readme.md for @apiclient.xyz/unifi

A comprehensive, unofficial TypeScript client for the UniFi ecosystem. Control your entire Ubiquiti infrastructure programmatically — Network devices, Protect cameras, Access doors, and Site Manager — all from a single, unified API. □□

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Installation

```
npm install @apiclient.xyz/unifi
# or
pnpm add @apiclient.xyz/unifi
# or
yarn add @apiclient.xyz/unifi
```

Features

□ Four UniFi APIs in One Package

API	Use Case	Authentication
UnifiController	Network devices, clients, VLANs, WLANs	API Key or Session

API	Use Case	Authentication
UnifiProtect	Cameras, NVR, motion events, recordings	API Key or Session + CSRF
UnifiAccess	Doors, users, NFC cards, access events	Bearer Token
UnifiAccount	Cloud Site Manager (ui.com)	API Key

□□ Developer-Friendly Design

- Full TypeScript support with comprehensive types
- Intuitive class-based resource management
- Async/await throughout
- Works with self-signed certificates (common on UniFi devices)

Quick Start

Network Controller (Local)

Manage your switches, access points, gateways, and connected clients:

```
import { UnifiController } from '@apiclient.xyz/unifi';

// Using API key (recommended - no login required)
const controller = new UnifiController({
  host: '192.168.1.1',
  apiKey: 'your-network-api-key',
  controllerType: 'unifi-os', // 'unifi-os' | 'udm-pro' | 'standalone'
  verifySsl: false, // Self-signed certs
});

// List all devices
const devices = await controller.deviceManager.listDevices();
for (const device of devices) {
  console.log(`${device.getDisplayName()} - ${device.isOnline() ? '□□' : '□□'}`);
}

// Get specific device types
const accessPoints = await controller.deviceManager.getAccessPoints();
```

```
const switches = await controller.deviceManager.getSwitches();
const gateways = await controller.deviceManager.getGateways();

// Manage connected clients
const clients = await controller.clientManager.listActiveClients();
const wirelessClients = await controller.clientManager.getWirelessClients();

// Find a client and block them
const troublemaker = await controller.clientManager.getClientByMac('aa:bb:cc:dd:ee:ff');
if (troublemaker) {
  await troublemaker.block();
}

// Get network configuration
const networks = await controller.getNetworks();
const wlans = await controller.getWlans();
const firewallRules = await controller.getFirewallRules();
const portForwards = await controller.getPortForwards();

// System info and health
const health = await controller.getHealth();
const systemInfo = await controller.getSystemInfo();
```

Protect NVR (Local)

Control your cameras, view motion events, manage recordings:

```
import { UnifiProtect } from '@apiclient.xyz/unifi';

const protect = new UnifiProtect({
  host: '192.168.1.1',
  apiKey: 'your-protect-api-key',
  verifySsl: false,
});

// Load camera data
await protect.refreshBootstrap();

// List all cameras
```

```
const cameras = await protect.cameraManager.listCameras();
for (const camera of cameras) {
  console.log(`□□${camera.name} - ${camera.isOnline() ? 'Online' : 'Offline'}`);
  if (camera.isDoorbell()) console.log('  □□Doorbell');
  if (camera.hasSmartDetect()) console.log('  □□Smart Detection enabled');
}

// Get cameras by status
const onlineCameras = await protect.cameraManager.getOnlineCameras();
const doorbells = await protect.cameraManager.getDoorbells();
const smartCameras = await protect.cameraManager.getSmartDetectCameras();

// Check recent motion
const recentMotion = await protect.cameraManager.getCamerasWithRecentMotion(300); // Last 5
min
const motionEvents = await protect.cameraManager.getAllMotionEvents({ limit: 50 });

// Control a camera
const frontDoor = await protect.cameraManager.getCameraByName('Front Door');
if (frontDoor) {
  await frontDoor.setRecordingMode('always'); // 'always' | 'detections' | 'never' |
'schedule'
  await frontDoor.setMicVolume(80);
  await frontDoor.restart();

  // Get RTSP stream URL
  const rtspUrl = frontDoor.getRtspUrl(0); // Channel 0 = highest quality
}

// NVR info
const nvrInfo = protect.getNvrInfo();
const storageInfo = protect.getStorageInfo();
const lights = protect.getLights();
const sensors = protect.getSensors();
```

