Jamf Pro API with Swift

virtual JNUC 2021

Richard Mallion

dataJAR Ltd

Email: richard@datajar.co.uk
Twitter: @richardmallion
MacAdmins Slack: @red5

Jamf Documentation

https://developer.jamf.com

https://xxx.jamfcloud.com/api

Two sets of APIs

Classic API

Jamf Pro API

Classic API

XML and JSON format responses for GET requests

Limited to XML data formats for PUT and POST requests

Classic API Base URL

https://xxx.jamfcloud.com/JSSResource

Authentication - Classic API

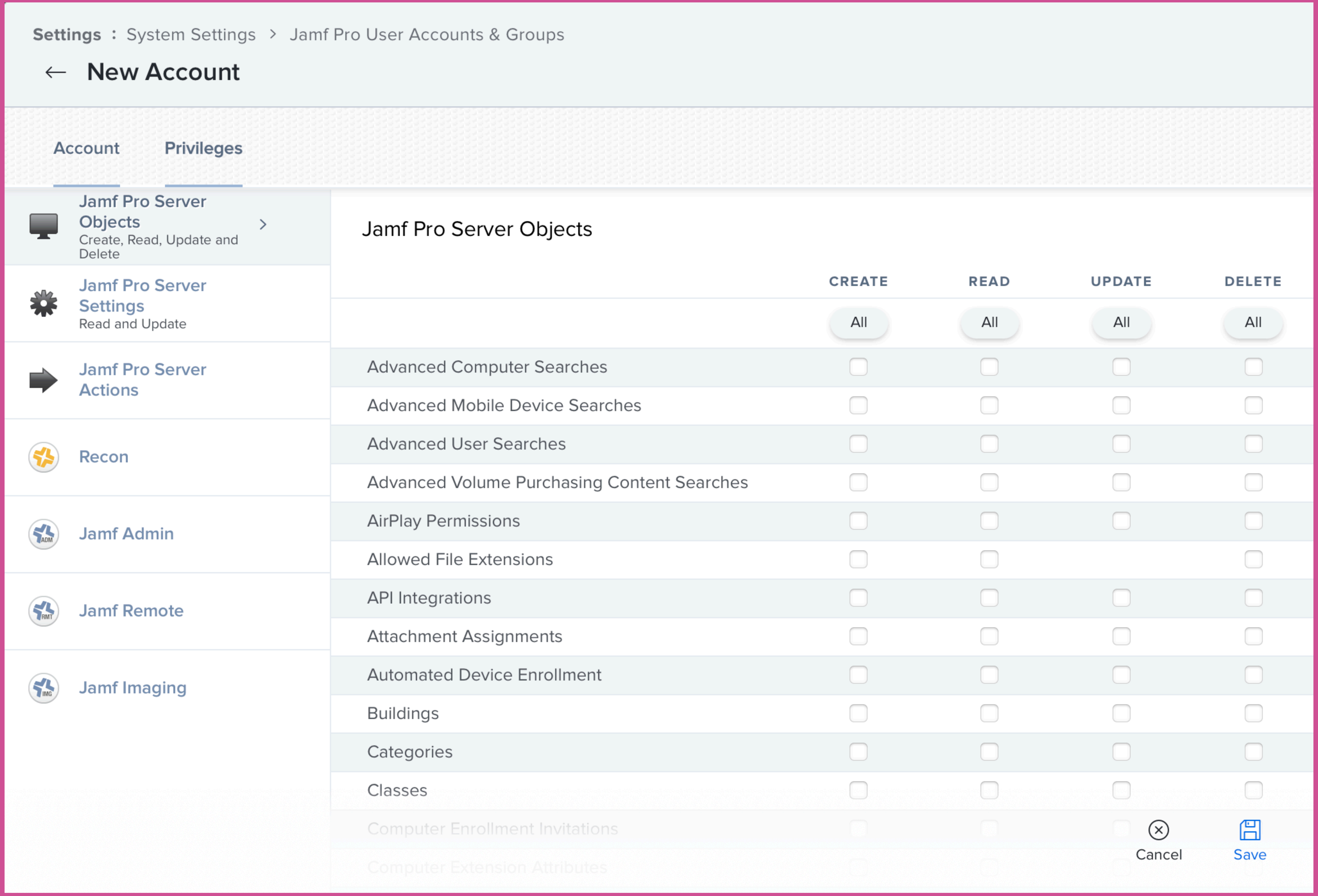
Supports Basic Authentication and uses the standard User Accounts and Groups

Authorisation

Set the account's permissions to match the permissions required for the API endpoint

https://developer.jamf.com/jamf-pro/docs/classic-api-minimum-required-privileges-and-endpoint-mapping

Authorisation - Objects



Settings : System Settings > Jamf Pro User Accounts & Groups

← New Account

Account | Privileges

- Jamf Pro Server Objects — Create, Read, Update and Delete
- Jamf Pro Server Settings — Read and Update
- Jamf Pro Server Actions
- Recon
- Jamf Admin
- Jamf Remote
- Jamf Imaging

