

Now We're Talking... Two-way Communication with Alma Using Webhooks

ELUNA Developers Day 2017 | Schaumburg, IL
Josh Weisman | VP Development, Resources Management

Agenda

1

Introduction to Webhooks

2

Anatomy of a Webhook Listener

3

Building Your First Webhook Listener

4

In the Wild: Hosting a Webhook Listener in the Public Cloud

5

Wrap Up

Introduction to Webhooks

What are Webhooks?

Webhooks are "user-defined HTTP callbacks".^[2] They are usually triggered by some event, such as pushing code to a repository^[3] or a comment being posted to a blog.^[4] When that event occurs, the source site makes an HTTP request to the URI configured for the webhook. Users can configure them to cause events on one site to invoke behaviour on another.

Wikipedia

<https://en.wikipedia.org/wiki/Webhook>

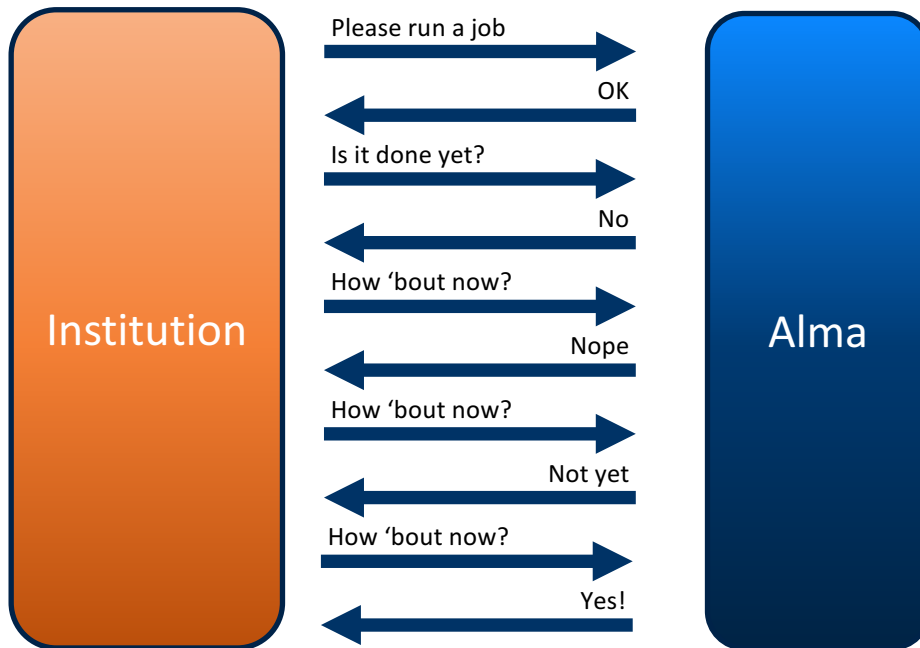
Webhook Benefits

- Alternative to “polling”- push rather than pull
- Respond to events in the system when they occur
- Asynchronous architecture
- Reduce API calls