Access Controller (Local)

Manage doors, users, credentials, and access events:

```typescript
import { UnifiAccess } from '@apiclient.xyz/unifi';

const access = new UnifiAccess({
  host: '192.168.1.1',
  token: 'your-bearer-token',
  verifySsl: false,
});

// List all doors
const doors = await access.doorManager.listDoors();
for (const door of doors) {
  console.log(`□□${door.getDisplayName()} - ${door.getStatus()}`);
}

// Control a door
const mainEntrance = await access.doorManager.getDoorByName('Main Entrance');
if (mainEntrance) {
  await mainEntrance.unlock();
  // Door auto-locks based on your Access settings

  console.log(`Locked: ${mainEntrance.isLocked()}`);
  console.log(`Open: ${mainEntrance.isOpen()}`);
}

// Manage users
const users = await access.getUsers();
const newUser = await access.createUser({
  first_name: 'John',
  last_name: 'Doe',
  email: 'john@example.com',
});

// Assign credentials
await access.setUserPin(newUser.id, '1234');
await access.assignNfcCard(newUser.id, 'card-token-here', 'Office Card');

// Grant/revoke door access
await access.grantAccess(newUser.id, mainEntrance.unique_id);
await access.revokeAccess(newUser.id, mainEntrance.unique_id);
```

```
// View access events
const recentEvents = await access.getRecentEvents(100);
const doorEvents = await access.getEvents({ doorId: mainEntrance.unique_id });

// Get policies and locations
const policies = await access.getPolicies();
const locations = await access.getLocations();
```

Site Manager (Cloud)

Manage multiple sites via the ui.com cloud API:

```
import { UnifiAccount } from '@apiclient.xyz/unifi';

// Cloud API uses api.ui.com
const account = new UnifiAccount({
  apiKey: 'your-cloud-api-key', // From ui.com
});

// List all sites
const sites = await account.siteManager.listSites();
for (const site of sites) {
  console.log(`□□${site.name} (${site.siteId})`);
}

// List all hosts (consoles)
const hosts = await account.hostManager.listHosts();

// Find specific site
const mainOffice = await account.siteManager.findSiteByName('Main Office');
```

Authentication Methods

API Keys (Recommended)

Generate API keys in your UniFi console settings. API keys don't expire and don't require login/logout:

```
// Network API key
const controller = new UnifiController({
  host: '192.168.1.1',
  apiKey: 'your-api-key',
  controllerType: 'unifi-os',
});

// Already authenticated - start using immediately
const devices = await controller.deviceManager.listDevices();
```

Session Authentication

For scenarios where API keys aren't available:

```
const controller = new UnifiController({
  host: '192.168.1.1',
  username: 'admin',
  password: 'your-password',
  controllerType: 'unifi-os',
});

// Must login first
await controller.login();

// Use the API
const devices = await controller.deviceManager.listDevices();

// Logout when done
await controller.logout();
```

Device Management

Working with UnifiDevice

```
const device = await controller.deviceManager.getDeviceByMac('aa:bb:cc:dd:ee:ff');

// Status checks
device.isOnline();      // Connected to controller?
device.isAccessPoint(); // Is this an AP?
device.isSwitch();      // Is this a switch?
device.isGateway();     // Is this a router/gateway?
device.hasUpgrade();    // Firmware update available?

// Actions
await device.restart();
await device.upgrade();
await device.rename('New Device Name');
await device.setLedOverride('off'); // 'default' | 'on' | 'off'

// Switch port configuration
await device.setPortConfig(1, {
  poe_mode: 'auto',
  name: 'Camera Port',
});

// Properties
device.ip;        // IP address
device.mac;       // MAC address
device.model;     // Model code (e.g., 'USW-24-POE')
device.version;   // Firmware version
device.uptime;    // Uptime in seconds
```

