If This Then That Automation with Webhooks

Shakopee
PUBLIC SCHOOLS
EDUCATION FORWARD

Nathaniel Strauss
March 3rd, 2020
BrainStorm K20 Wisconsin Dells

Agenda

- Why do we automate?
- What are webhooks?
- Zapier and webhooks
- Rolling your own webhook receiver
- How we use webhooks internally
- Brainstorm time

Resources to follow along

http://bit.ly/brainstorm-webhooks

Intro

whoami

Nathaniel Strauss
IT Manager

Been with the district about 4 years.
Primarily responsible for 1:1 program.

Shakopee Public Schools

- ~8000 students
- 1:1 iPad K-8
- 1:1 MacBook 9-12
- Certified staff have Macs
- PCs
- Chromebooks

Why automate?

- Manual processes and data entry are the enemy.
- No matter how good a human is at data entry, a computer will always be better and faster.
- Bad data in means bad data out.
- Making decisions based on bad data leads to poor outcomes.

What’s already automated?

- SIS - student enrollment, class rostering, reports
- LMS - Syncing classes with SIS, grade passback
- Account management - SIS/HR system to LDAP, AD/LDAP to external directories, SAML/OAuth authentication
- Wireless authentication - 802.1x certificate generation and/or NAC
- Device management - SCCM/Config Mgr, Google admin console, Apple MDM

Maybe parts of those processes are automated, but still require manual work. What steps can be picked out and made better with automation?

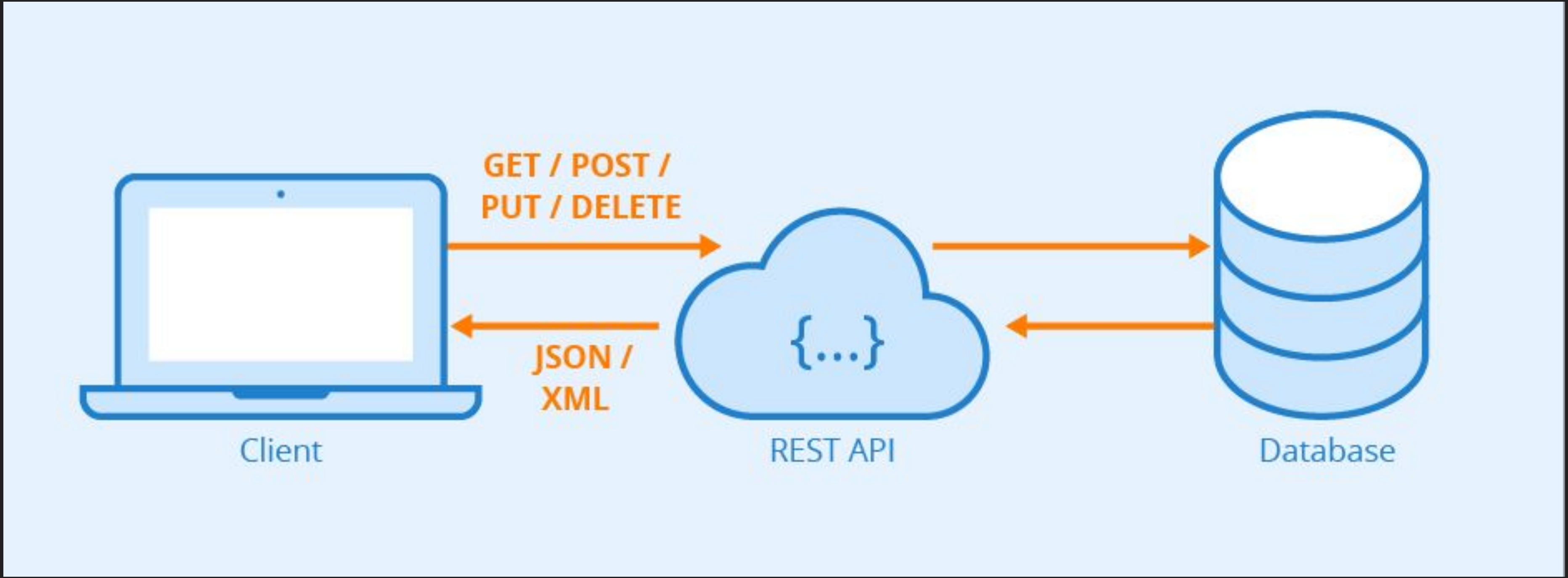
What is an API call?