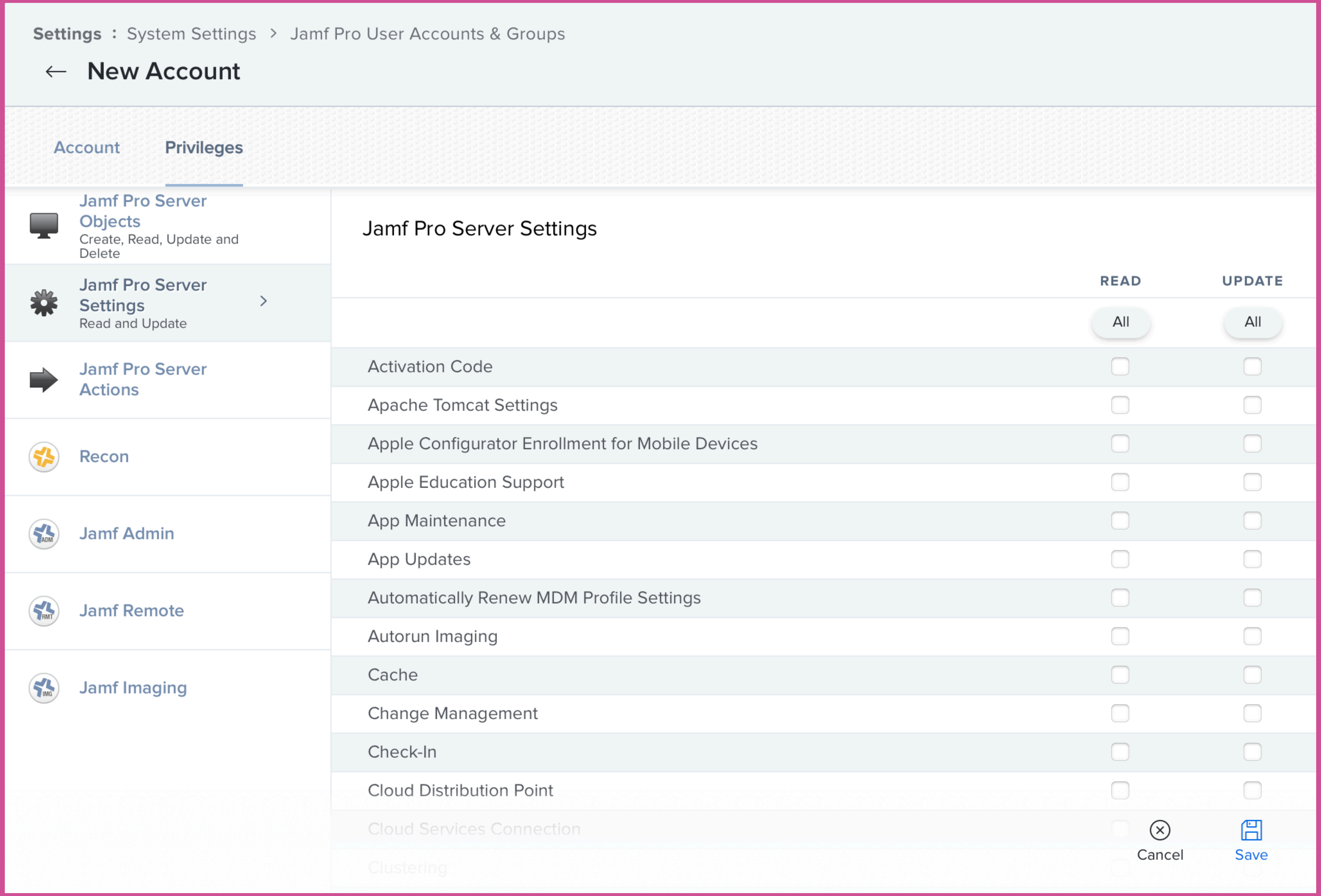
Jamf Pro Server Objects

	CREATE	READ	UPDATE	DELETE
	All	All	All	All
Advanced Computer Searches	☐	☐	☐	☐
Advanced Mobile Device Searches	☐	☐	☐	☐
Advanced User Searches	☐	☐	☐	☐
Advanced Volume Purchasing Content Searches	☐	☐	☐	☐
AirPlay Permissions	☐	☐	☐	☐
Allowed File Extensions	☐	☐		☐
API Integrations	☐	☐	☐	☐
Attachment Assignments	☐	☐	☐	☐
Automated Device Enrollment	☐	☐	☐	☐
Buildings	☐	☐	☐	☐
Categories	☐	☐	☐	☐
Classes	☐	☐	☐	☐
Computer Enrollment Invitations	☐	☐	☐	☐
Computer Extension Attributes	☐	☐	☐	☐

