



jamf

|

NATION

User Conference

OCTOBER 24-26

2017



Bob Gandler

Apple Platform Engineer/1-to-1 Coordinator
St. Andrew's Episcopal School



API Noob to Ninja

Agenda for Today

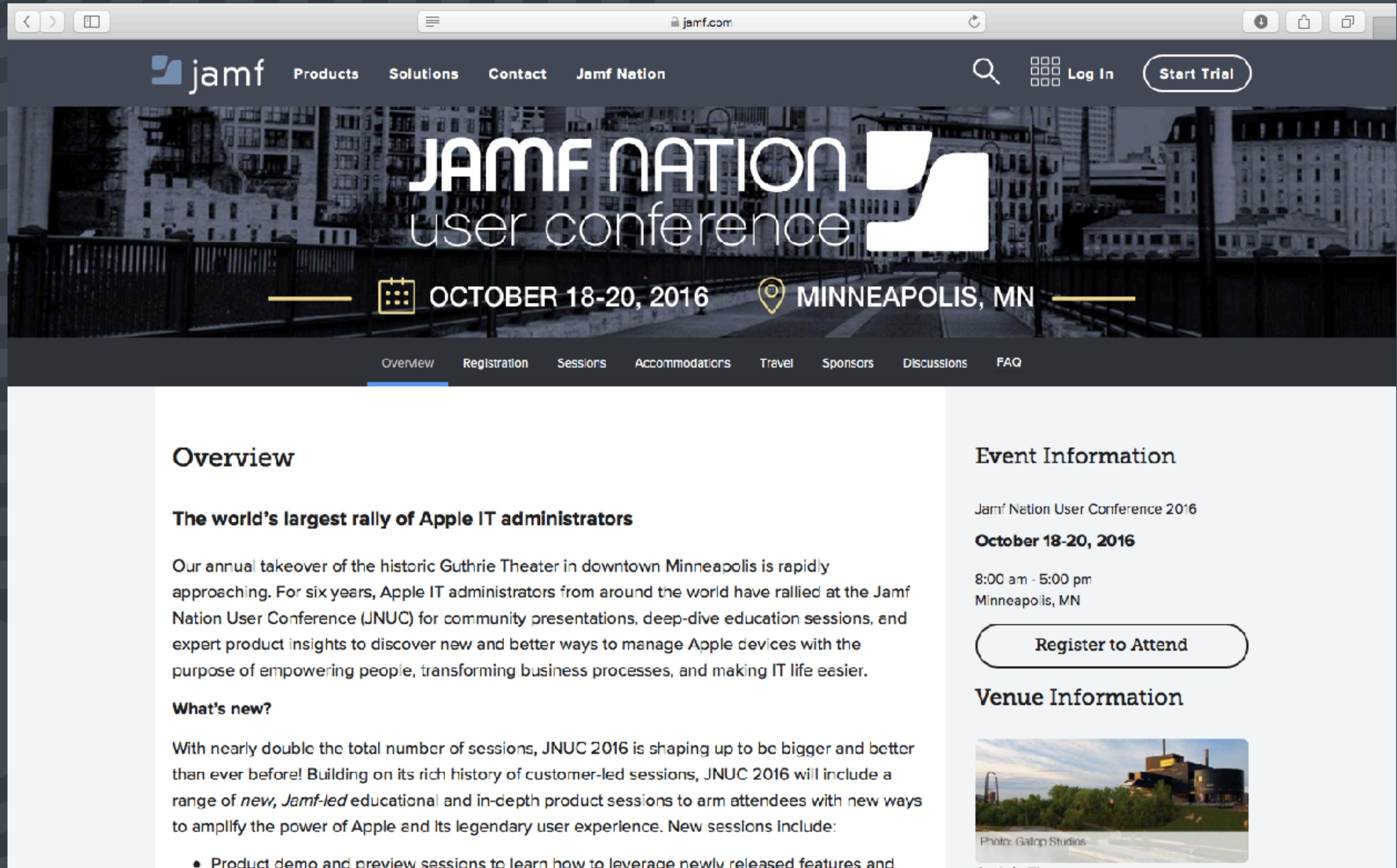
- What is an API and the Jamf API
- Learn how to use the Jamf API
- See examples on using the Jamf API
- Take over the world!