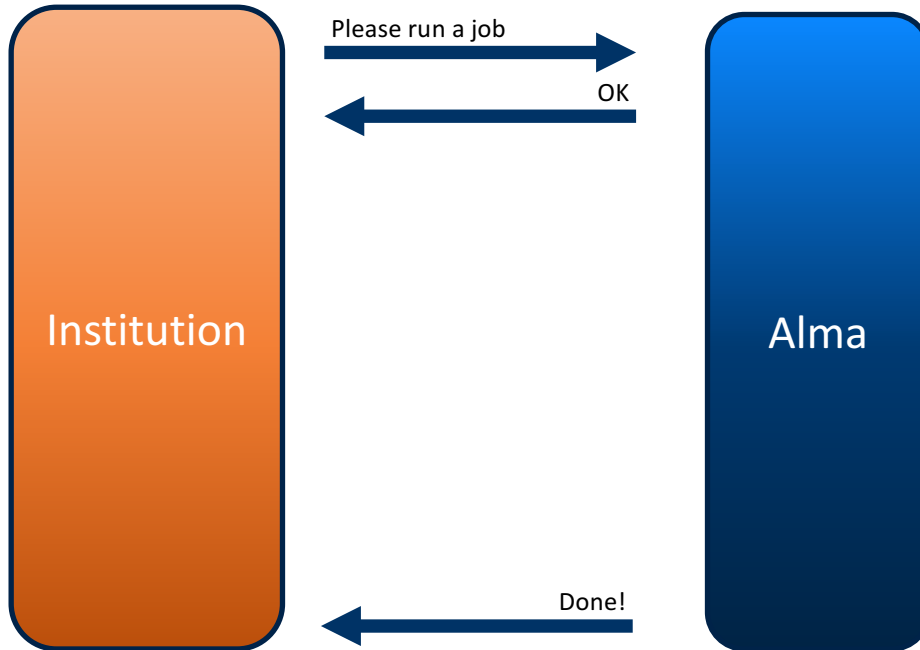


Source: <http://www.webhooks.org>

Polling- “Pull”



Webhooks- “Push”



Webhooks in Alma

- Support for the following events:
 - Job end- Scheduled or manual jobs
 - Notifications
 - User update (from June 2017)
- Additional events coming- loans and requests in the mid term roadmap
- Configure with a webhook integration profile

Anatomy of a Webhook

Webhook Listener - GET

- Alma “challenges” the webhook listener when the integration profile is activated
- This ensures an active listener is available at the provided URI
- The listener should reply to the GET request with the challenge sent in the query string

```
GET  
/webhook?challenge=1234
```

```
200 OK  
{“challenge”:“1234”}
```

Webhook Listener - POST

- Alma sends a POST request to the listener each time the specified event occurs
- The listener processes the request and returns response to Alma

```
POST/webhook
X-Exl-Signature: g4njfSmb5BqPPH=

{"id":"1767426760000561","action":"job_end",
"time":"2016-09-07T08:48:45Z",
"job_instance":{"id":"1767426760000561",
"name":"Add set 1767425500000561 to collections:
8182119880000561.."}
...
```

```
200 OK
```

Webhook Listener - POST

POST /webhook

X-Exl-Signature: 0gFT8DgzoOCNyApuMvlz62bz9KIuhvr+r14nQw3rFJ0=

```
{  
  "id": "1767426760000561",  
  "action": "job_end",  
  "time": "2016-09-07T08:48:45Z",
```

Base64-Encoded
HMAC SHA256
Signature

Webhook details

```
  "job_instance": {  
    "id": "1767426760000561",  
    "name": "Add set 1767425500000561 to collections: 8182..",  
    "submit_time": "2016-09-07T08:48:14Z",  
    "start_time": "2016-09-07T08:48:36Z",  
    "end_time": "2016-09-07T08:48:45Z",  
    "progress": 100,  
    "status": {  
      "value": "COMPLETED_SUCCESS",  
      "desc": "Completed Successfully"  
    },  
  },  
}
```

Action-specific
payload (eg. Job
Instance)
JSON or XML

Signature

- Ensures message came from Alma and that it was not tampered with
- Listener can validate the signature (recommended, but not required)
- Base64 encoded HMAC SHA256

```
function validateSignature(body, secret, signature) {  
  var hash = crypto.createHmac('SHA256', secret)  
    .update(JSON.stringify(body))  
    .digest('base64');  
  return (hash === signature);  
}
```

Building your first Webhook listener




First Webhook Listener

- Node.js Application
- Handles challenge GET
- Receives incoming POST requests
 - Validates signature
 - Ready to call method based on event type/action



DEMO:

Building Your First Webhook Listener



In the Wild: Hosting a Webhook Listener in the Public Cloud

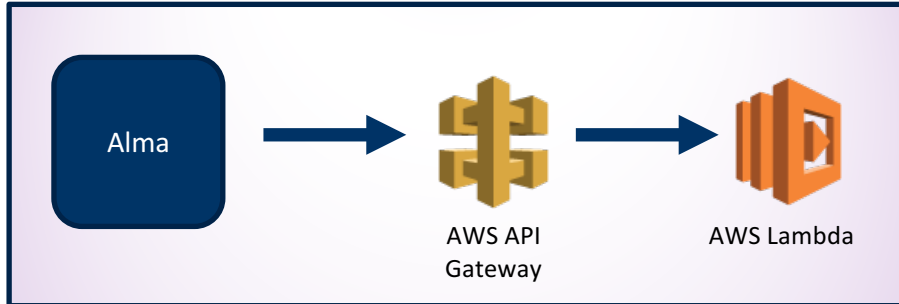


Webhook Listeners in the Cloud

- Webhook listeners require a certain amount of infrastructure
- Prefer not to host the infrastructure locally
- Public cloud infrastructure allows hosting of webhook listeners for a very low cost (“almost free”)

