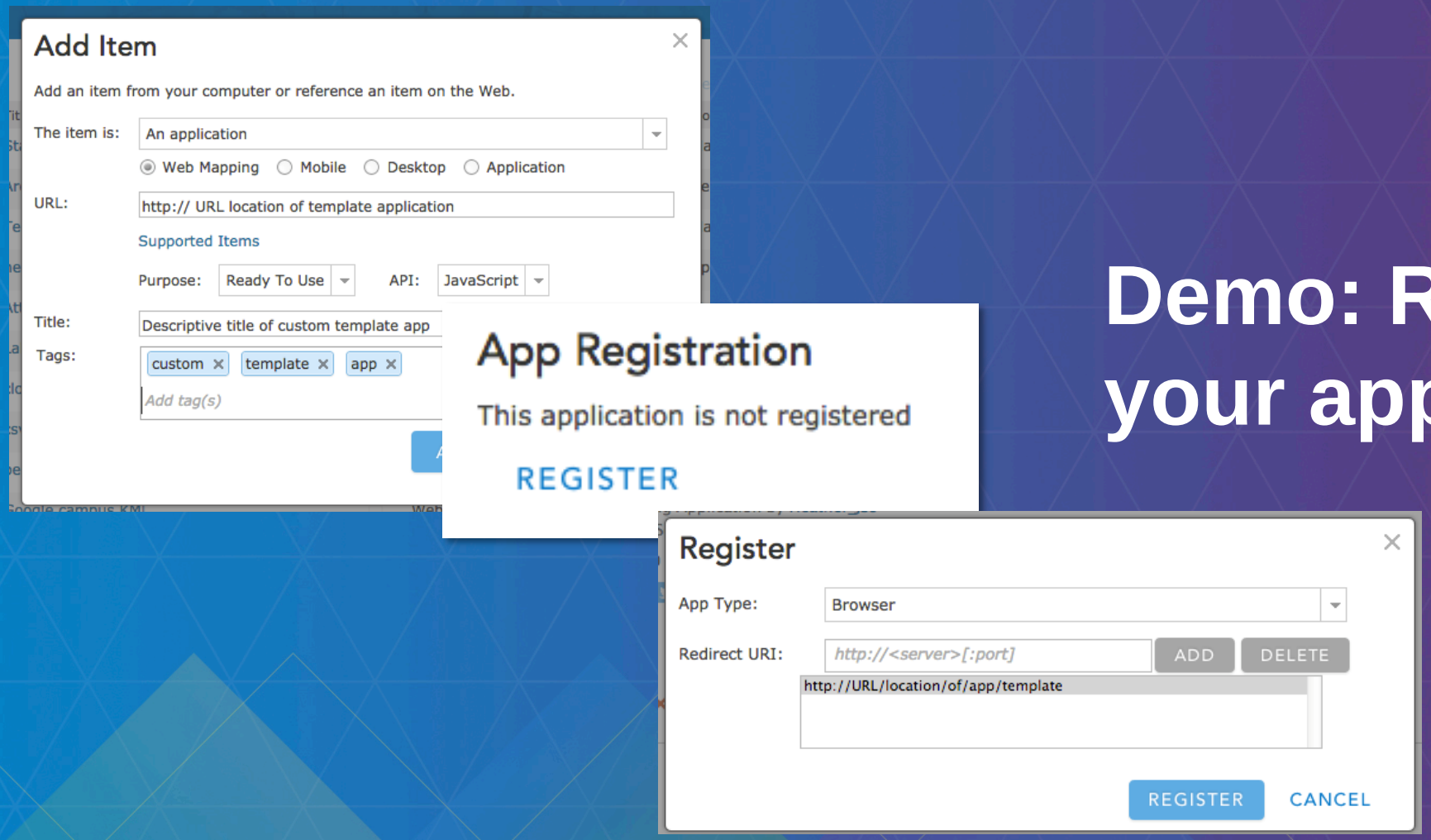
- Types of apps
- Traditional server-based token authentication
- **OAuth2 authentication**
- User login authentication
- Application authentication
- Resource proxy

OAuth2 authentication: Why to use this and when?