A REST API call is a way to programmatically interact with a web service.

Client
Requests the data or action from the web application (resource) endpoint.

- Read actions - GET
- Write actions - PUT (update), POST (create), DELETE

Resource
The web service itself, typically backed by a database.



REST API basics

API Example - iTunes API

Let’s get data about the Google Drive iOS app from Apple’s iTunes API.

Refer to snippets.md “Example App Store API GET”

Mac/Linux

```
curl "https://itunes.apple.com/us/lookup?id=507874739" | python -m json.tool
```

Windows

```
(Invoke-WebRequest -Uri
"https://itunes.apple.com/us/lookup?id=507874739").Content | ConvertFrom-Json
| ConvertTo-Json
```

JSON

API and webhook data comes in many formats. JSON and XML are the most popular.

JSON (**J**ava**S**cript **O**bject **N**otation) is organized by data based on key-value pairs (objects) and ordered lists (arrays).

```
{
  "id": 4911,
  "firstName":"Nathaniel",
  "lastName":"Strauss",
  "isStudent": true
};
```

Reading JSON Values

Most languages come with built-in tools to read JSON. These one liners download raw JSON from the iTunes API and print the app version key value.

Note the API returns JSON in a nested list. Most of the data lives under the first index of that list. That's why `[“results”][0]` or `select -expand results` is used

Mac/Linux

```
curl "https://itunes.apple.com/us/lookup?id=507874739" | python -c 'import
sys, json; print json.load(sys.stdin)["results"][0]["version"]'
```

Windows

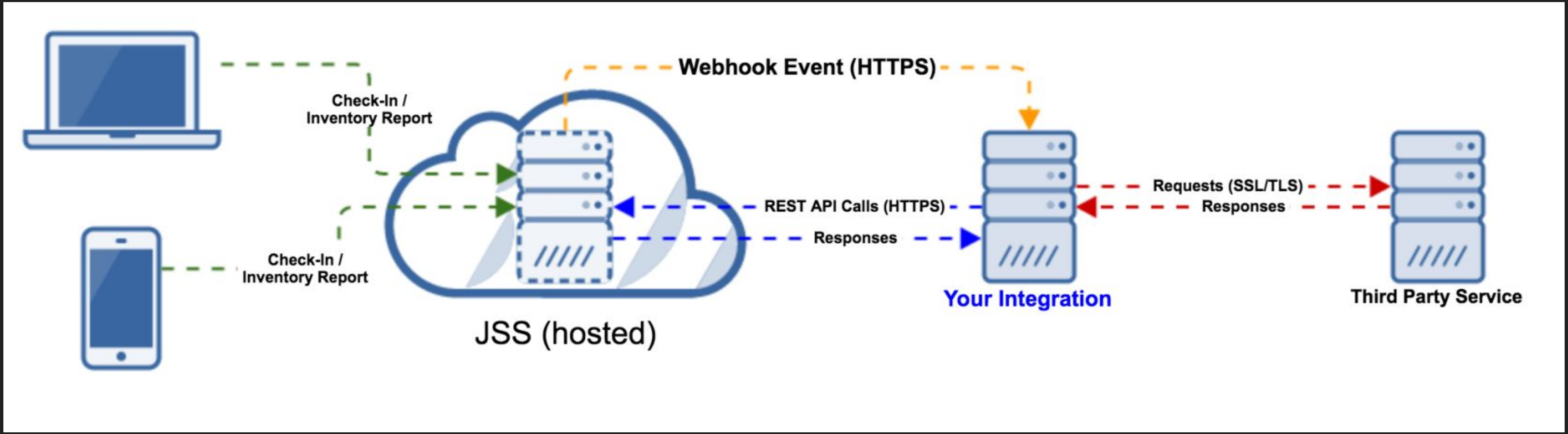
```
(Invoke-WebRequest -Uri
"https://itunes.apple.com/us/lookup?id=507874739").Content | ConvertFrom-Json
| select -expand results | Select-Object -ExpandProperty version
```

What is a webhook?