Public Cloud Webhook Listener: AWS

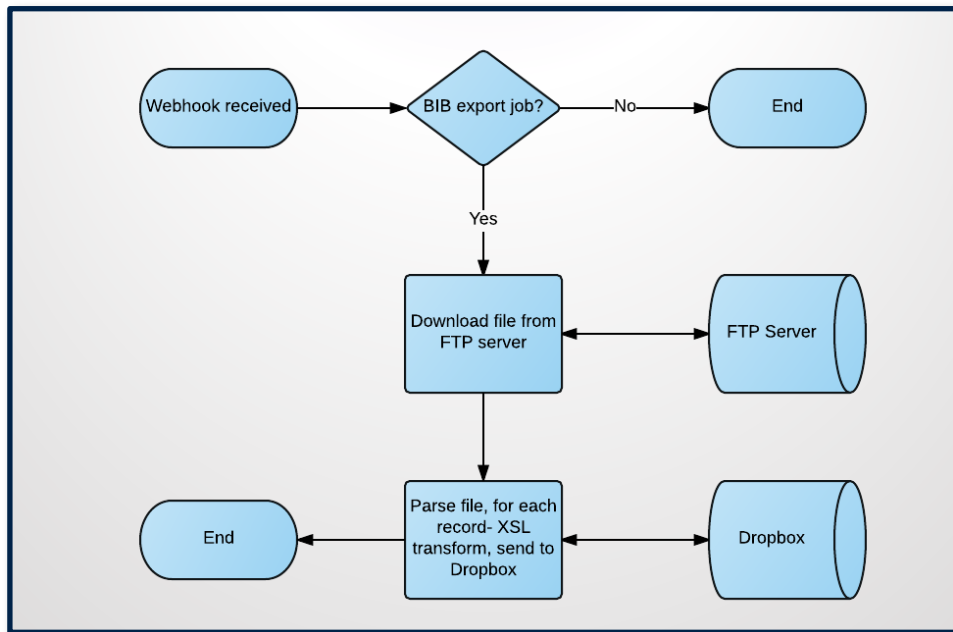
- Lambda Microservice in Node.js
- API Gateway
- Demo: Send SMS to job initiator when job is complete



Public Cloud Webhook Listener: Azure

- ASP.NET 4.6 MVC Web API application in C#
- Deployed as Web App on Azure
- Demo: Download exported file, parse, transform each record, upload to Dropbox
 - Represents use case where records exported from Alma need to be sent to another system

