Portal Login

Endpoints Summary

Method	Path	Swagger
`POST`	`/portal_login/`	Swagger ↗

> The portal_login API provides functionality for managing captive portal authentication flows, allowing access points to redirect users through a centralized authentication system. This endpoint is primarily used in WiFi network environments where user authentication and session management are required before granting internet access.

Base URL: `https://gate.zequenze.com/api/v1`

Authentication: All endpoints require a Bearer token:

```
Authorization: Bearer <your-api-token>
```

Overview

The portal_login API category is designed for WiFi captive portal systems that need to authenticate users before granting network access. This API handles the critical handoff between access points, controllers, and the authentication platform.

Key Concepts:

- **Captive Portal Flow**: When users connect to WiFi, they're redirected to an authentication page before accessing the internet
- **Session Management**: The API creates authenticated sessions with configurable timeout periods
- **Multi-device Support**: Tracks individual devices by MAC address and assigns IP addresses

- **Controller Integration**: Works with network controllers to manage access point configurations

Common Integration Scenarios:

- Hotel WiFi systems requiring room number authentication
- Corporate guest networks with email-based login
- Public WiFi hotspots with terms of service acceptance
- Retail locations with social media login requirements

The portal_login endpoint serves as the authentication gateway, processing user credentials and device information to establish authorized network sessions.

Endpoints

POST /portal_login/

Description: Executes a portal login operation to authenticate a user device and establish a network session. This endpoint processes authentication data from captive portals and creates authorized sessions with the network controller, enabling internet access for the authenticated device.

Use Cases:

- Authenticate hotel guests using room credentials before granting WiFi access
- Process corporate visitor registration and create temporary network access
- Handle social media login flows for public WiFi hotspots
- Validate terms of service acceptance for retail location WiFi

Full URL Example:

```
https://gate.zequenze.com/api/v1/portal_login/
```

Request Body Parameters:

Parameter	Type	Required	Description
data	string	Yes	JSON string containing authentication and device information

Data Object Structure: The `data` parameter should contain a JSON string with the following authentication information:

Field	Type	Required	Description
uid	string	Yes	Unique identifier for the user (email, username, room number, etc.)
ap_mac	string	Yes	MAC address of the access point handling the connection
mac	string	Yes	MAC address of the user's device
ip	string	Yes	IP address assigned to the user's device
original_url	string	No	The original URL the user was trying to access
controller_address	string	No	IP address of the network controller
session_timeout	integer	No	Maximum session duration in seconds
idle_timeout	integer	No	Idle timeout period in seconds

cURL Example:

```
curl -X POST "https://gate.zequenze.com/api/v1/portal_login/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "data":
"{\"uid\":\"guest@hotel.com\",\"ap_mac\":\"aa:bb:cc:dd:ee:ff\",\"mac\":\"11:22:33:44:55:66\",\
"ip\":\"192.168.1.100\",\"original_url\":\"https://www.google.com\",\"controller_address\":\"1
92.168.1.1\",\"session_timeout\":3600,\"idle_timeout\":1800}"
  }'
```

Example Request Body:

```
{
  "data":
"{\"uid\":\"room_401\",\"ap_mac\":\"aa:bb:cc:dd:ee:ff\",\"mac\":\"11:22:33:44:55:66\",\"ip\":\
"192.168.1.100\",\"original_url\":\"https://www.example.com\",\"controller_address\":\"192.168
.1.1\",\"session_timeout\":7200,\"idle_timeout\":3600}"
}
```

Example Response:

```
{
  "page_id": 12345,
  "original_url": "https://www.example.com",
  "ap_mac": "aa:bb:cc:dd:ee:ff",
  "mac": "11:22:33:44:55:66",
  "ip": "192.168.1.100",
  "uid": "room_401",
  "controller_address": "192.168.1.1",
  "session_timeout": 7200,
  "idle_timeout": 3600
}
```


Response Fields:

Field	Type	Description
page_id	integer	Unique identifier for this portal session
original_url	string	The URL the user was originally trying to access
ap_mac	string	MAC address of the access point
mac	string	MAC address of the authenticated device
ip	string	IP address assigned to the device
uid	string	User identifier used for authentication
controller_address	string	Network controller IP address
session_timeout	integer	Maximum session duration in seconds
idle_timeout	integer	Idle timeout duration in seconds

Response Codes:

Status	Description
201	Success - Portal login completed successfully
400	Bad Request - Invalid data format or missing required fields
401	Unauthorized - Invalid or missing API token
422	Unprocessable Entity - Authentication failed or invalid user credentials
500	Internal Server Error - Server-side processing error

Common Use Cases

Use Case 1: Hotel Guest Authentication

Authenticate hotel guests using their room number and last name. The portal captures the guest's device information and creates a session that expires at checkout time.

Implementation: Use the guest's room number as the `uid` and set `session_timeout` based on their stay duration.

Use Case 2: Corporate Guest Network

Process visitor registration where guests provide their email address and accept terms of service before accessing the corporate guest WiFi.

Implementation: Use the guest's email as the `uid` and include the original destination URL to redirect them after authentication.

Use Case 3: Public WiFi with Social Login

Handle authentication flows where users log in through social media platforms (Facebook, Google) to access free WiFi in retail locations.

Implementation: Use the social media user ID as the `uid` and set appropriate session timeouts for fair usage policies.

Use Case 4: Event WiFi Management

Manage WiFi access for conference attendees using registration codes or badge scan data, with sessions that expire at the end of each day.

Implementation: Use attendee registration codes as the `uid` with daily session timeouts and controller integration for bandwidth management.

Best Practices

Data Format:

- Always ensure the `data` parameter contains properly escaped JSON strings
- Validate MAC addresses are in the correct format (aa:bb:cc:dd:ee:ff) before sending
- Include session timeouts to prevent indefinite network access

Error Handling:

- Implement retry logic for temporary network failures (5xx errors)
- Validate user credentials before calling the API to minimize 422 responses
- Log authentication attempts for security monitoring and troubleshooting

Security Considerations:

- Never log or expose the full API token in client-side code
- Use HTTPS for all API communications to protect authentication data
- Implement rate limiting to prevent abuse of the authentication endpoint

Performance Tips:

- Cache successful authentication responses to avoid duplicate API calls
- Set appropriate session timeouts to balance user experience with resource usage
- Monitor API response times and implement timeout handling in your application

Session Management:

- Store the returned `page_id` for session tracking and potential logout operations
- Respect the configured timeout values and handle session expiration gracefully
- Consider implementing session renewal for long-term users

Revision #4
Created 2026-02-04 05:15:33 UTC by ipena@zequenze.com
Updated 2026-02-11 03:20:04 UTC by ipena@zequenze.com