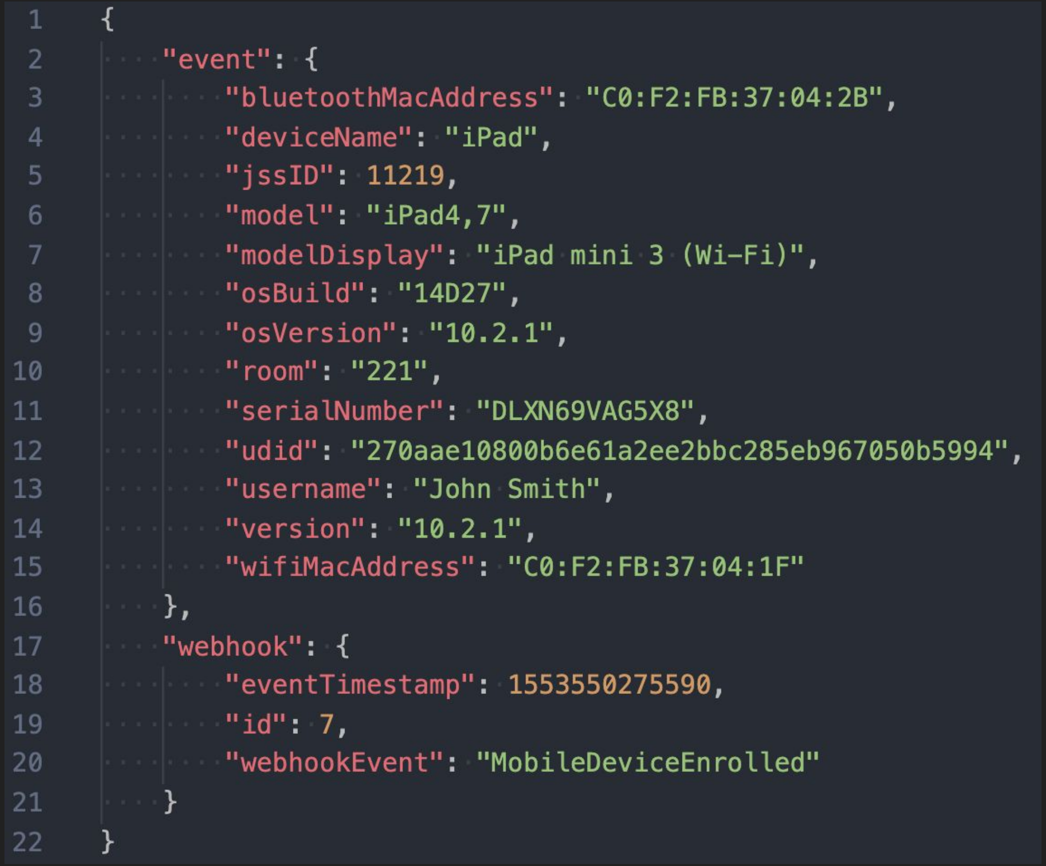
A webhook is an HTTP request triggered by an application event.

It can be thought of as a reverse API call. Instead of polling an API for data, webhooks send data when an event occurs.

1. An event occurs.
2. Application sends a webhook with data about that event.
3. The receiving endpoint sends an acknowledgement (callback).
4. Receiving server then does ***something*** with the data.



Webhook components



```json
{
    "event": {
        "bluetoothMacAddress": "C0:F2:FB:37:04:2B",
        "deviceName": "iPad",
        "jssID": 11219,
        "model": "iPad4,7",
        "modelDisplay": "iPad mini 3 (Wi-Fi)",
        "osBuild": "14D27",
        "osVersion": "10.2.1",
        "room": "221",
        "serialNumber": "DLXN69VAG5X8",
        "udid": "270aae10800b6e61a2ee2bbc285eb967050b5994",
        "username": "John Smith",
        "version": "10.2.1",
        "wifiMacAddress": "C0:F2:FB:37:04:1F"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "webhookEvent": "MobileDeviceEnrolled"
    }
}
```

Example webhook JSON - /webhook_examples/hook_example1.json

Webhook vs API GET

The advantage of webhooks is real time event data.