Public Cloud Webhook Listener: Azure

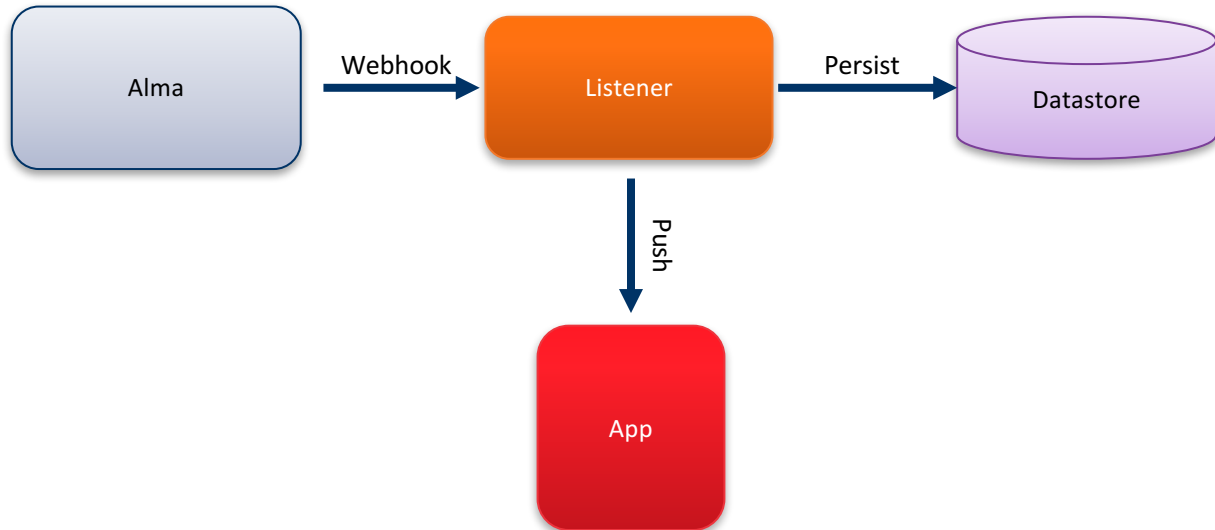


Notification Use Case

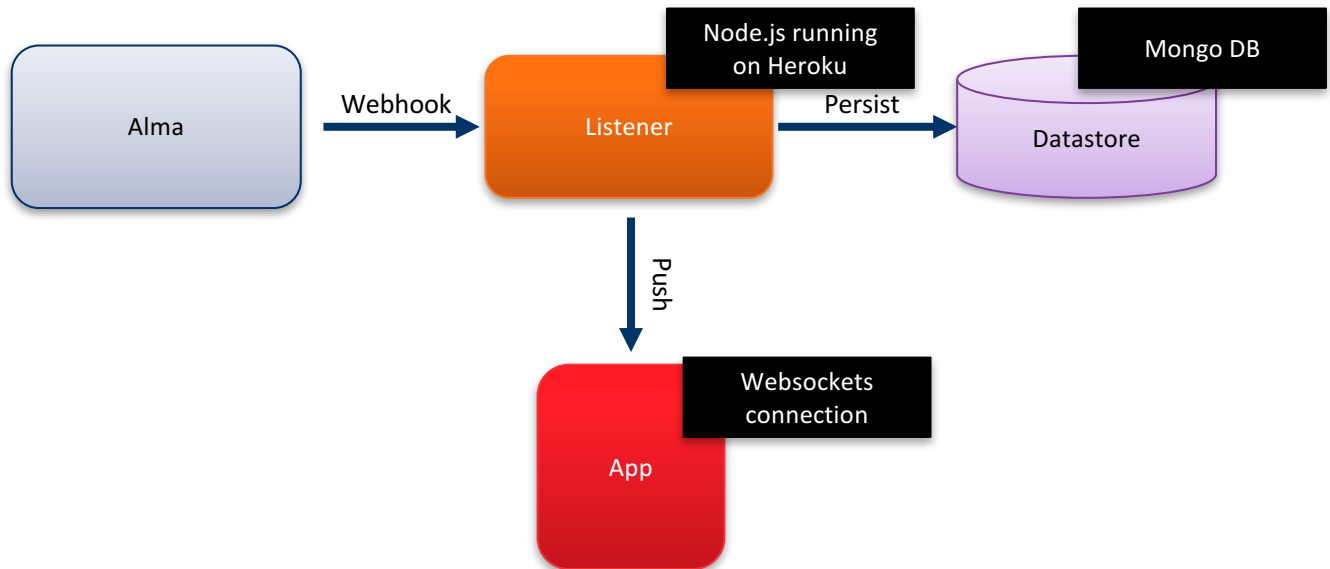
- Enables real time notifications when relevant events happen
 - i.e. requested item arrives, request cancelled, due date changed
- Common use case- mobile application



Elements of a Notification Flow



Elements of a Notification Flow - Example





DEMO:

Hosting a Webhook Listener in the Public Cloud

Wrapping Up



Webhook Tips

- Listener Error Codes
 - 200 OK
 - 400 Logged as error
 - 500 Retry 3 times each hour
- Short timeout
 - Use asynchronous patterns for long running tasks

References

- Webhooks [documentation](#) on the Developer Network
- Webhooks [blog posts](#)

Wrap Up

- Start thinking about how the webhook model can change your integrations with Alma
- Add your event type requests to the Idea Exchange
- Ask questions on the Developer Network Forum

Have fun!

THANK YOU

josh.weisman@exlibrisgroup.com