- The “**Why**?”
  - The registered app provides information on how the application is being used, whom is accessing it, credit usage, etc?
- The “**When**?”
  - Access users’ private data
  - Create and publish content
  - Access premium content and services on AGO

Registering your app

- Possible in organizational ArcGIS Online, on-premise portal (as of 10.3), or Developers site
- Generates:
  - App ID (`client_id`)
  - App Secret (`client_secret`)



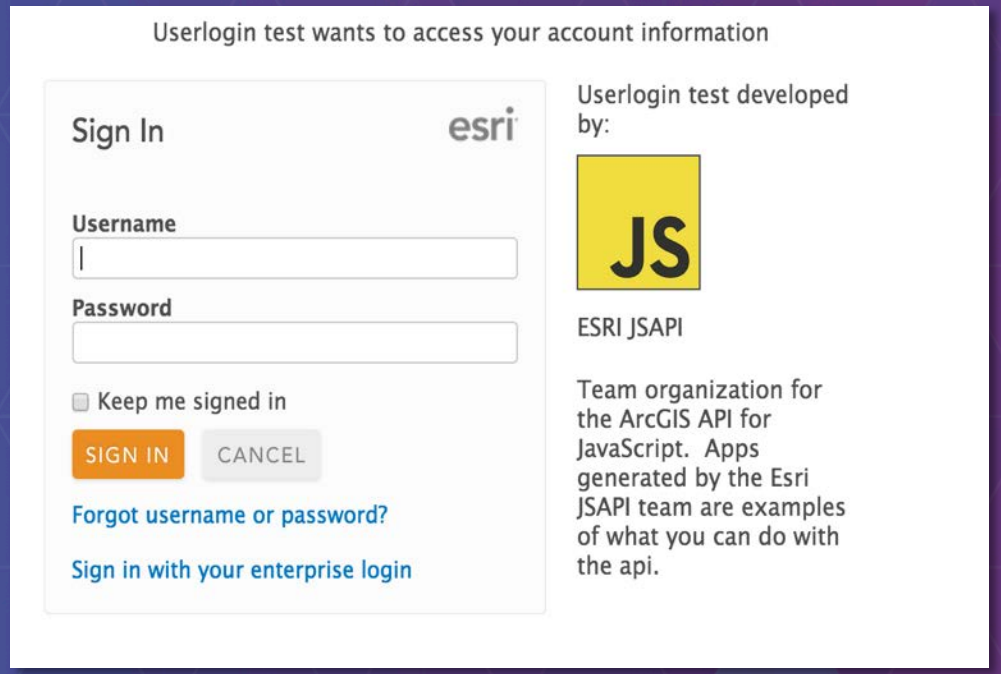
Demo: Registering your app

Agenda

- Types of apps
- Traditional token-based authentication
- OAuth2 authentication
- **User login authentication**
- Application authentication
- Resource proxy

OAuth2: User login authentication

- Works with named users
- Server-side login page
- Provides single sign on
  - If already in active portal session, user is not required to enter credentials again
  - Approval screen to grant access to the app