With REST API we have to constantly poll the application to find out if data has changed. Webhooks trigger only when an event occurs.

For example, I want to know when an iPad checks into my MDM.

API - Multiple Events	**Webhook - One Event**
Send API GET requests until the data changes. Over and over and over and over...	Webhook is sent to endpoint when iPad check in event occurs. Server endpoint then handles data.

What can a webhook do?

Webhooks are good at...

- Automating event driven tasks.
- Being the “glue” between applications.
- Chaining together multiple tasks.
  - Event 1 > system 1 > event 2 > system 2

Zapier

www.zapier.com

zapier.com/page/webhooks/

Zapier Example 1 - Webhook to Google Sheets

Add info to a Google Sheet from new Webhook POST requests

Goal: Send a webhook to a URL to create new rows in a Google Sheet.

Webhook event (us) → Zapier (app) → Action (Sheet)

1. Create a new Google Sheet and name it.
2. Start a new Zap.
3. Configure the webhook and get custom URL.
4. Send a test webhook as sample data.
5. Set up Google Sheet as a target for webhook.
6. Map spreadsheet columns to webhook keys.

Zapier Example 1 - Webhook to Google Sheets

7. Send webhooks to write rows to the spreadsheet.

Mac/Linux

```
curl https://hooks.zapier.com/hooks/catch/12345/abcd/ -X POST -d
@/Users/nstrauss/github/brainstorm-webhooks/webhook_examples/hook_example1.json

# Let's send a few at a time to simulate a real application
./simple_loop.sh
```

Windows

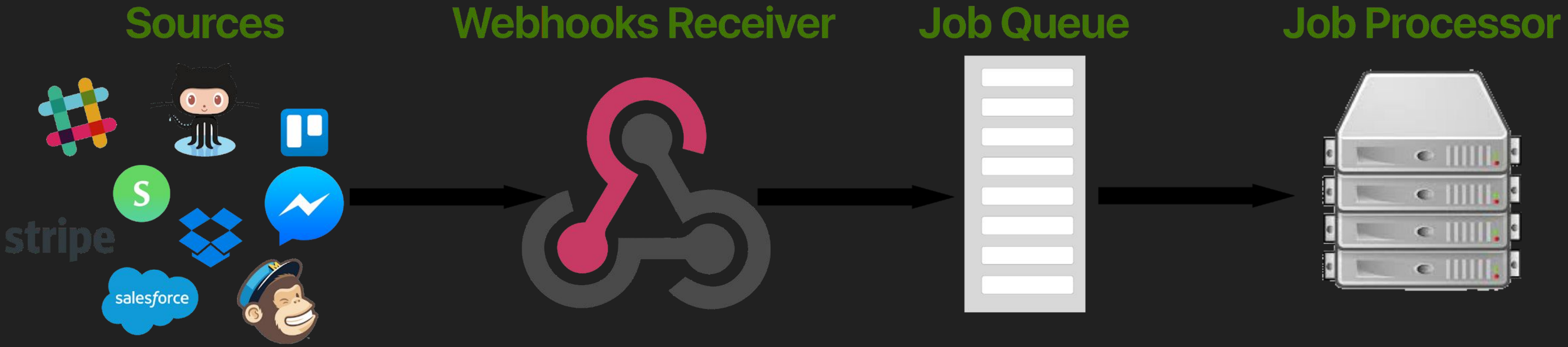
```
(Invoke-WebRequest -Uri
"https://itunes.apple.com/us/lookup?id=507874739").Content | ConvertFrom-Json |
ConvertTo-Json
```

Zapier Example 2 - Google Forms to Docs

Goal: Create a Google Doc based on data submitted from a Google Form.