Cancel | Save

virtual JNUC 2021

Authorisation - Settings



Settings : System Settings > Jamf Pro User Accounts & Groups

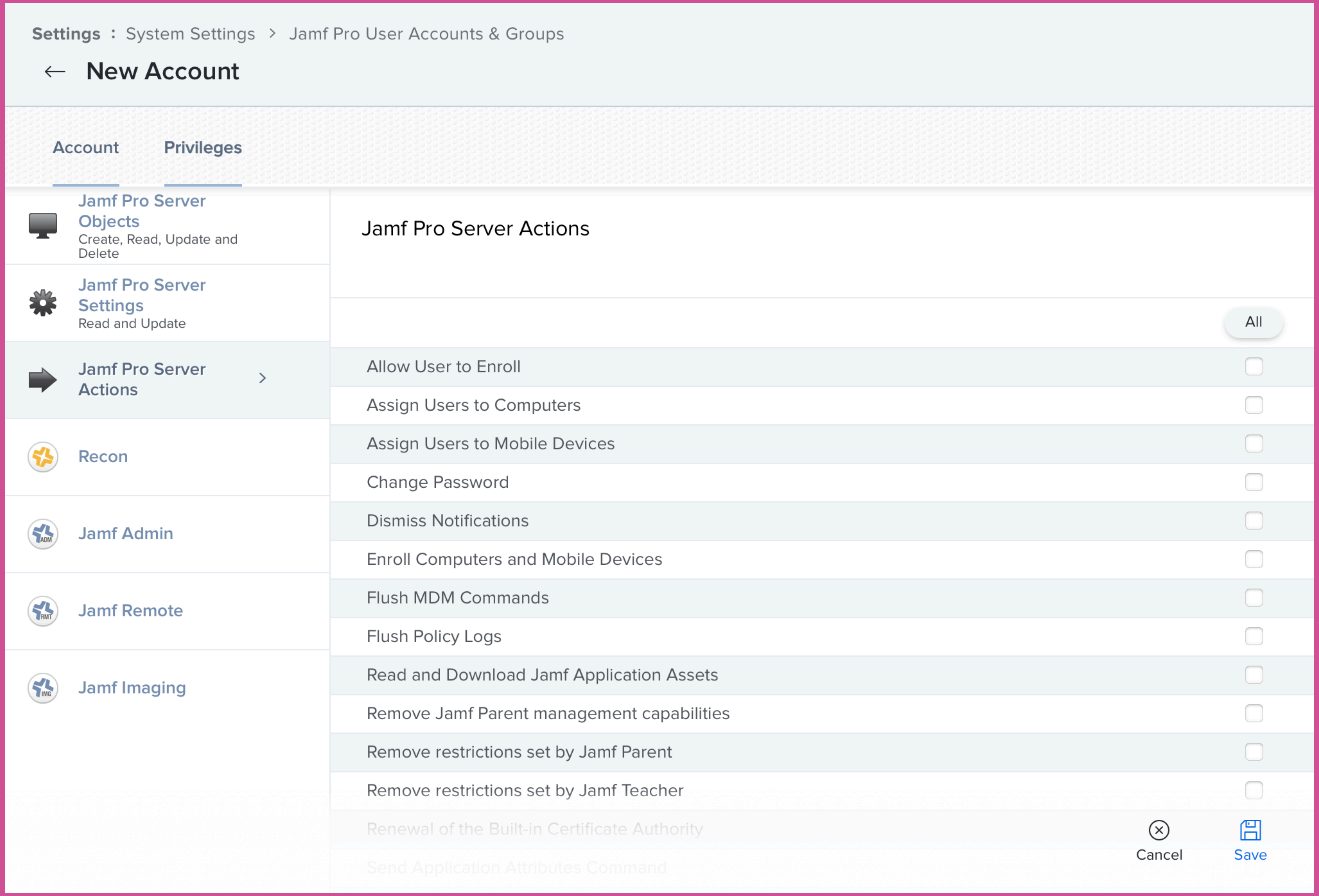
← New Account

Jamf Pro Server Settings

	READ	UPDATE
	All	All
Activation Code	☐	☐
Apache Tomcat Settings	☐	☐
Apple Configurator Enrollment for Mobile Devices	☐	☐
Apple Education Support	☐	☐
App Maintenance	☐	☐
App Updates	☐	☐
Automatically Renew MDM Profile Settings	☐	☐
Autorun Imaging	☐	☐
Cache	☐	☐
Change Management	☐	☐
Check-In	☐	☐
Cloud Distribution Point	☐	☐
Cloud Services Connection	☐	
Clustering	☐	

virtual JNUC 2021

Authorisation - Actions



Settings : System Settings > Jamf Pro User Accounts & Groups

New Account

Account | Privileges

- Jamf Pro Server Objects
  Create, Read, Update and Delete
- Jamf Pro Server Settings
  Read and Update
- Jamf Pro Server Actions
- Recon
- Jamf Admin
- Jamf Remote
- Jamf Imaging

Jamf Pro Server Actions

	All
Allow User to Enroll	☐
Assign Users to Computers	☐
Assign Users to Mobile Devices	☐
Change Password	☐
Dismiss Notifications	☐
Enroll Computers and Mobile Devices	☐
Flush MDM Commands	☐
Flush Policy Logs	☐
Read and Download Jamf Application Assets	☐
Remove Jamf Parent management capabilities	☐
Remove restrictions set by Jamf Parent	☐
Remove restrictions set by Jamf Teacher	☐
Renewal of the Built-in Certificate Authority	
Send Application Attributes Command	

virtual JNUC 2021

Authorisation

Jamf	***HTTP***
Create	*POST*
Read	*GET*
Update	*PUT*
Delete	*DELETE*

Classic - Credentials

Credential = Base64 encoded username:password

```
let authString = "Richard" + ":" + "password"
```

Classic - Credentials

Credential = Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString
```

Classic - Credentials

Credential = Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString.data(using: .utf8)?
```

Classic - Credentials

Credential = Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString.data(using: .utf8)?.base64EncodedString()
```

virtual JNUC 2021

Classic - Credentials

Credential = Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString.data(using: .utf8)?.base64EncodedString()
```

Quick overview of optionals

- Optionals are used in situations where a value may be absent
  - There is a value and it equals x
  - There is no value at all: nil

Quick overview of optionals

- Optionals are created by appending `?` to a type
- So, `Int` must always contain a real integer
- An `Int?` might be an integer or might be missing a value , nil

Quick overview of optionals

- Optionals are best thought of as boxes that may or may not contain a value
- Because optionals may or may not be empty, you cannot use them freely
- You have to unwrap them to get to the actual value

Quick overview of optionals

- You can force unwrap an optional
- But, this will cause a runtime error if the optional is empty, nil
- Only do this if you are 100% sure a value exists
- To force unwrap, append the optional with a !
- let statusCode = optionalStatusCode!