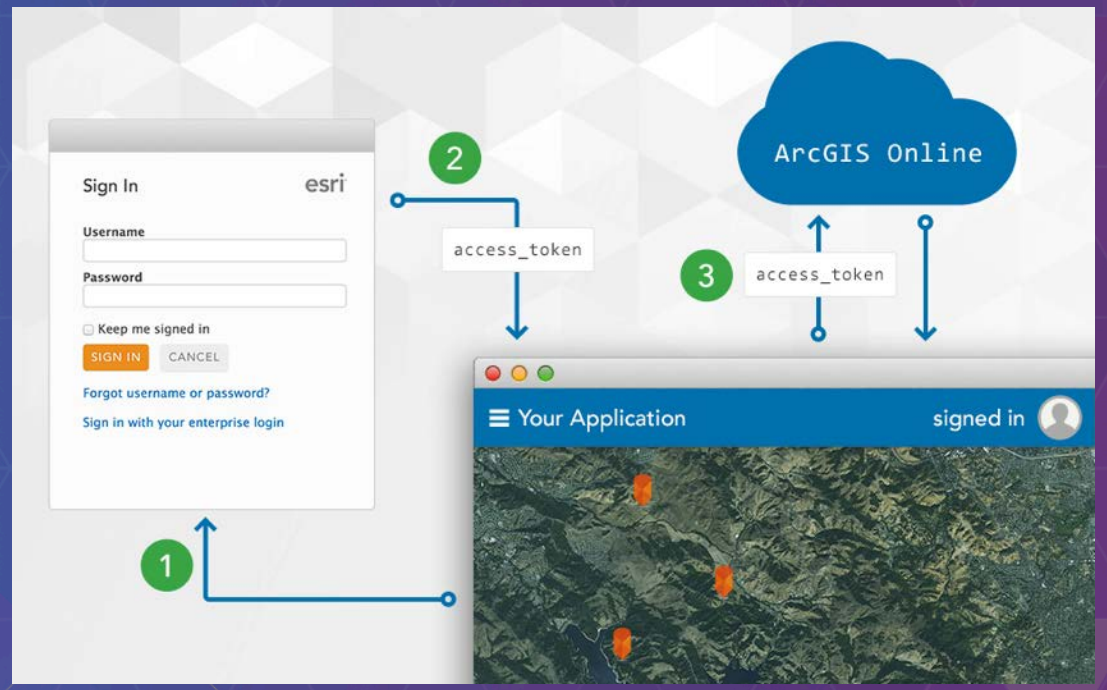


OAuth2: User login workflows

- **Working with browser-based or mobile applications?**
  - 1 step workflow, i.e. Implicit Grant
- **Desktop, mobile, or server-side web application?**
  - 2 step workflow, i.e. Authorization Grant

OAuth2: 1 step user-login workflow

1. App directs user to `/authorize` endpoint
2. Valid user/pass?
3. Redirect back to app at provided `redirect_uri`
4. `access_token` is added to URL
5. App can parse the URL for token use



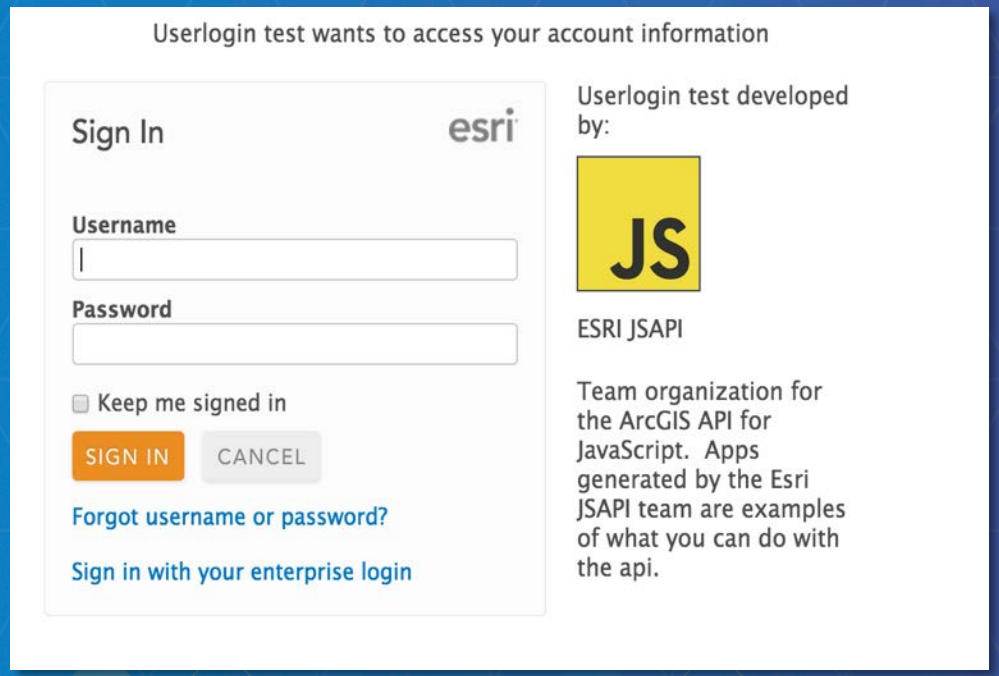
OAuth2: Identity Manager

- Client SDKS provide OAuth2 functionality via the Identity Manager
- Handles the complexity of calling endpoints and parsing tokens
- Example: JS API Identity Manager
  - OAuthInfo class -> pass in registered App ID
  - Pass this information to the identity manager

App Registration

App ID:	6PWiPJo3hQEODTpd
App Secret:	Show Secret
App Type:	browser
Redirect URI's:	http://heatherg.esri.com

```
var info = new OAuthInfo({
  appId: "6PWiPJo3hQEODTpd",
  popup: true
});
esriId.registerOAuthInfos([info]);
```



Demo: User login

Enterprise Logins

- Login to ArcGIS Online using your enterprise login (Active Directory, LDAP, …)
- Uses the SAML standard
- Setting up Enterprise Logins – Doc

