Agenda

• What are webhooks?
  - Advantages of implementing
• Key terms
• Demo
• Additional use cases
• Administration Tips
• Resources
Introducing webhooks
Automate notifications based on portal actions

An action occurs.
- Something is published
- A user creates a new account
- An item is deleted

The webhook is triggered.
- Delivers JSON description of event
- Relevant info fields

Your platform of choice kicks in.
- IFTTT
- Integromat
- Microsoft Flow

It performs an action.
- Emails you
- Posts in Slack
- Alexa announces it at the dinner table (maybe not)

Example JSON:
```
{
  "info": {
    "webhookName": "Group monitoring",
    "webhookId": "72f0d26eb74c9c8a22a9c6731a",
    "portalURL": "https://example.com/portal",
    "when": "2023-04-15T12:00:00Z"
  },
  "events": [
    {
      "username": "administrator",
      "userId": "173d0d46b6134dbf90c5080aad8b6208",
      "when": "2023-04-15T12:00:00Z",
      "operation": "update",
      "source": "group",
      "id": "173d0d46b6134dbf90c5080aad8b6208",
      "properties": {}
    }
  ]
}``
What is a webhook?

• A webhook is a new ArcGIS Enterprise capability that will automatically provides other applications with event-driven information, delivered as an HTTPS request (POST)

• They can be used to create automated and integrative workflows, adding new extensibility to ArcGIS Enterprise.
Advantages of using webhooks

• More efficient than polling

Polling:

Webhook:

• Allow users to create custom event-driven workflows, that can be integrated across multiple systems.
• Creates new and more efficient opportunities for automation.
Scope

- Webhooks can only be created, updated, and deleted by portal admins via the ArcGIS Portal directory (Sharing API)
  - Registration and management of webhooks are also supported via the ArcGIS API for Python.

- Webhooks can be created to subscribe to events pertaining to items, users and groups in Portal for ArcGIS.
  - For 10.7, these events are primarily *create, update, and delete* –type events.
  - The REST API doc was updated to document this resource
How to: create webhooks

1. Determine which client to use
2. Build webhook sequence
3. Turn sequence on

- Set up webhook in Portal API
- Include variables from webhook JSON message
- Receive messages each time a user is added
3rd Party Workflow Automation Software

- Visually create workflows that integrate ArcGIS Enterprise with other apps.
- Leverage the out of the box integrations for:
Microsoft Flow is part of Office 365
Automating e-mail notifications

A quick demonstration with ArcGIS Enterprise and Microsoft Flow
Webhooks
3 example use cases

- `/groups`: When someone shares an item to a collaboration group, notify group members through Slack.

- `/items`: If the sharing settings are changed for a layer that contains confidential information, email administrators.

- `/users`: Once a user is deleted, notify your admin via text.
Key Terms: Trigger event

- This is the operation you set to trigger your webhook. For example, you can configure your webhook to be triggered when a specific item is updated in your organization, or when an item is shared. A webhook can have more than one trigger event.

<table>
<thead>
<tr>
<th>Trigger event</th>
<th>URI example</th>
</tr>
</thead>
<tbody>
<tr>
<td>All trigger events for all items</td>
<td>/items</td>
</tr>
<tr>
<td>Add item to the portal</td>
<td>/items/add</td>
</tr>
<tr>
<td>All trigger events for a specific item</td>
<td>/items/&lt;itemID&gt;</td>
</tr>
<tr>
<td>Delete a specific item</td>
<td>/items/&lt;itemID&gt;/delete</td>
</tr>
<tr>
<td>Update a specific item's properties</td>
<td>/items/&lt;itemID&gt;/update</td>
</tr>
<tr>
<td>Move an item or changing ownership of the item</td>
<td>/items/&lt;itemID&gt;/move</td>
</tr>
<tr>
<td>Publish a specific item</td>
<td>/items/&lt;itemID&gt;/publish</td>
</tr>
<tr>
<td>Share a specific item</td>
<td>/items/&lt;itemID&gt;/share</td>
</tr>
<tr>
<td>Unshare a specific item</td>
<td>/items/&lt;itemID&gt;/unshare</td>
</tr>
</tbody>
</table>
# Key Terms: Trigger event

## Trigger events for Portal users:

<table>
<thead>
<tr>
<th>Trigger event</th>
<th>URI example</th>
</tr>
</thead>
<tbody>
<tr>
<td>All trigger events for all users in the portal</td>
<td>/users</td>
</tr>
<tr>
<td>Add user to the org</td>
<td>/users/add</td>
</tr>
<tr>
<td>All trigger events associated with a specific user</td>
<td>/users/&lt;username&gt;</td>
</tr>
<tr>
<td>Delete a specific user</td>
<td>/users/&lt;username&gt;/delete</td>
</tr>
<tr>
<td>Update a specific user's profile</td>
<td>/users/&lt;username&gt;/update</td>
</tr>
<tr>
<td>Disable a specific user's account</td>
<td>/users/&lt;username&gt;/disable</td>
</tr>
<tr>
<td>Enable a specific user's account</td>
<td>/users/&lt;username&gt;/enable</td>
</tr>
<tr>
<td>Specific user signs in</td>
<td>/users/&lt;username&gt;/signIn</td>
</tr>
<tr>
<td>Specific user signs out</td>
<td>/users/&lt;username&gt;/signOut</td>
</tr>
</tbody>
</table>

## Trigger events for Portal groups:

<table>
<thead>
<tr>
<th>Trigger event</th>
<th>URI example</th>
</tr>
</thead>
<tbody>
<tr>
<td>All trigger events for all groups</td>
<td>/groups</td>
</tr>
<tr>
<td>Add group</td>
<td>/groups/add</td>
</tr>
<tr>
<td>All trigger events for a specific group</td>
<td>/groups/&lt;groupID&gt;</td>
</tr>
<tr>
<td>Update a specific group</td>
<td>/groups/&lt;groupID&gt;/update</td>
</tr>
<tr>
<td>Delete a specific group</td>
<td>/groups/&lt;groupID&gt;/delete</td>
</tr>
<tr>
<td>Enable <strong>Delete Protection</strong> for a specific group</td>
<td>/groups/&lt;groupID&gt;/protect</td>
</tr>
<tr>
<td>Disable <strong>Delete Protection</strong> for a specific group</td>
<td>/groups/&lt;groupID&gt;/unprotect</td>
</tr>
<tr>
<td>Invite a user to a specific group</td>
<td>/groups/&lt;groupID&gt;/invite</td>
</tr>
<tr>
<td>Add a user to a specific group</td>
<td>/groups/&lt;groupID&gt;/addUsers</td>
</tr>
</tbody>
</table>
Key Terms: Payload

• Once a webhook is triggered, a payload is delivered to the specified payload URL in JSON format. Each event follows a similar JSON schema with information that is relevant to the event.