Quick overview of optionals

- A safer approach is to check if the optional is not nil and then unwrap
- We can use an if/let statement called optional binding
- It evaluates to true only if the optional is not nil, it then automatically unwraps it

```
If let statusCode = optionalStatusCode {
    // we have a real value , statusCode
}
```

1. Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString.data(using: .utf8)?.base64EncodedString()
```

virtual JNUC 2021

2. URL of computers endpoint

```
let computerURLString = "https://xxx.jamfcloud.com/JSSResource/computers"

let computerURL = URL(string: computerURLString)!
```

3. Create a URLRequest

`URLRequest` encapsulates two essential properties of a load request

- The URL to load
- The policies used to load it

In addition, for HTTP and HTTPS requests, `URLRequest` includes the HTTP method (`GET`, `POST`, and so on) and the HTTP headers.

- Accept: indicates what kind of response from the server the client can accept
- Content-Type: is about the content of the current request or response. So if your request has no payload, you don't use a content-type request header.

3. Create a URLRequest

```
var request = URLRequest(url: computerURL)
```

virtual JNUC 2021

3. Create a URLRequest

```
var request = URLRequest(url: computerURL)
request.httpMethod = "GET"
```

3. Create a URLRequest

```
var request = URLRequest(url: computerURL)
request.httpMethod = "GET"
request.setValue("Basic \(base64)", forHTTPHeaderField: “Authorization")
```

3. Create a URLRequest

```
var request = URLRequest(url: computerURL)
request.httpMethod = "GET"
request.setValue("Basic \(base64)", forHTTPHeaderField: “Authorization")
request.setValue("application/json", forHTTPHeaderField: “Accept")
```

4. URLSession, pass it the request and a closure to handle the response

The `URLSession` class and related classes provide an API for downloading data from and uploading data to endpoints indicated by URLs.

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request)
```

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in

// Code to handle response goes here

}
```

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in


// Code to handle response goes here

}
task.resume()
```

5. data, response and error

data: The data returned by the server

response: An object that provides response metadata, such as HTTP headers and status codes

error: An error object that indicates why the request failed, nil if the request was successful

Response Codes

200	Request successful
201	Request to create or update object successful
400	Bad request.
401	Authentication failed.
403	Invalid permissions.
404	Resource not found.
409	Conflict.
500	Internal server error.
502	Bad Gateway.

5. Checking the error and response code

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in



}
task.resume()
```

5. Checking the error and response code

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    }
















}
task.resume()
```

5. Checking the error and response code

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    }







}
task.resume()
```

6. Handle the returned data

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {







        }
    }
}
task.resume()
```

6. Handle the returned data

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let str = String(decoding: data, as: UTF8.self)
            print(str)




        }
    }
}
task.resume()
```

6. Handle the returned data

```
{
    "computers":
    [
        {"id":27,"name":"Mac 1"},
        {"id":28,"name":"Mac 2"}
    ]
}
```

7. Swift data model

```swift
struct AllComputers: Codable {
    let computers: [Computer]
}

struct Computer: Codable {
    let id: Int
    let name: String
}
```

```json
{
        "computers":
        [
                {"id":27,"name":"Mac 1"},
                {"id":28,"name":"Mac 2"}
        ]
}
```

8. Decode the returned json

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {








        }
    }
}
task.resume()
```

8. Decode the returned json

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let decoder = JSONDecoder()







        }
    }
}
task.resume()
```

8. Decode the returned json

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let decoder = JSONDecoder()

            if let computers = try? decoder.decode(AllComputers.self, from: data) {

                //Handle data

                print(computers)
            }
        }
    }
}
task.resume()
```

Classic - Get Example

Find computer by ID

https://xx.jamfcloud.com/JSSResource/computers/id

1. URL of computers endpoint with id of computer

```
let computerURLString = "https://xxx.jamfcloud.com/JSSResource/computers/id/28"

let computerURL = URL(string: computerURLString)!
```

2. A lot more information

2. A lot more information

```
{
    "general":{...}
    "location":{...}
    "purchasing":{...}
    "peripherals":{...}
    "hardware":{...}
    "certificates":[...]
    "software":{...}
    "extension_attributes":[...]
    "groups_accounts":{...}
    "configuration_profiles":[...]
}
```

3. Create a Swift Model?

```swift
struct Computer: Codable {

    let item1: String
    //.....
    let item100: String

}
```

```json
{
"general":{...}
"location":{...}
"purchasing":{...}
"peripherals":{...}
"hardware":{...}
"certificates":[...]
"software":{...}
"extension_attributes":[...]
"groups_accounts":{...}
"configuration_profiles":[...]
}
```

4. Return a subset of data

```
let computerURLString = "https://xxx.jamfcloud.com/JSSResource/computers/id/28/subset/general&location&hardware"

let computerURL = URL(string: computerURLString)!
```

```
{
"general":{...}
"location":{...}
"hardware":{...}
}
```

More manageable but maybe still not all needed?