But, first story time



JNUC 2016



The screenshot shows the homepage of the JNUC 2016 website. At the top, there is a navigation bar with links for Products, Solutions, Contact, Jamf Nation, a search icon, Log In, and a Start Trial button. The main banner features a black and white photograph of a city skyline at night with the text "JAMF NATION user conference" overlaid. Below the banner, the event details are listed: "OCTOBER 18-20, 2016" and "MINNEAPOLIS, MN". A navigation menu below the banner includes Overview (which is underlined), Registration, Sessions, Accommodations, Travel, Sponsors, Discussions, and FAQ. The "Overview" section contains two subsections: "The world's largest rally of Apple IT administrators" and "What's new?". The "Event Information" section provides details about the conference: "Jamf Nation User Conference 2016", "October 18-20, 2016", "8:00 am - 5:00 pm", and "Minneapolis, MN". It also features a "Register to Attend" button. The "Venue Information" section includes a photo of the Guthrie Theater.

Overview

The world's largest rally of Apple IT administrators

Our annual takeover of the historic Guthrie Theater in downtown Minneapolis is rapidly approaching. For six years, Apple IT administrators from around the world have rallied at the Jamf Nation User Conference (JNUC) for community presentations, deep-dive education sessions, and expert product insights to discover new and better ways to manage Apple devices with the purpose of empowering people, transforming business processes, and making IT life easier.

What's new?

With nearly double the total number of sessions, JNUC 2016 is shaping up to be bigger and better than ever before! Building on its rich history of customer-led sessions, JNUC 2016 will include a range of *new, Jamf-led educational* and in-depth product sessions to arm attendees with new ways to amplify the power of Apple and its legendary user experience. New sessions include:

- Product demo and preview sessions to learn how to leverage newly released features and

Event Information

Jamf Nation User Conference 2016
October 18-20, 2016
8:00 am - 5:00 pm
Minneapolis, MN

[Register to Attend](#)

Venue Information

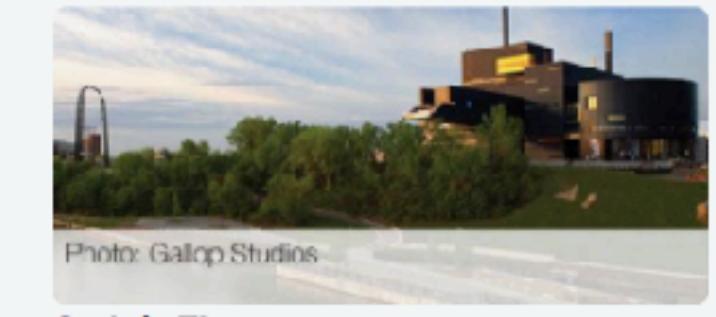


Photo: Gallop Studios
Guthrie Theater



“Take a script you already have and just figure out how to do it with the API.”

Joel Rennich - Trusource Labs



```
#!/bin/sh

#get current user info from AD
getUser=`ls -l /dev/console | awk '{ print $3 }'`"
getClass=`dscl '/Active Directory/ACADEMIC/All Domains' -read /Users/$getUser dsAttrTypeNative:distinguishedName | \
awk '{ FS=","; print $2 }' | \
awk '{ FS="="; print $2 }' | \
tail -1`"

#-----JAMF API WAY-----

#API login info
apiUser=APIUSERNAME
apiPass=APIPASSWORD
jamfProURL="https://yourjamfproserver:8443"

#update via serial number
apiURL="JSSResource/computers/serialnumber"
MacSerial=$(ioreg -l | grep IOPlatformSerialNumber | awk '{print $NF}' | tr -d "\n")
#XML header stuff
xmlHeader="xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?&gt;"

#API data load
apiData=&lt;computer&gt;&lt;location&gt;&lt;username&gt;$getUser&lt;/username&gt;&lt;real_name&gt;$getRealName&lt;/real_name&gt;&lt;department&gt;$getClass&lt;/department&gt;&lt;/location&gt;&lt;/computer&gt;
curl -sSkiu ${apiUser}:${apiPass} "${jamfProURL}/${apiURL}/${MacSerial}" \
-H "Content-Type: text/xml" \
-d "${xmlHeader}${apiData}" \
-X PUT &gt; /dev/null</pre
```



What the heck is an API?

API = Application Program Interface

Which actually means....what?