```json
{
  "info" : {
    "webhookId" : "b1280bce338e4a96a981b09c50ee409c",
    "webhookName" : "Myhook",
    "portalURL" : "https://dev004029.esri.com/portal/",
    "when" : 1547056298561
  },
  "events" : [{
    "userId" : "4ee8b3d66a2e4da3a56f8d256f1df7ce",
    "username" : "admin",
    "when" : 1547056298554,
    "operation" : "share",
    "source" : "item",
    "id" : "186bad9e3dd847ab9a601b505879f86d",
    "properties" : {
      "sharedToGroups" : ["Everyone", "4adc30bb03054812a846fa592de105de"]
    }
  }]
}
```
{
    "info": {
        "webhookId": "b1280bce338e4a96a981b09c50ee409c",
        "webhookName": "Myhook",
        "portalURL": "https://FQDNm/portal/",
        "when": 1547056298561
    },
    "events": [{
        "userId": "4ee8b3d66a2e4da3a56f8d256f1d7f7ce",
        "username": "admin",
        "when": 1547056298554,
        "operation": "share",
        "source": "item",
        "id": "186bad9e3dd847ab9a601b505879f86d",
        "properties": {
            "sharedToGroups": ["Everyone", "4adc30bb03054812a846fa592de105de"]
        }
    }]
}
Key Terms: Payload URL

- A payload URL must be provided when creating a webhook; this defines where the payload will be delivered.

- Since all payloads are delivered through an HTTPS POST request, the webhook receiver must be configured to communicate over HTTPS and be reachable by the Portal.

- Number of options for how and where a webhook is received:
  - Enterprise SDK sample (Java servlet),
  - Third party services: Microsoft Flow, Zapier, IFTTT, Integromat
  - Custom web servers: Node JS, Python Flask
How to manage webhooks?

- We’ve seen the Web Interface
How to manage webhooks?

- Every action is available as **REST API** calls
  - So you can manage webhooks deployment using scripts

---

### JSON response example

