Device App User Authenticate

Endpoints Summary

Method	Path	Swagger
`POST`	`/device_app_user_authenticate/`	Swagger ↗

> The Device App User Authentication endpoint provides secure credential validation for user authentication within device applications. This endpoint validates username and password combinations and is typically used during login flows for mobile or desktop applications that need to authenticate users against the GATE system.

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

Authentication: All endpoints require a Bearer token:

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

Overview

The Device App User Authenticate API category provides a single, focused endpoint for validating user credentials in device applications. This endpoint is designed to handle authentication requests from mobile apps, desktop applications, or IoT devices that need to verify user credentials against the GATE authentication system.

This endpoint accepts user credentials (username and password) and validates them against the system's user database. It's commonly used in scenarios where applications need to authenticate users before granting access to protected resources or functionality. The endpoint follows secure authentication practices and should be used over HTTPS connections to protect credential transmission.

Key features include:

- Secure credential validation
- Support for username/password authentication
- Integration with device applications
- RESTful API design for easy integration

Endpoints

POST /device_app_user_authenticate/

Description: Validates user credentials by accepting a username and password combination and returning authentication results. This endpoint is specifically designed for device applications that need to authenticate users before granting access to protected features or data.

Use Cases:

- Mobile app login screens where users enter credentials
- Desktop application authentication flows
- IoT device user validation before accessing device features
- Single sign-on integration for device applications
- User credential verification in embedded systems

Full URL Example:

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

Parameters:

Parameter	Type	In	Required	Description
data	string	body	Yes	JSON string containing user authentication credentials (username and password)

Request Body Schema:

```
{
  "username": "string (required)",
  "password": "string (required)"
}
```

cURL Example:

```
curl -X POST "https://gate.zequenze.com/api/v1/device_app_user_authenticate/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "username": "john.doe@company.com",
    "password": "SecurePassword123!"
  }'
```

Example Request Body:

```
{
  "username": "john.doe@company.com",
  "password": "SecurePassword123!"
}
```

Example Response (Success):

```
{
  "username": "john.doe@company.com",
  "authenticated": true,
  "user_id": 12345,
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_expires": "2024-01-15T18:30:00Z",
  "user_profile": {
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@company.com",
    "role": "device_user",
    "permissions": ["device_access", "data_view"]
  }
}
```

Example Response (Failed Authentication):

```
{
  "username": "john.doe@company.com",
  "authenticated": false,
  "error": "Invalid credentials",
  "error_code": "AUTH_FAILED"
```

```
}
```

Response Codes:

Status	Description
201	Created - Authentication successful, user credentials validated
400	Bad Request - Missing or invalid username/password format
401	Unauthorized - Invalid API token or authentication failed
422	Unprocessable Entity - Valid request format but authentication failed
429	Too Many Requests - Rate limit exceeded for authentication attempts
500	Internal Server Error - Server error during authentication process

---

Common Use Cases

Use Case 1: Mobile App Login

When developing a mobile application that requires user authentication, use this endpoint during the login flow. After users enter their credentials, send them to this endpoint for validation and receive an access token for subsequent API calls.

Use Case 2: IoT Device User Verification

For IoT devices with user interfaces, this endpoint can verify that users have valid credentials before allowing access to device configuration or sensitive data.

Use Case 3: Desktop Application Authentication

Desktop applications can integrate this endpoint into their login screens to authenticate users against the centralized GATE user database, enabling consistent user management across platforms.

Use Case 4: Kiosk or Embedded System Login

Public kiosks or embedded systems can use this endpoint to authenticate users before providing access to personalized content or restricted functionality.

Use Case 5: Third-Party Application Integration

External applications integrating with the GATE system can use this endpoint to authenticate users and maintain consistent user sessions across multiple platforms.

Best Practices

- **Secure Transmission**: Always use HTTPS when transmitting credentials to protect sensitive user data during authentication requests.
- **Error Handling**: Implement proper error handling for different response codes. Don't expose detailed error messages to end users that might reveal system information.
- **Rate Limiting**: Implement client-side rate limiting to prevent excessive authentication attempts, which could trigger server-side rate limits or security measures.
- **Token Management**: Store received access tokens securely on the device and implement proper token refresh mechanisms when tokens expire.
- **Input Validation**: Validate username and password format on the client side before sending requests to reduce unnecessary API calls and improve user experience.
- **Credential Security**: Never log or store user passwords in plain text. Hash sensitive data and follow secure coding practices for credential handling.
- **Session Management**: Implement proper session timeout handling and allow users to securely log out, invalidating their access tokens.
- **Retry Logic**: Implement exponential backoff for failed authentication attempts to handle temporary network issues gracefully while respecting rate limits.

Revision #4
Created 2026-02-04 05:14:13 UTC by ipena@zequenze.com
Updated 2026-02-11 03:16:42 UTC by ipena@zequenze.com