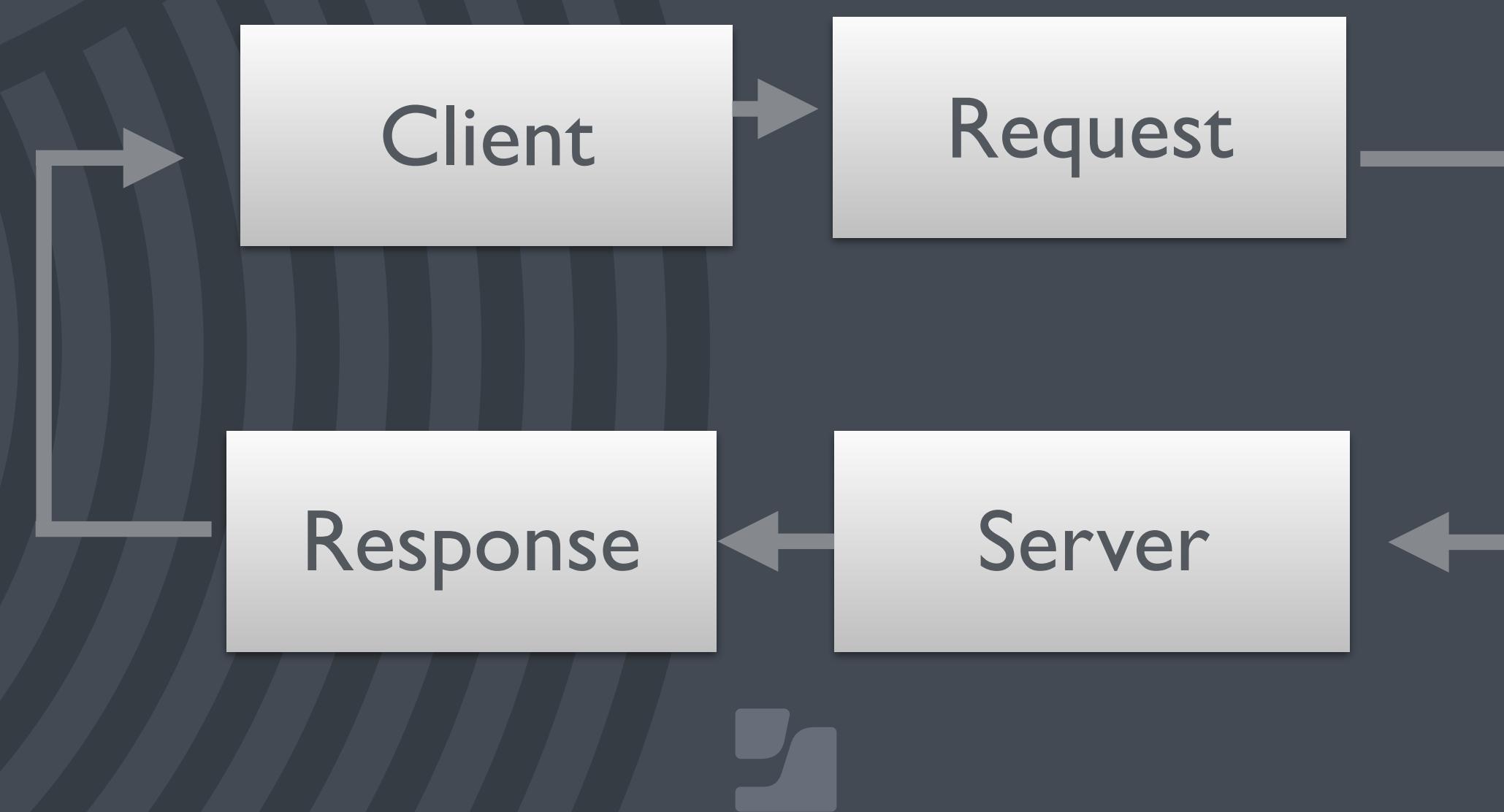
It's an Interface that allows you to easily Program and build programs or scripts to interact with the Application



The Jamf API

The Jamf API is a RESTful API

- REpresentational State Transfer
- Request = URL, HTTP Method, Headers, and Form Data
- Response = Status Code, Header, and Body



API Functions

CRUD

- Create - POST
- Read - GET
- Update - PUT
- Delete - DELETE



2 Formats

XML vs JSON

- Both similar to each other
- Both are different from each other
- Jamf API can **ONLY** accept XML but can return either



```
<?xml version="1.0" encoding="UTF-8"?>
<computer>
    <general>
        <id>2005</id>
        <name>FA0001</name>
        <mac_address>D4:61:9D:39:83:44</mac_address>
        <alt_mac_address>9A:00:0F:B4:27:10</alt_mac_address>
        <ip_address>10.10.220.237</ip_address>
        <last_reported_ip>10.10.220.237</last_reported_ip>
        <serial_number>C1MTX3Q9J1WK</serial_number>
        <udid>A575CB1D-BE8D-52E8-A750-9ED1AEBF9179</udid>
        <jamf_version>9.99.0-t1494340586</jamf_version>
        <platform>Mac</platform>
        ...
    </general>
</computer>
```



```
{  
  "computer":{  
    "general":{  
      "id":2005,  
      "name":"FA0001",  
      "mac_address":"D4:61:9D:39:83:44",  
      "alt_mac_address":"9A:00:0F:B4:27:10",  
      "ip_address":"10.10.220.237",  
      "last_reported_ip":"10.10.220.237",  
      "serial_number":"C1MTX3Q9J1WK",  
      "udid":"A575CB1D-BE8D-52E8-A750-9ED1AEBF9179",  
      "jamf_version":"9.99.0-t1494340586",  
      "platform":"Mac",  
      ...  
    }  
  }  
}
```