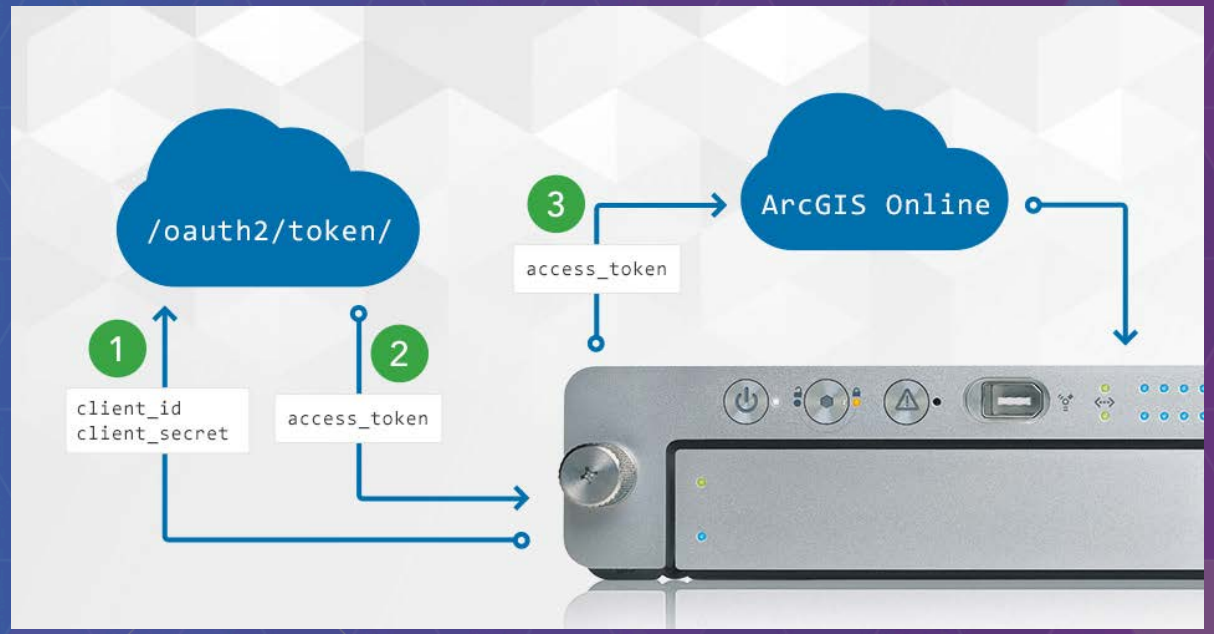
- Nothing changes for the App Developer
  - Use standard OAuth workflow (redirect user to `/authorize` URL as usual)
  - Portal detects enterprise login if configured for the organization
  - Redirects user to their enterprise provider
  - Enterprise redirects user to portal upon login
  - Portal generates token and sends it to the app

OAuth 2.0: Application Authentication

- Users of the App are unknown to the ArcGIS platform
- No login prompts
- App logs in to the ArcGIS platform on behalf of itself
- App must contain valid app credentials
- App is responsible for keeping the app secret secure

OAuth2: App authentication workflow

1. App makes POST to /token endpoint with `client_id` and `client_secret` with `grant_type=client_credentials`
2. `Access_token` in JSON response
3. Parse token and use in requests as needed.



