Device App User Update

Endpoints Summary

Method	Path	Swagger
POST	/device_app_user_update/	Swagger ↗

> The device_app_user_update API endpoint allows mobile applications and device management systems to update user information associated with devices. This endpoint is essential for maintaining accurate user profiles, device associations, and subscriber data in mobile network environments.

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_update API category provides functionality for updating comprehensive user information linked to mobile devices and network subscribers. This endpoint is particularly valuable for:

- **Mobile Network Operators (MNOs)** managing subscriber data and device associations
- **Device Management Platforms** updating user profiles and device bindings
- **Mobile Applications** syncing user credentials and device identifiers
- **IoT Systems** maintaining device-to-user mappings

The endpoint supports updating critical identifiers including device numbers (MSISDN), hardware identifiers (IMEI, IMSI, ICCID), user credentials, and service classifications. This comprehensive approach ensures that all aspects of the user-device relationship can be maintained accurately across the system.

Common scenarios include user onboarding, device transfers between users, credential updates, and service class modifications. The endpoint is designed to handle both individual field updates and bulk profile modifications efficiently.

Endpoints

POST /device_app_user_update/

Description: Updates comprehensive user information associated with a device, including network identifiers, hardware identifiers, user credentials, and service classifications. This endpoint is essential for maintaining accurate user-device relationships in mobile networks and device management systems.

Use Cases:

- Update user credentials when a subscriber changes their login information
- Associate a new device with an existing user account during device upgrades
- Modify service class when a user changes their subscription plan
- Update network identifiers during SIM card replacements or number porting

Full URL Example:

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

Parameters:

Parameter	Type	In	Required	Description
data	string	body	Yes	JSON string containing user and device information to update

Request Body Fields:

Field	Type	Required	Description
number	string	No	Device number (MSISDN) - the phone number associated with the device
generic_id	string	No	Generic unique identifier from the operating system or app store

Field	Type	Required	Description
imei	string	No	International Mobile Equipment Identity - unique device hardware identifier
imsi	string	No	International Mobile Subscriber Identity - unique SIM card identifier
iccid	string	No	Integrated Circuit Card Identifier - unique SIM card serial number
username	string	No	User's login username for authentication
password	string	No	User's password (should be properly hashed)
user_external_id	string	No	External system identifier for the user
user_class	string	No	User's class of service (e.g., "premium", "standard", "basic")

cURL Example:

```
curl -X POST "https://gate.zequenze.com/api/v1/device_app_user_update/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "data":
"{\"number\":\"+1234567890\",\"username\":\"john.doe\",\"user_class\":\"premium\",\"imei\":\"1
23456789012345\",\"user_external_id\":\"ext_12345\"}"
  }'
```

Example Request Body:

```
{
  "data":
"{\"number\":\"+1234567890\",\"generic_id\":\"com.example.app.user123\",\"imei\":\"12345678901
2345\",\"imsi\":\"310260123456789\",\"iccid\":\"89014103211118510720\",\"username\":\"john.doe
\",\"password\":\"$2a$10$N9qo8uLOickgx2ZMRZoMye\",\"user_external_id\":\"ext_12345\",\"user_cl
ass\":\"premium\"}"
}
```

Example Response (201 Created):

```
{
  "number": "+1234567890",
  "generic_id": "com.example.app.user123",
  "imei": "123456789012345",
  "imsi": "310260123456789",
  "iccid": "89014103211118510720",
  "username": "john.doe",
  "password": "$2a$10$N9qo8uLOickgx2ZMRZoMye",
  "user_external_id": "ext_12345",
  "user_class": "premium",
  "updated_at": "2024-01-15T10:30:00Z",
  "status": "success"
}
```


Response Codes:

Status	Description
201	Created - User information successfully updated
400	Bad Request - Invalid data format or missing required fields
401	Unauthorized - Invalid or missing authentication token
422	Unprocessable Entity - Data validation errors
500	Internal Server Error - Server-side processing error

Common Use Cases

Use Case 1: User Credential Update

When a user changes their login credentials, update their username and password while maintaining device associations. This ensures continued access while preserving all device-specific identifiers.

Use Case 2: Device Upgrade

During a device upgrade, transfer user information to the new device by updating the IMEI and other hardware identifiers while keeping the user credentials and service class intact.

Use Case 3: Service Plan Change

When a user upgrades or downgrades their service plan, update the user_class field to reflect their new subscription tier, enabling appropriate service provisioning.

Use Case 4: SIM Card Replacement

After replacing a damaged or lost SIM card, update the IMSI and ICCID values while maintaining all other user information and device associations.

Use Case 5: External System Integration

Synchronize user data with external CRM or billing systems by updating the user_external_id field, enabling cross-platform user tracking and management.

Best Practices

- **Data Validation**: Always validate device identifiers (IMEI, IMSI, ICCID) against industry standards before submission to prevent data corruption
- **Password Security**: Ensure passwords are properly hashed using strong algorithms (bcrypt, Argon2) before sending to the API
- **Partial Updates**: Only include fields that need updating in the request to minimize data transfer and processing time
- **Error Handling**: Implement robust error handling to manage validation failures and network timeouts gracefully
- **Idempotency**: Design your integration to handle duplicate requests safely, as network issues may cause request retries
- **Rate Limiting**: Implement appropriate request throttling to avoid overwhelming the API service
- **Data Privacy**: Ensure compliance with data protection regulations when handling user credentials and personal identifiers
- **Logging**: Maintain audit logs of user updates for security and compliance purposes, but avoid logging sensitive data like passwords

---

Revision #4
Created 2026-02-04 05:14:24 UTC by ipena@zequenze.com
Updated 2026-02-11 03:17:15 UTC by ipena@zequenze.com