<https://codebeautify.org/jsonviewer>

How to use the Jamf API

<https://YOURJAMFPROSERVER:8443/API>

JSS REST API Resource Documentation

Use of the JSS REST API is subject to the terms and conditions of the [API license agreement](#).

/accounts	Show/Hide	List Operations	Expand Operations
/activationcode	Show/Hide	List Operations	Expand Operations
/advancedcomputersearches	Show/Hide	List Operations	Expand Operations
/advancedmobiledevicesearches	Show/Hide	List Operations	Expand Operations
/advancedusersearches	Show/Hide	List Operations	Expand Operations
/allowedfileextensions	Show/Hide	List Operations	Expand Operations
/buildings	Show/Hide	List Operations	Expand Operations
/byoprofiles	Show/Hide	List Operations	Expand Operations
/categories	Show/Hide	List Operations	Expand Operations
/classes	Show/Hide	List Operations	Expand Operations
/commandflush	Show/Hide	List Operations	Expand Operations
/computerapplications	Show/Hide	List Operations	Expand Operations
/computerapplicationusage	Show/Hide	List Operations	Expand Operations
/computercheckin	Show/Hide	List Operations	Expand Operations
/computercommands	Show/Hide	List Operations	Expand Operations
/computerconfigurations	Show/Hide	List Operations	Expand Operations
/computerextensionattributes	Show/Hide	List Operations	Expand Operations
/computergroups	Show/Hide	List Operations	Expand Operations



Tools to use for the API

- API Documentation
- Postman
- Curl
- Language of choice
(BASH, Python, Perl, PHP, Swift, Ruby...)



JSS REST API Resource Documentation

Use of the JSS REST API is subject to the terms and conditions of the [API license agreement](#).

/accounts	Show/Hide	List Operations	Expand Operations
/activationcode	Show/Hide	List Operations	Expand Operations
/advancedcomputersearches	Show/Hide	List Operations	Expand Operations
✓ /advancedmobiledevicesearches	Show/Hide	List Operations	Expand Operations
/advancedusersearches	Show/Hide	List Operations	Expand Operations
/allowedfileextensions	Show/Hide	List Operations	Expand Operations
/buildings	Show/Hide	List Operations	Expand Operations
/byoprofiles	Show/Hide	List Operations	Expand Operations
/categories	Show/Hide	List Operations	Expand Operations
/classes	Show/Hide	List Operations	Expand Operations
/commandflush	Show/Hide	List Operations	Expand Operations
/computerapplications	Show/Hide	List Operations	Expand Operations
/computerapplicationusage	Show/Hide	List Operations	Expand Operations
/computercheckin	Show/Hide	List Operations	Expand Operations
/computercommands	Show/Hide	List Operations	Expand Operations
/computerconfigurations	Show/Hide	List Operations	Expand Operations
/computerextensionattributes	Show/Hide	List Operations	Expand Operations
/computergroups	Show/Hide	List Operations	Expand Operations



Postman

<https://www.getpostman.com>



Postman

Runner Import +

Builder Team Library

SYNC OFF Sign In

No Environment

New Tab + ...