What happens if more attributes are added?

```
{
"general":{
"id":1
"name":"Admins iMac"
"mac_address":"E0:AC:CB:97:36:G4"
"network_adapter_type":"Ethernet"
"alt_mac_address":"E0:AC:CB:97:36:G4"
"alt_network_adapter_type":"IEEE80211"
"ip_address":"10.1.1.1"
"last_reported_ip":"192.0.0.1"
"serial_number":"C02Q7KHTGFWF"
"udid":"55900BDC-347C-58B1-D249-F32244B11D30"
"jamf_version":"9.99.0-t1494340586"
"platform":"Mac"
"barcode_1":"string"
"barcode_2":"string"
"asset_tag":"string"
"remote_management":{...}
"mdm_capable":true
"mdm_capable_users":{...}
"management_status":{...}
"report_date":"2021-05-24T12:17:18.822Z"
"report_date_epoch":1499470624555
"report_date_utc":"2017-07-07T18:37:04.555-0500"
"last_contact_time":"2021-05-24T12:17:18.822Z"
"last_contact_time_epoch":1499470624555
"last_contact_time_utc":"2017-07-07T18:37:04.555-0500"
"initial_entry_date":"2021-05-24T12:17:18.822Z"
"initial_entry_date_epoch":1499470624555
"initial_entry_date_utc":"2017-07-07T18:37:04.555-0500"
"last_cloud_backup_date_epoch":1499470624555
"last_cloud_backup_date_utc":"2017-07-07T18:37:04.555-0500"
"last_enrolled_date_epoch":1499470624555
"last_enrolled_date_utc":"2017-07-07T18:37:04.555-0500"
"distribution_point":"string"
"sus":"string"
"netboot_server":"string"
"site":{...}
"itunes_store_account_is_active":true
}

"location":{
"username":"JBetty"
"realname":"Betty Jackson"
"real_name":"Betty Jackson"
"email_address":"jbetty@company.com"
"position":"Systems Engineer"
"phone":"123-555-6789"
"phone_number":"123-555-6789"
"department":"Sales Staff"
"building":"New York Office"
"room":1159
}

"hardware":{
"make":"Apple"
"model":"13-inch Retina MacBook Pro (Late 2013)"
"model_identifier":"MacBookPro11,1"
"os_name":"Mac OS X"
"os_version":"10.13.2"
"os_build":"17C88"
"master_password_set":true
"active_directory_status":"AD.company.com"
"service_pack":"string"
"processor_type":"Intel Core i5"
"processor_architechture":"x86_64"
"processor_speed":2600
"processor_speed_mhz":2600
"number_processors":1
"number_cores":2
"total_ram":16384
"total_ram_mb":16384
"boot_rom":"MBP111.0142.B00"
"bus_speed":0
"bus_speed_mhz":0
"battery_capacity":90
"cache_size":3072
"cache_size_kb":3072
"available_ram_slots":0
"optical_drive":"string"
"nic_speed":"n/a"
"smc_version":"2.16f68"
"ble_capable":true
"sip_status":"Enabled"
"gatekeeper_status":"App Store and identified developers"
"xprotect_version":2098
"institutional_recovery_key":"Not Present"
"disk_encryption_configuration":"Individual and Institutional Encryption"
"filevault_2_users":[...]
"storage":[...]
"mapped_printers":[...]
}
```

virtual JNUC 2021

5. Build a model for the attributes of interest

```
struct Subset: Codable {
    let computer: Computer
}

struct Computer: Codable {
    let general: General
    let location: Location
    let hardware: Hardware
}
```

```
struct General: Codable {
    let id: Int
    let name: String,
    let macAddress: String,
    let serialNumber: String,
    let lastContactTime: String
}

struct Hardware: Codable {
    let model: String,
    let osVersion: String
}

struct Location: Codable {
    let username: String,
    let realName: String,
    let emailAddress: String
}
```

6. Add CodingKeys that serve as the authoritative list of properties

```
struct Hardware: Codable {
    let model: String
    let osVersion: String





}
```

6. Add CodingKeys that serve as the authoritative list of properties

```
struct Hardware: Codable {
    let model: String
    let osVersion: String

    enum CodingKeys: String, CodingKey {
        case model
        case osVersion = "os_version"
    }
}
```

6. Add CodingKeys that serve as the authoritative list of properties

```
struct Subset: Codable {
    let computer: Computer
}

struct Computer: Codable {
    let general: General
    let location: Location
    let hardware: Hardware
}
```

```
struct General: Codable {
    let id: Int
    let name, macAddress, serialNumber, lastContactTime: String

    enum CodingKeys: String, CodingKey {
        case id, name
        case macAddress = "mac_address"
        case serialNumber = "serial_number"
        case lastContactTime = "last_contact_time"
    }
}

struct Hardware: Codable {
    let model, osVersion: String

    enum CodingKeys: String, CodingKey {
        case model
        case osVersion = "os_version"
    }
}

struct Location: Codable {
    let username, realName, emailAddress: String

    enum CodingKeys: String, CodingKey {
        case username
        case realName = "real_name"
        case emailAddress = "email_address"
    }
}
```