```
{
  "webhooks": [
    {
      "id": "70c0f6e244d38d3b080a563e04143",
      "accountID": "8245098584989207",
      "payloadURL": "https://example.com:8080",
      "secret": "",
      "active": true,
      "status": "all_updated",
      "configurationPolicy": {
        "maxNumberOfFailures": 5,
        "dayOfWeek": 5
      }
    },
    {
      "id": "60cfe226e209be5d8e80a563e04143",
      "created": 1544023170018,
      "modified": 1544023170018,
      "events": [
        "/"
      ]
    }
  ]
}
```
How to manage webhooks?

- Also possible to use the ArcGIS API for Python
  - The library to manage your geospatial platform
- By using the **Webhook manager** and **Webhook class**

```python
In [11]: from arcgis.gis import GIS
In [12]: gis = GIS("https://cedessltest.westeurope.cloudapp.azure.com/portal","myuser","mysecretpass")
In [21]: whManager = gis.admin.webhooks

In [22]: whManager.list()
Out[22]: [WebHook @ all items>, WebHook @ allItems-otherone>, WebHook @ allItems-new>, WebHook @ allItems>, WebHook @ allItems-other>

In [15]: whManager.list()[0].properties
Out[15]:
```

```
In [20]: for webhook in whManager.list():
   # webhook.deactivate()
   webhook.activate()
```
Okay so what can I do with this?
Get new insights into your GIS usage

• By defining a webhook on all events
  - Who
  - What
  - When

• A lot of external connectors to save this info
  - Push into Excel, Google sheet, …
  - Push into SQL Server, Oracle, PostgreSQL, …
  - Push into a PowerBI Datasource

• With data, it’s not complex to create a usage dashboard
  - Check out the Blog Post: How we did it: Webhooks at Dev Summit 2019
Additional use cases

• **Group-based workflows**
  - Trigger webhook when an item has been shared to a specific group, or if an unauthorized user has been added to a secured group.

• **Processing new items added to ArcGIS Enterprise portal**
  - When a new item has been added generate thumbnails, update the metadata, share to groups of users etc.

New item is added → Webhook is triggered and payload is sent out to payload URL → Payload received and processed by Microsoft Flow → Use Python API for ArcGIS to generate thumbnails, update the metadata etc. → New item is updated with thumbnails, metadata etc.
Manage more complex workflows

• Manage the process of portal item publication from staging to production

• Using ArcGIS Enterprise, Microsoft Flow, Azure DevOps Work items, Teams, and Azure serverless functions
  - Step 1: When a user shares an item to the “review” group, a work item is created
  - Step 2: When the review is finished
    - If ok: the item is cloned to the production environment using Azure serverless Python functions
    - In not ok: We nicely inform the user about the correct publishing process
Workflow - Step 1:

- Share an item/map to the « Review_group »
- Staging ArcGIS Enterprise
- Microsoft Flow
- Create a task/work item in Azure DevOps
- Notify the reviewers in Teams
Workflow - Step 2:

- Review the map
- Update the Work item status
- Execute a Python script in Azure function
- Deep copy the content
- Update sharing properties

Production ArcGIS Enterprise → Staging ArcGIS Enterprise
You also have webhooks available with Survey123
Administration tips

• Avoid redundancy
• Security
• Troubleshooting
  - Portal logs
  - Notification status
  - Deactivate
  - Fiddler
• Advanced settings
  - Number of delivery attempts
  - Notification timeout
  - Elapsed time between delivery attempts
What’s next

• Support for additional Item, User and Group events
• Support for Feature layer editing events
• Integration with webhook services such as Microsoft Flow, Integromat, Zapier etc. in ArcGIS Enterprise
• UI and UX within ArcGIS Enterprise portal
Resources

- GeoDev Webinar

- GitHub repository:
  - https://github.com/Esri/webhooks-samples

- ArcGIS Enterprise Overview:

- Documentation:
  - https://developers.arcgis.com/rest/users-groups-and-items/webhooks.htm

- Blogs:
Questions ?
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the feedback section

Complete answers and select “Submit”