History Collections

GET Enter request URL Params Send Save

Authorization Headers Body Pre-request Script Tests Cookies Code

Type No Auth

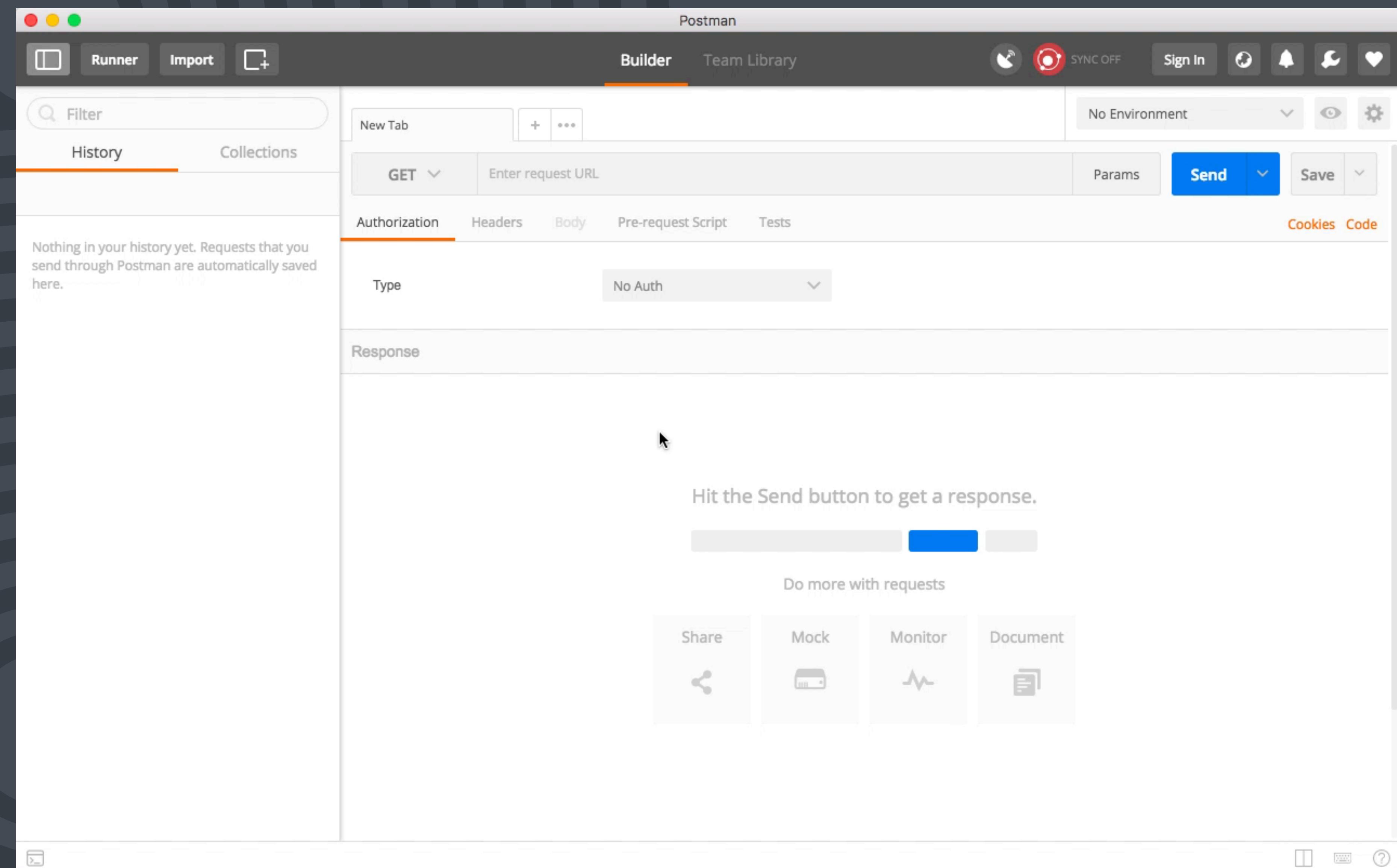
Nothing in your history yet. Requests that you send through Postman are automatically saved here.

Response

Hit the Send button to get a response.

Do more with requests

Share Mock Monitor Document



Postman

Runner Import

Builder Team Library

No Environment

Filter https://casper.saes.org

History Collections Clear all

GET https://casper.saes.org:8443/JSSResource/advancedcomputersearches/name/loaners Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Cookies Code

Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	text/xml				
<input checked="" type="checkbox"/> Authorization	Basic ZGVtbzokWGYwM2J5c2ludWIwZnVmdQ==				
New key	Value	Description			

Body Cookies Headers (8) Tests Status: 200 OK Time: 804 ms Size: 10.1 KB

Pretty Raw Preview XML

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <advanced_computer_search>
3    <id>68</id>
4    <name>Loaners</name>
5    <view_as>Standard Web Page</view_as>
6    <sort_1/>
7    <sort_2/>
8    <sort_3/>
9    <criteria>
10       <size>1</size>
11       <criterion>
12          <name>Computer Group</name>
13          <priority>0</priority>
14          <and_or>and</and_or>
15          <search_type>member of</search_type>
16          <value>Loaners</value>
17          <opening_paren>false</opening_paren>
18          <closing_paren>false</closing_paren>
19       </criterion>
20     </criteria>
21   <display_fields>

```



Curl

Curl is actually a powerful tool NOT just useful for API task

- API scripting
- Download files off of a web server
 - curl -o LOCALFILEPATH SERVERFILEPATH
- Check the Status of a web server
 - curl -sLI WEBSITE
- Get an external IP address quickly
 - curl ipinfo.io



curl for Jamf API use