7. Create a URLRequest

```
var request = URLRequest(url: url)
request.httpMethod = "GET"
request.setValue("Basic \(apikey)", forHTTPHeaderField: "Authorization")
request.setValue("application/json", forHTTPHeaderField: "Accept")
```

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in









}
task.resume()
```

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle errror
    }








}
task.resume()
```

virtual JNUC 2021

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle errror
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    }









}
task.resume()
```

virtual JNUC 2021

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle errror
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {






        }
    }
}
task.resume()
```

virtual JNUC 2021

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle errror
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let decoder = JSONDecoder()





        }
    }
}
task.resume()
```

8. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle errror
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let decoder = JSONDecoder()
            if let computer = try? decoder.decode(Subset.self, from: data) {
                //Handle data
                print(computer)
            }
        }
    }
}
task.resume()
```

Classic - POST Example

MDM Command - Mobile Device, Enable Bluetooth

https://xx.jamfcloud.com/JSSResource/mobiledevicecommands/command

1. URL of mobile endpoint with command and id of mobile device

```
let mobileURLString = "https://xxx.jamfcloud.com/JSSResource/mobiledevicecommands/command/SettingsEnableBluetooth/id/4"

let mobileURL = URL(string: mobileURLString)!
```

BlankPush
ClearPasscode
ClearRestrictionsPassword
DeviceLocation
DisableLostMode
EnableLostMode
EraseDevice
PasscodeLockGracePeriod
PlayLostModeSound
RestartDevice
Settings
SettingsDisableAppAnalytics
SettingsDisableBluetooth
SettingsEnablePersonalHotspot
Sett ingsDisablePersonalHotspot
SettingsDisableDataRoaming
SettingsDisableDiagnosticSubmission
SettingsDisableVoiceRoaming
SettingsEnableAppAnalytics
SettingsEnableBluetooth
SettingsEnableDataRoaming
SettingsEnableDiagnosticSubmission
SettingsEnableVoiceRoaming
ShutDownDevice
UnmanageDevice
UpdateInventory

2. Create a URLRequest

```
var request = URLRequest(url: mobileURL)
request.httpMethod = “POST"
request.setValue("Basic \(base64)", forHTTPHeaderField: “Authorization")
```

3. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 201 {
        //handle error
    } else {
        print("Bluetooth was enabled")
    }
}
task.resume()
```

Classic - POST Example

MDM Command - Mobile Device, LostMode with message

https://xx.jamfcloud.com/JSSResource/mobiledevicecommands/command

1. URL of mobile endpoint with command and id of mobile device

```
let mobileURLString = "https://xxx.jamfcloud.com/JSSResource/mobiledevicecommands/
command/EnableLostMode"

let mobileURL = URL(string: mobileURLString)!
```

2. Create a URLRequest

```
var request = URLRequest(url: mobileURL)
request.httpMethod = “POST"
request.setValue("Basic \(base64)", forHTTPHeaderField: “Authorization")
request.setValue("text/xml", forHTTPHeaderField: "Content-Type")
```

3. Create the Request Body

```
<mobile_device_command>
    <command>EnableLostMode</command>
    <lost_mode_message>\(message)</lost_mode_message>
    <lost_mode_with_sound>true</lost_mode_with_sound>
    <mobile_devices>
       <mobile_device>
       <id>\(deviceID)</id>
       </mobile_device>
    </mobile_devices>
 </mobile_device_command>
```

```
let requestBody =  "<mobile_device_command><command>EnableLostMode</
command><lost_mode_message>\(message)</
lost_mode_message><lost_mode_with_sound>true</
lost_mode_with_sound><mobile_devices><mobile_device><id>\(deviceID)</id></mobile_device></
mobile_devices></mobile_device_command>”

request.httpBody = requestBody.data(using: String.Encoding.utf8)
```

3. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in

    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 201 {
        //handle error
    } else {
        print("LostMode command successfully sent")
    }
}
task.resume()
```

Classic - PUT Example

Update an existing static mobile group

https://xx.jamfcloud.com/JSSResource/mobiledevicegroups/name

1. URL of mobile group endpoint with name of group

```
let mobileURLString = "https://xxx.jamfcloud.com/JSSResource/mobiledevicegroups/name/VIPs"

let mobileURL = URL(string: computerURLString)!
```

2. Create the URLRequest

```
var request = URLRequest(url: mobileURL)
request.httpMethod = “POST"
request.setValue("Basic \(base64)", forHTTPHeaderField: “Authorization")
request.setValue("text/xml", forHTTPHeaderField: "Content-Type")
```

3. Request Body to add a device

```
<mobile_device_group>
    <mobile_device_additions>
        <mobile_device>
            <id>4</id>
        </mobile_device>
    </mobile_device_additions>
</mobile_device_group>
```

```
let requestBody = "<mobile_device_group><mobile_device_additions><mobile_device><id>4</
id></mobile_device></mobile_device_additions></mobile_device_group>"

request.httpBody = requestBody.data(using: String.Encoding.utf8)
```