OAuth2: Limitations with Application Authentication

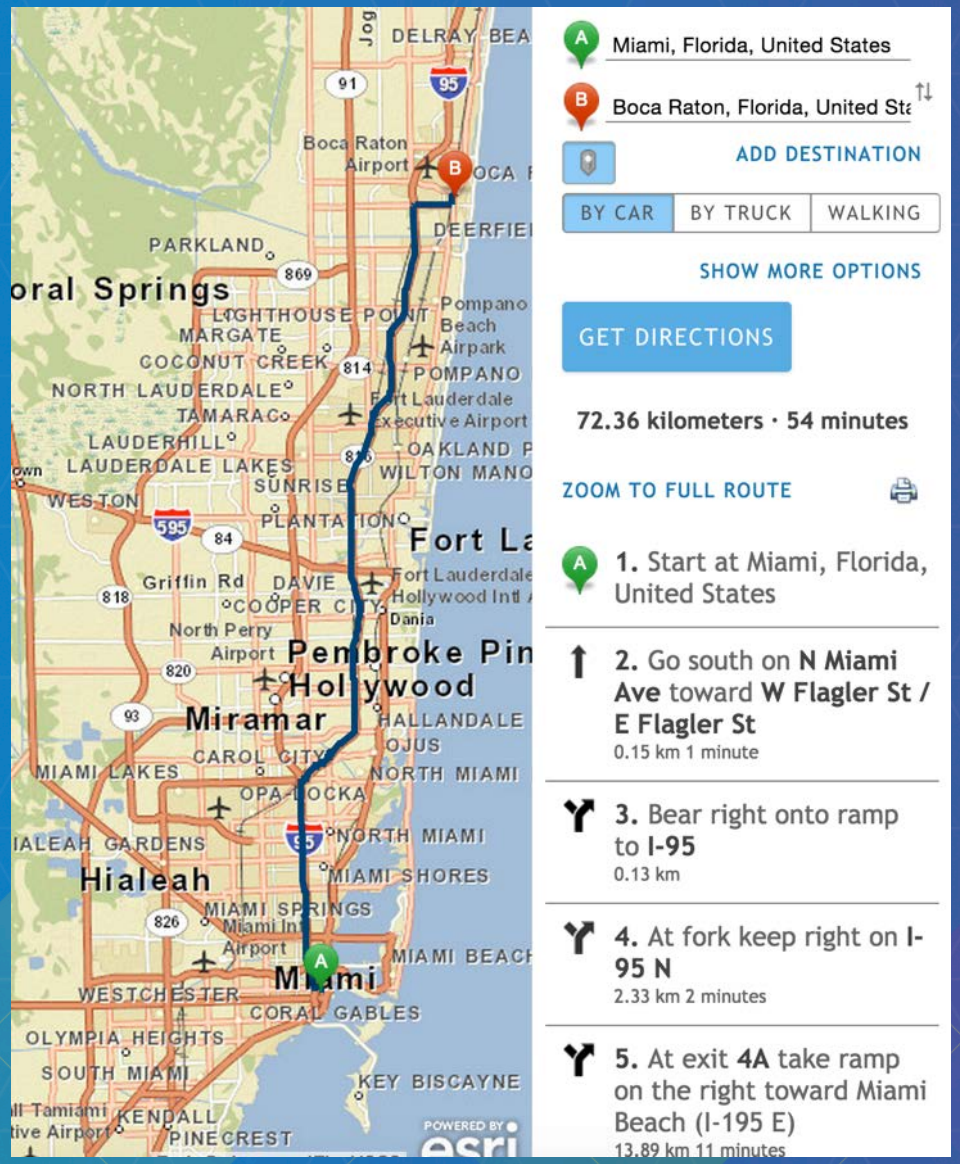
- Applications can only read or query private data content.
  - Cannot modify, upload, create, or delete content.
- Content Restrictions
  - Members of an AGO org can only access private content owned by the person who created the application.
- Marketplace restrictions
  - Cannot list applications in ArcGIS Marketplace. In order to do so, must work with user logins.

App Logins – Proxy Use Case Steps

- Register app
- Configure proxy with app credentials
- Proxy uses app credentials to get app token from the portal
- Front-end app calls into the proxy
- Proxy uses app token to call into secured portal resources
- Proxy returns results to the app

Sample Proxy configuration file

```
<ProxyConfig
    allowedReferers="*"
    logFile="proxy_log_xml.log"
    mustMatch="true">
  <serverUrls>
   <serverUrl
      url="https://route.arcgis.com"
      oauth2Endpoint="https://www.arcgis.com/sharing/oauth2"
      clientId="i3GEwDN2AL0JLc6S"
      clientSecret="Put the secret in here"
      rateLimit="120"
      rateLimitPeriod="60"
      matchAll="true"/>
   <serverUrl
      url="http://sampleserver6.arcgisonline.com"
      username="user1"
      password="Put the password here"
      rateLimit="120"
      rateLimitPeriod="60"
      matchAll="true"/>
  </serverUrls>
</ProxyConfig>
```



Demo: OAuth App login

OAuth2: Security with application authentication

- Never expose client_secret
- Keep secure server-side in proxy
  - Rate limiting against server-based misuse

Updated proxies

- https://github.com/Esri/resource-proxy
  - DotNet, JSP, and PHP
- Access resources secured with token-based authentication, i.e. premium credit-based services
- Resource and referer-based rate limiting
- Access cross domain resources
- Enabled logging

Proxies

- https://github.com/Esri/resource-proxy
- Install proxy based on README.md file
- Can also use token based authentication in addition to OAuth2
- Locking it down by referrer: Only requests coming from listed referrers are proxies
- Rate limits
  - `rateLimitPeriod`: The time period (in minutes) which the rateLimit is tracked.
  - `rateLimit`: Maximum number of requests for a specific referer over the given rateLimitPeriod.

Conclusion

- Use OAuth2 for user logins
- Benefits include usage tracking, enterprise logins, etc.
- Identity Manager simplifies login workflow in client SDKs
- Use app tokens to access secured resources in certain use cases
- https://developers.arcgis.com/authentication/ for additional information.

More information

- developers.arcgis.com/authentication
- Working with Secure Resources (JS doc)
- Using the proxy page (JS doc)
- ArcGIS Online set up enterprise logins