```
curl
```

```
-X GET
```

```
https://YOURJAMFPROSERVER:8443/JSSResource/computers
```

```
-u YOURUSERNAME
```



Examples in BASH with XML





```
-----JAMF API STUFF-----  
  
#API login info  
apiuser='API_USER_NAME'  
apipass='API_PASSWORD'  
jamfProURL="https://YourJAMFProServer:8443"  
  
#update via serial number  
apiURL="JSSResource/computers/serialnumber"  
MacSerial=$(ioreg -l | grep IOPlatformSerialNumber | awk '{print $NF}' | tr -d "\")  
  
#XML header stuff  
xmlHeader="xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?&gt;"<br/  
#API data load  
apiData="  
<computer>  
  <location>  
    <username>$getUser</username>  
    <real_name>$getRealName</real_name>  
    <department>$getClass</department>  
  </location>  
</computer>"  
curl -sSu ${apiuser}:${apiPass} "${jamfProURL}/${apiURL}/${MacSerial}" \  
-H "Content-Type: text/xml" \  
-d "${xmlHeader}${apiData}" \  
-X PUT > /dev/null
```

```
#!/bin/sh

IFS=$','

#API login info
apiuser='API_USER_NAME'
apipass='API_PASSWORD'
jamfProURL="https://YourJAMFProServer:8443"

APIresult=$(curl -s -X GET "$jamfProURL/JSSResource/departments" \
-u "$apiuser:$apipass" \
-H "accept: text/xml" \
| xmllint --format - | \
awk -F '>|<' '/<name>/ {print $3",")}'

echo $APIresult
```



Examples in Python with JSON



```
#!/usr/bin/python

import urllib2
import base64
import json

#set the layer we're looking into
computers = response_data['computer_reports']

#loops and stuff
for record in computers:
    student_laptop = record['Computer_Name']
    student_username = record['Username']
    Email = record['Email_Address']
    FullName = record['Full_Name']
    FirstName = FullName.strip(" ")
    FirstName = FirstName.split(' ')[0]
    mycommand = "echo 'Hello "+FirstName+",\n\nBlahBlahBlah Email Body' | mail -s 'My Email Subject' " + Email
    os.system(mycommand)
    print mycommand
```

```
-----
```

```
reponse = urllib2.urlopen(request, context=context)
```

```
#request the json data
response = urllib2.urlopen(request)
response_data = json.loads(response.read())
```

```
print mycommand
```

```
{  
  "computer_reports": [  
    {  
      "Computer_Name": "SDA0070",  
      "Username": "e20freedman",  
      "Full_Name": " Ethan Freedman",  
      "Email_Address": "e20freedman@school.edu",  
      "Department": "Class of 20",  
      "Last_Check_in": "2017-06-08 08:14:40"  
    },  
    {  
      "Computer_Name": "SDA0092",  
      "Username": "s21flanders",  
      "Full_Name": " Stephen 21. Flanders",  
      "Email_Address": "s21flanders@school.edu",  
      "Department": "Class of 21",  
      "Last_Check_in": "2017-06-06 21:07:45"  
    },  
  ]  
}
```



Example of using the API with PHP



<u>Laptop</u>	<u>User</u>	<u>Date Out</u>	<u>Date Returned</u>	
1320loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
AA2009loaner		mm / dd / yyyy	11/11/2011	<input type="button" value="Submit"/>
AA2013loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
AA2061loaner		mm / dd / yyyy	2017/02/07	<input type="button" value="Submit"/>
AA2076LOANER		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
AA2088loaner		mm / dd / yyyy	2017/08/25	<input type="button" value="Submit"/>
AA2102loaner		mm / dd / yyyy	3/4/2005	<input type="button" value="Submit"/>
AA2113loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
AA2114loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
AA2119loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
BA2011loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
BA2110loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
BA2151loaner		mm / dd / yyyy	2017/06/05	<input type="button" value="Submit"/>
BA2215loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
BA2296loaner		mm / dd / yyyy	2017/09/11	<input type="button" value="Submit"/>
BA2350loaner		mm / dd / yyyy	2017/09/13	<input type="button" value="Submit"/>
SAA2039loaner		mm / dd / yyyy	2017-06-08	<input type="button" value="Submit"/>
SAA2094loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
SBA2023loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
SBA2134loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>
SBA2231loaner		mm / dd / yyyy	2017/09/13	<input type="button" value="Submit"/>
SBA2232loaner		mm / dd / yyyy	2017-09-20	<input type="button" value="Submit"/>