3. Request Body to remove a device

```
<mobile_device_group>
    <mobile_device_deletions>
        <mobile_device>
            <id>4</id>
        </mobile_device>
    </mobile_device_deletions>
</mobile_device_group>
```

```
let requestBody = "<mobile_device_group><mobile_device_deletions><mobile_device><id>4</
id></mobile_device></mobile_device_deletions></mobile_device_group>"

request.httpBody = requestBody.data(using: String.Encoding.utf8)
```

virtual JNUC 2021

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 201 {
        //handle error
    } else {
        print("Device added to group")
    }
}
task.resume()
```

Jamf Pro API

The Jamf Pro API was designed to interact with JSON data types when interacting with most endpoints

Some endpoints include support for interacting with other data types

Jamf Pro API Base URL

https://xxx.jamfcloud.com/api

Authorisation

Set the account's permissions to match the permissions required for the API endpoint

https://developer.jamf.com/jamf-pro/docs/privileges-and-deprecations

Authentication - Jamf Pro API

Supports Bearer Token authentication and uses the standard User Accounts and Groups

Token expires after 30 minutes

A currently valid token can be used to generate a new token with a fresh 30 minutes validity period

Jamf Pro - Authentication

Request Token by sending a POST to /v1/auth/token

1. Base64 encoded username:password

```
let authString = "Richard" + ":" + “password"

let base64 = authString.data(using: .utf8)?.base64EncodedString()
```

2. URL of auth endpoint

```
let tokenURLString = “https://xxx.jamfcloud.com/api/v1/auth/token"

let tokenURL = URL(string: tokenURLString)!
```

3. Create the URLRequest

```
var request = URLRequest(url: url)
request.httpMethod = "POST"
request.setValue("Basic \(apikey)", forHTTPHeaderField: "Authorization")
request.setValue("application/json", forHTTPHeaderField: "Accept")
```

4. Create a model for the token

```swift
struct JamfProAuth: Decodable {
    let token: String
    let expires: String
}
```

```json
{
    "token": "eyJhbGciOiJIUzUxMiJ9...",
    "expires": "2021-07-21T22:18:21.636Z"
}
```

virtual JNUC 2021

5. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            if let auth = try? JSONDecoder().decode(JamfProAuth.self, from: data) {
                print("Token: \(auth.token)")
                print("Expires: \(auth.expires)")
            }
        }
    }
}
task.resume()
```

6. Renew Token

```
let tokenURLString = "https://xxx.jamfcloud.com/api/v1/auth/keep-alive"

request.setValue("Bearer \(firstToken)", forHTTPHeaderField: "Authorization")
```

Response Codes

200	Request successful
201	Request to create or update object successful
202	The request was accepted, but the processing has not completed.
204	Request successful. Resource successfully deleted
400	Bad request.
401	Authentication failed.
403	Invalid permissions.
404	Resource not found.
409	Conflict.
412	Precondition failed.
413	Payload too large.
414	Request-URI too long
500	Internal server error.
502	Bad Gateway.

Jamf Pro - Pre-Stage Example

Get device scope for a specific Computer Prestage

https://xx.jamfcloud.com/api/v2/computer-prestages/id/scope

1. URL of computer-prestage endpoint and id of prestage

```
let prestageURLString = "https://xxx.jamfcloud.com/api/v2/computer-prestages/2/scope"

let prestageURL = URL(string: prestageURLString)!
```

virtual JNUC 2021

2. Create the URLRequest

```
var request = URLRequest(url: mobileURL)
request.httpMethod = “GET"
request.setValue("Bearer \(token)", forHTTPHeaderField: "Authorization")
request.setValue("application/json", forHTTPHeaderField: "Accept")
```

3. Create the Swift Model

```swift
struct ComputerPrestageCurrentScope: Codable {
    let prestageId: String
    let assignments: [ComputerPreStsgeScopeAssignment]
    let versionLock: Int
}

struct ComputerPreStsgeScopeAssignment: Codable {
    let serialNumber: String
    let assignmentDate: String
    let userAssigned: String
}
```

```json
{
    "prestageId":"string"
        "assignments":[
            {
                "serialNumber":"string"
                "assignmentDate":"2021-07-23T15:53:34.581Z"
                "userAssigned":"string"
            }
        ]
    "versionLock":0
}
```

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        if let data = data {
            let decoder = JSONDecoder()
            if let currentScope = try? JSONDecoder().decode(ComputerPrestageCurrentScope.self, from: data) {
                print(currentScope)
            }
        }
    }
}
task.resume()
```

Jamf Pro - Pre-Stage Example

Add device scope for a specific computer prestage

https://xx.jamfcloud.com/api/v2/computer-prestages/id/scope

1. URL of computer-prestage endpoint and id of prestage

```
let prestageURLString = "https://xxx.jamfcloud.com/api/v2/computer-prestages/2/scope"

let prestageURL = URL(string: prestageURLString)!
```

2. Create the URLRequest

```
request.httpMethod = "POST"
request.setValue("Bearer \(token)", forHTTPHeaderField: "Authorization")
request.setValue("application/json", forHTTPHeaderField: "Accept")
request.setValue("application/json", forHTTPHeaderField: "Content-Type")
```

3. Create the json for the httpBody

```
let json: [String: Any] = ["serialNumbers": [“C07FH1G1Q6NV" , “C87GH1K2Q6BA"],
                           "versionLock": depVersionLock]
```

3. Create the json for the httpBody

```
let json: [String: Any] = ["serialNumbers": [“C07FH1G1Q6NV" , “C87GH1K2Q6BA"],
                              "versionLock": depVersionLock]

If let jsonData = try? JSONSerialization.data(withJSONObject: json) {

   request.httpBody = jsonData

}
```

virtual JNUC 2021

4. URLSession, pass it the request and a closure to handle the response

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if (response as? HTTPURLResponse)?.statusCode != 200 {
        //handle error
    } else {
        print("We updated the prestage")
    }
}
task.resume()
```