Fill out the form!
http://bit.ly/brainstorm-webhooks-form

Zapier Example 2 - Google Forms to Docs



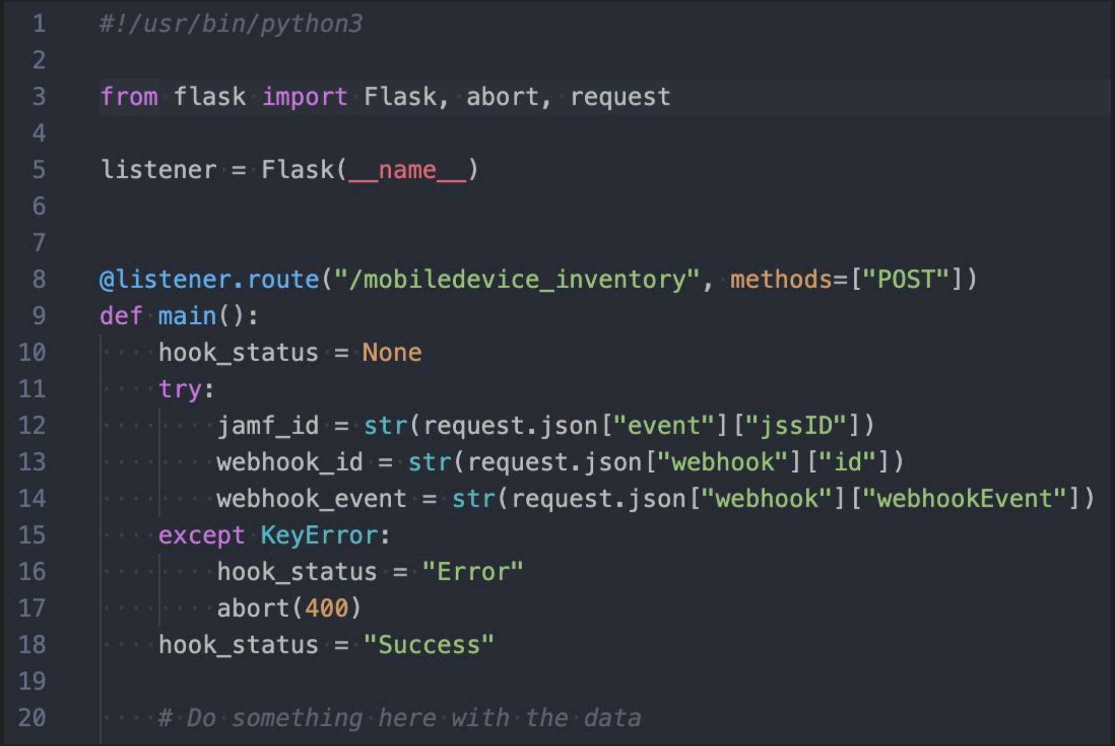
Webhooks Job Queue

Zapier Example 2 - Google Forms to Docs

Now let’s take a look at the Google Doc. This example works by chaining webhooks together.

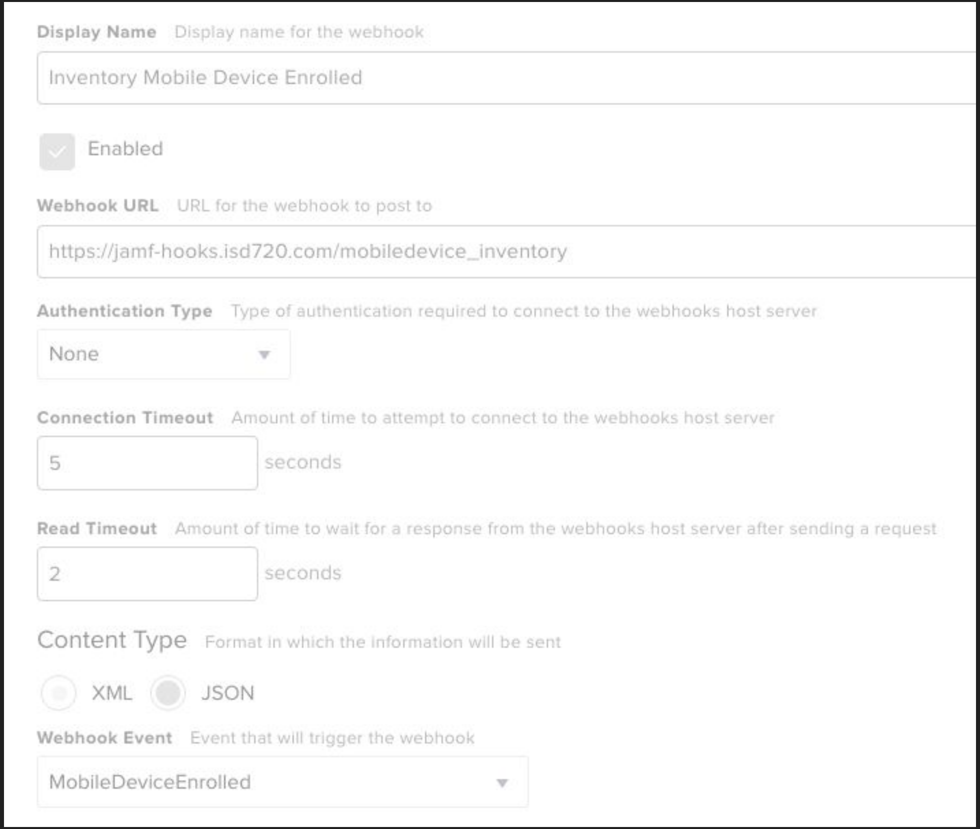
Form submitted → webhook POST → webhook received → Google Doc updated