```
<tr style="font-size: 15pt; text-decoration: underline;">
    <td>Laptop</td>
    <td>User</td>
    <td>Date Out</td>
    <td>Date Returned</td>
    <td></td>
</tr>

<?php
#Advanced Saved Computer Search ID needed here
#Fields need to be shown in the Search - Name, DateReturned, DateOut, Availability, Username, Department, Serial Number, JSS ID
$Read_result=ReadJSS("computerreports/id/68");

$TodayDate = date('Y/m/d');

$jss_array = json_decode($Read_result, true);
$device_array = $jss_array['computer_reports'];
foreach($device_array as $device){
    echo "<form action=\"\" method=\"Post\" name=\"laptopsform\" id=\"laptopsform\" onSubmit=\"return DoubleCheck('".$device['Computer_Name']."' );\">";
    echo "<tr style=\"font-size: 12pt;\">";
    echo "<input type=\"hidden\" name=\"laptop\" value=\"".$device['Serial_Number']."\">";
    echo "<td><a href=\"https://casper.saes.org:8443/computers.html?id=".$device['JSS_Computer_ID']."' target=\"_blank\">".$device['Computer_Name']."></a></td>";
    if($device['Availability'] == "No"){
        echo "<td><a href=\"mailto:".$device['Username']."@saes.org\" target=\"_blank\">".$device['Username']."></a></td>";
        echo "<td>".$device['DateOut']."></td>";
        echo "<td><input type=\"date\" name=\"DateReturned\" value=\"".$TodayDate."\"></td>";
        echo "<input type=\"hidden\" name=\"avail\" value=\"checkin\">";

        echo "<td><input type=\"submit\" value=\"Submit\" id=\"submit\" name=\"submit\"></td>";
        echo "</form>";
    } else if ($device['Availability'] == "Yes"){
        echo "<td><input type=\"text\" name=\"User\"></td>";

        echo "<td><input type=\"date\" name=\"DateOut\" value=\"".$TodayDate."\"></td>";
        echo "<td>".$device['DateReturned']."></td>";

        echo "<input type=\"hidden\" name=\"avail\" value=\"checkout\">";
        echo "<td><input type=\"submit\" value=\"Submit\" id=\"submit\" name=\"submit\"></td>";
        echo "</form>";
    }
}