Swift 5.5 - WWDC 2021

New Asynchronous APIs
Cleans up your code
Makes your code easier to read

https://developer.apple.com/videos/play/wwdc2021/10132/

Existing code

```
let task = URLSession.shared.dataTask(with: request) { (data, response, error) in
    if let error = error {
        //handle error
    } else if let response = response as? HTTPURLResponse {
        //handle error
    } else {
        if let data = data {








        }
    }
}
task.resume()
```

New try/await

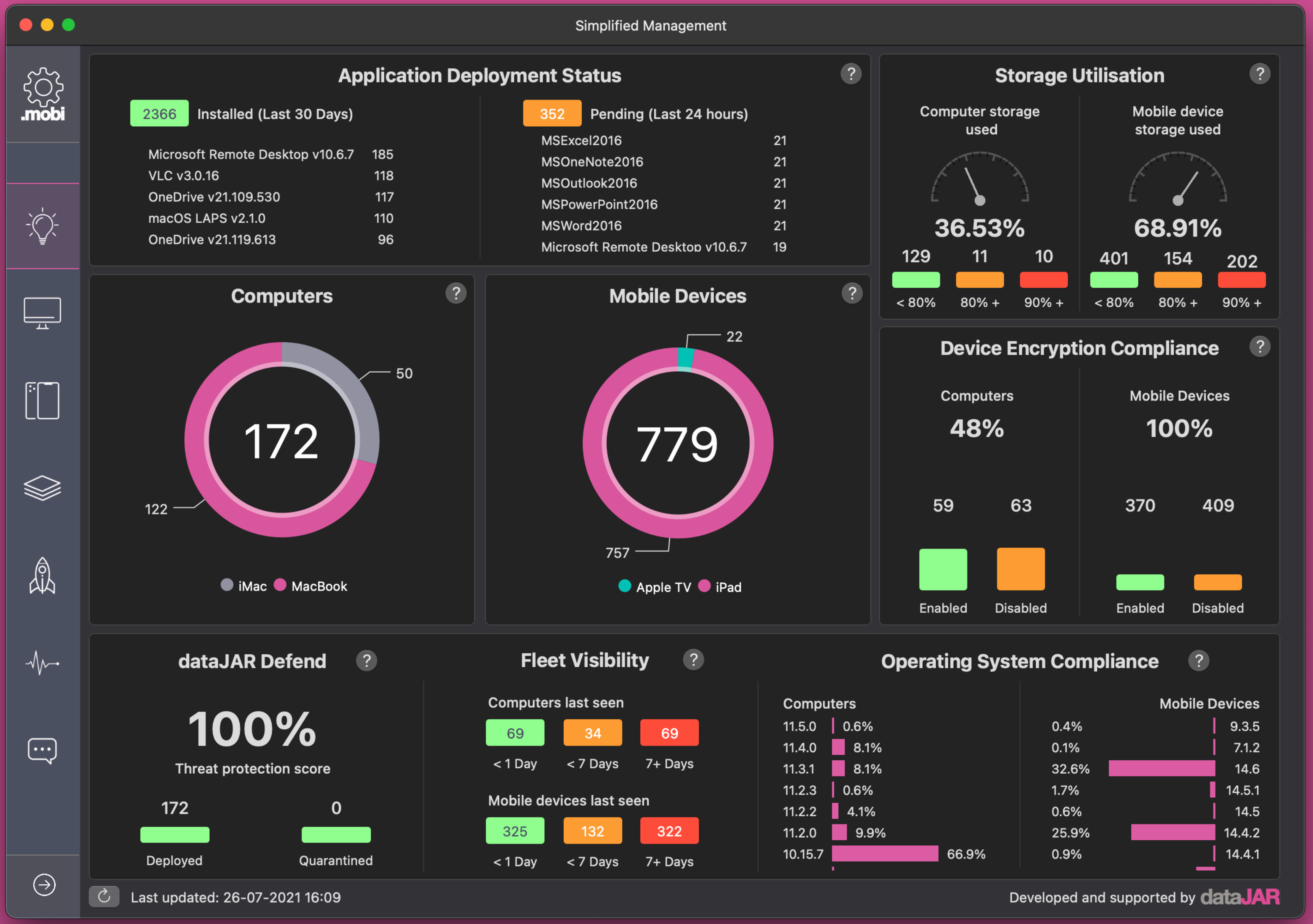
```
let (data, response) = try await URLSession.shared.data(for: request)
let decoder = JSONDecoder()
. . . . .
```

Where does dataJAR use Swift?

Native Apps

Command-line tools

Occasionally scripts


Simplified Management

Application Deployment Status

2366 Installed (Last 30 Days)

Microsoft Remote Desktop v10.6.7	185
VLC v3.0.16	118
OneDrive v21.109.530	117
macOS LAPS v2.1.0	110
OneDrive v21.119.613	96

352 Pending (Last 24 hours)

MSExcel2016	21
MSOneNote2016	21
MSOutlook2016	21
MSPowerPoint2016	21
MSWord2016	21
Microsoft Remote Desktop v10.6.7	19

Last updated: 26-07-2021 16:09

Developed and supported by dataJAR

virtual JNUC 2021

https://github.com/dataJAR/JNUC2021-JamfProSwift