Working with UnifiClient

```
const client = await controller.clientManager.getClientByIp('192.168.1.100');

// Connection info
client.isWireless();       // WiFi or wired?
client.isGuest();          // Guest network?
client.getConnectionType(); // "Wireless (MySSID)" or "Wired (Port 5)"
client.getSignalQuality(); // "Excellent" | "Good" | "Fair" | "Poor"
client.getDataUsage();     // Total bytes (TX + RX)
```

```
// Actions
await client.block();      // Block from network
await client.unblock();    // Unblock
await client.reconnect();  // Kick and reconnect
await client.rename('Living Room TV');

// Properties
client.ip;       // IP address
client.mac;      // MAC address
client.hostname; // Device hostname
client.essid;    // WiFi network name
client.signal;   // Signal strength (dBm)
client.tx_bytes; // Upload bytes
client.rx_bytes; // Download bytes
```

Working with UnifiCamera

```
const camera = await protect.cameraManager.getCameraById('camera-id');

// Status
camera.isOnline();
camera.isDoorbell();
camera.hasSmartDetect();
camera.hasRecentMotion(60); // Motion in last 60 seconds?
camera.getTimeSinceLastMotion(); // Seconds since last motion

// Streaming
camera.getRtspUrl(0);            // High quality RTSP
camera.getHighQualityChannel();
camera.getMediumQualityChannel();
camera.getLowQualityChannel();

// Settings
await camera.setRecordingMode('detections');
await camera.setSmartDetectTypes(['person', 'vehicle']);
await camera.setMicVolume(50);
await camera.setSpeakerVolume(75);
```

```
await camera.rename('Garage Camera');
await camera.restart();
```

Working with UnifiDoor

```
const door = await access.doorManager.getDoorById('door-id');

// Status
door.isLocked();   // Lock engaged?
door.isOpen();     // Door physically open?
door.isClosed();   // Door physically closed?
door.getStatus();  // "Locked, Closed" etc.

// Actions
await door.unlock();
await door.lock();
await door.rename('Back Door');
await door.setAlias('Employee Entrance');
```

API Reference

Entry Point Classes

Class	Description	Manager Classes
`UnifiController`	Local Network Controller	`deviceManager`, `clientManager`
`UnifiProtect`	Local Protect NVR	`cameraManager`
`UnifiAccess`	Local Access Controller	`doorManager`
`UnifiAccount`	Cloud Site Manager	`siteManager`, `hostManager`

Resource Classes

Class	Represents
`UnifiDevice`	Network device (AP, switch, gateway)

Class	Represents
`UnifiClient`	Connected network client
`UnifiCamera`	Protect camera
`UnifiDoor`	Access door
`UnifiSite`	Site Manager site
`UnifiHost`	Site Manager host/console

Controller Types

Type	Description
`unifi-os`	UniFi OS consoles (UDM, UDM Pro, Cloud Key Gen2+)
`udm-pro`	Alias for unifi-os
`standalone`	Standalone software controller

SSL/TLS Handling

UniFi devices typically use self-signed certificates. Set `verifySsl: false` to allow connections:

```
const controller = new UnifiController({
  host: '192.168.1.1',
  apiKey: 'key',
  verifySsl: false, // Allow self-signed certs
});
```

For production environments with proper certificates, set `verifySsl: true`.

Environment Variables Example

Create a `.env` or use your preferred env management:

```
# Network Controller
UNIFI_CONSOLE_IP=192.168.1.1
UNIFI_NETWORK_API_KEY=your-network-key
```

```
# Protect
UNIFI_PROTECT_API_KEY=your-protect-key

# Access
UNIFI_ACCESS_HOST=192.168.1.1
UNIFI_ACCESS_TOKEN=your-bearer-token

# Site Manager (Cloud)
UNIFI_CLOUD_API_KEY=your-cloud-key
```

TypeScript Support

This package is written in TypeScript and exports comprehensive types:

```
import {
  // Entry points
  UnifiController,
  UnifiProtect,
  UnifiAccess,
  UnifiAccount,

  // Resources
  UnifiDevice,
  UnifiClient,
  UnifiCamera,
  UnifiDoor,
  UnifiSite,
  UnifiHost,

  // Managers
  DeviceManager,
  ClientManager,
  CameraManager,
  DoorManager,
  SiteManager,
  HostManager,
```

```
  // Interfaces
  INetworkDevice,
  INetworkClient,
  IProtectCamera,
  IAccessDoor,
  // ... and many more
} from '@apiclient.xyz/unifi';
```

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #6
Created 2026-03-28 10:48:22 UTC by foss.global Team
Updated 2026-03-28 12:14:18 UTC by foss.global Team