?>
</div>
</BODY>
</HTML>
```

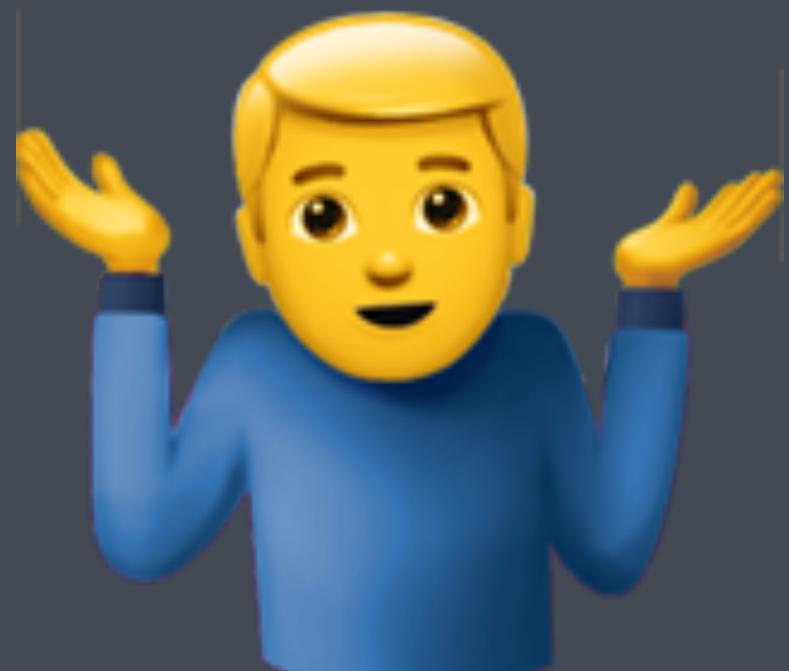
Other Small Projects

- Update iPad owner info via a CSV
- Update every Mobile App's Self Service status
- Export all Extension Attributes and Scripts in the JSS
- Move a computer to a Static Group via Self Service policy



Problems with the API

- Security
 - Permissions
 - Plain Text Passwords
- Not Always Clear
- Doesn't always work



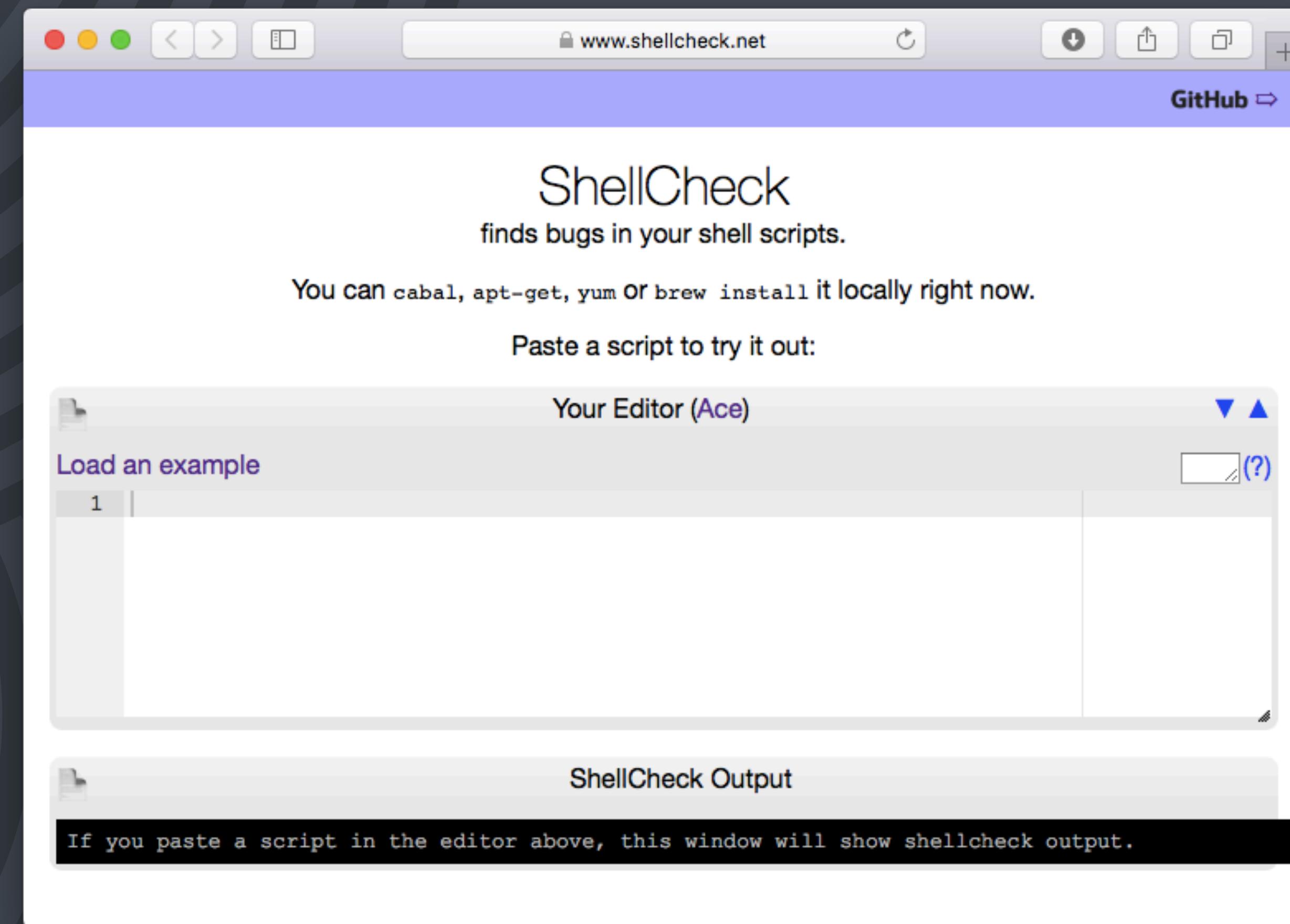
Resources to learn the Jamf API

- **Jamf Nation**
<http://jamfnation.jamfsoftware.com>
- **The Unofficial Reference for the JAMF Pro API**
<https://unofficial-jss-api-docs.atlassian.net>
- **The JSS REST API for Everyone**
<http://www.macbrained.org/the-jss-rest-api-for-everyone/>
- **MacAdmins Slack Channel**



One last resource...

www.shellcheck.net



<https://github.com/boberito>



Everyone can learn.



Thank you!