Roll Your Own Webhook Receiver



```python
#!/usr/bin/python3

from flask import Flask, abort, request

listener = Flask(__name__)


@listener.route("/mobiledevice_inventory", methods=["POST"])
def main():
    hook_status = None
    try:
        jamf_id = str(request.json["event"]["jssID"])
        webhook_id = str(request.json["webhook"]["id"])
        webhook_event = str(request.json["webhook"]["webhookEvent"])
    except KeyError:
        hook_status = "Error"
        abort(400)
    hook_status = "Success"

    # Do something here with the data
```

- Python Flask based webhook receiver
- /mobiledevice_inventory endpoint URL
- Jamf Pro sends the webhook event and the receiving server acts on it
- hook_receiver_example.py


Display Name Display name for the webhook

Inventory Mobile Device Enrolled

[✓] Enabled

Webhook URL URL for the webhook to post to

https://jamf-hooks.isd720.com/mobiledevice_inventory

Authentication Type Type of authentication required to connect to the webhooks host server

None

Connection Timeout Amount of time to attempt to connect to the webhooks host server

5 seconds

Read Timeout Amount of time to wait for a response from the webhooks host server after sending a request

2 seconds

Content Type Format in which the information will be sent

XML [✓ JSON]

Webhook Event Event that will trigger the webhook

MobileDeviceEnrolled

Jamf Pro webhook settings

Demo - iPad Enrolled

- iPad name changed to student name
- Inventory status set to Deployed
- User and location info set from LDAP (AD)

Automated, reliable, no burden of knowledge.

Demo - iPad Wiped

- Name changed to serial
- Inventory status set to In Stock
- User and location information cleared out

K20 Products with Hooks

Common solutions with webhooks out of the box.

- Canvas
- Moodle
- Schoology
- Incident IQ
- Slack (incoming)
- Microsoft Teams (incoming)
- Jamf Pro
- Trello
- Lots more

Brainstorm

What kind of common tasks can you automate with webhooks?

Event $\rightarrow$ Response

Ticket created $\longrightarrow$ Send a Slack message

Form response submitted $\longrightarrow$ Google Doc created

New student added to a course $\longrightarrow$ Instructor notified

Start with existing processes and identify small steps that can be automated. Eventually those smaller pieces will add up.

Contact

Nathaniel Strauss
nstrauss@shakopee.k12.mn.us

https://github.com/nstrauss
https://www.linkedin.com/in/nathaniel-straus
https://twitter.com/nwstrauss_

Slack - MacAdmins (https://www.macadmins.org/) and MinnEdTech

Resources

- http://bit.ly/brainstorm-webhooks
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f
- https://canvas.instructure.com/doc/api/
- https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/
- https://codeburst.io/what-are-webhooks-b04ec2bf9ca2
- https://simonfredsted.com/1583
- https://medium.com/@jsneedles/your-webhooks-endpoint-should-do-almost-nothing-d246378a85e5
- https://bryson3gps.wordpress.com/2016/08/10/webhooks-come-to-the-jss/
- https://github.com/brysontyrrell/Example-JSS-Webhooks
- https://zapier.com/page/webhooks/