API Authentication Token

This API endpoint allows users to authenticate and obtain a Bearer token required for all subsequent API requests. The token must be included in the Authorization header of every API call to access protected endpoints.

Endpoint Information

`POST` **`/api/auth-login`**

Authentication Flow

How Authentication Works

1. **Step 1:** Send login credentials to `/api/auth-login`
2. **Step 2:** Receive Bearer token in response
3. **Step 3:** Include token in Authorization header for all API requests
4. **Step 4:** Token remains valid until logout or expiration

Request Parameters

All parameters should be sent in the request body as JSON or form data.

Parameter	Type	Required	Description
email	String	REQUIRED	User's email address for authentication
password	String	REQUIRED	User's password for authentication

Request Example

JSON Request Body

```
{ "email": "user@example.com", // Required: User's email address "password": "your_password" // Required:
User's password }
```

cURL Example

```
curl -X POST "https://your-domain.com/api/auth-login" \ -H "Content-Type: application/json" \ -d '{ "email":
"user@example.com", "password": "your_password" }'
```

Response Format

Success Response

HTTP Status: 200 OK

```
{ "access_token": "1|eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...", // Bearer token for API authentication "user":
{ // Authenticated user information "id": 123, "name": "John Doe", "email": "user@example.com",
"profile_type": 2, // User role/permission level "branch_id": 1, "isp_id": 1, "created_at": "2024-01-
15T10:30:00.000000Z", "updated_at": "2024-01-15T14:30:00.000000Z" } }
```

Error Responses

Authentication Errors

HTTP Status: 401 Unauthorized

Invalid Credentials

```
{ "message": "Invalid Credentials" }
```

Missing Required Fields

```
{ "message": "The email field is required.", "errors": { "email": ["The email field is required."],
"password": ["The password field is required."] } }
```

Account Disabled

```
{ "message": "Account is disabled or suspended" }
```

Using the Bearer Token

Once you receive the authentication token, you must include it in the Authorization header of all subsequent API requests:

Authorization Header Format

```
Authorization: Bearer 1|eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
```

Example API Request with Token

```
curl -X POST "https://your-domain.com/api/v1/subscriber/activation" \ -H "Content-Type: application/json" \ -
H "Authorization: Bearer 1|eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..." \ -d '{ "subscriber_id": 12345 }'
```

Token Management

Important Token Guidelines

- **Security:** Never expose your Bearer token in client-side code or logs
- **Storage:** Store tokens securely (encrypted storage, environment variables)
- **Token Format:** Tokens are prefixed with an ID (e.g., "1|actualtoken...")
- **Revocation:** Tokens can be revoked by calling the logout endpoint
- **Scope:** Each token is tied to a specific user account and permissions

Token Validation Errors

If your token is invalid, expired, or missing, API endpoints will return:

```
{ "message": "Unauthenticated." } // HTTP Status: 401 Unauthorized
```

Integration Best Practices

1. **Implement Token Refresh:** Automatically refresh tokens before they expire
2. **Handle 401 Errors:** Redirect to login when receiving unauthorized responses
3. **Secure Storage:** Use secure storage mechanisms for token persistence
4. **Error Handling:** Implement comprehensive error handling for authentication failures
5. **Logout Functionality:** Provide users with ability to invalidate tokens
6. **Rate Limiting:** Respect authentication rate limits to avoid account lockouts

User Profile Types

The authentication response includes a `profile_type` field that determines user permissions:

Profile Type	Role	Permissions
1	Admin	Full system access and management
2	Salesperson	Subscriber management and transactions
3	Reseller	Limited subscriber management within hierarchy
4	Sub-Reseller	Restricted subscriber operations
5	Retailer	Basic subscriber operations

Testing

Authentication Testing Checklist

- Test with valid credentials to ensure token generation
- Test with invalid credentials to verify error handling
- Test with missing fields to validate input requirements
- Test token usage in subsequent API calls
- Test token expiration and refresh scenarios
- Verify proper error responses for unauthorized requests

Add Subscriber Balance API

This API endpoint allows authorized users (salespersons, resellers, sub-resellers, retailers) to add balance to a subscriber's account. The API includes comprehensive validation, permission checks, and duplicate payment prevention.

Endpoint Information

`POST` `/api/v1/subscriber/payments/add-balance`

Authentication

This API requires Bearer token authentication. You must first obtain a token from the `/api/auth-login` endpoint and include it in the Authorization header of your request.

Required Authentication Steps

1. **Step 1:** Get Bearer token from `POST /api/auth-login` with email/password
2. **Step 2:** Include token in Authorization header: `Authorization: Bearer {token}`
3. **Step 3:** User must be a salesperson, reseller, sub-reseller, or retailer with appropriate permissions

Permission Requirements:

- Users with profile_type >= 3 (resellers, sub-resellers, retailers) can only update subscribers they have permission to manage
- The system validates user permissions before processing the payment

Request Parameters

All parameters should be sent in the request body as JSON or form data.

Parameter	Type	Required	Description
subscriber_id	Integer	REQUIRED	The unique ID of the subscriber to add balance to
payment_amount	Numeric	REQUIRED	The amount to add to subscriber's balance (must be a valid number)
payment_method	Integer	OPTIONAL	Payment method ID (defaults to 1 for cash payment)
payment_note	String	OPTIONAL	Additional note for the payment (defaults to "Salesperson Balance Topup")
salesperson_balance_cut_status	Boolean/Integer	OPTIONAL	Whether to deduct the amount from salesperson's balance

Payment Methods

Understanding payment methods is crucial for proper API integration. Each method has specific behavior and requirements.

Method ID	Payment Type	Description	When Used
1	Cash	Default payment method for manual transactions	When no payment_method specified (default)
4	Subscriber Balance	Internal balance transfer (not applicable for adding balance)	Used in other operations
6	bKash	Mobile financial service payment	Popular choice for mobile payments

Request Example

JSON Request Body

```
{ "subscriber_id": 12345, // Required: ID of subscriber to add balance to "payment_amount": 500.00, // Required: Amount to add to subscriber's balance "payment_method": 1, // Optional: Payment method ID (1=Cash, 6=bKash, etc.) "payment_note": "Monthly balance topup", // Optional: Note for the payment transaction "salesperson_balance_cut_status": 1 // Optional: 1=Deduct from salesperson balance, 0=Don't deduct }
```

cURL Example

```
curl -X POST "https://your-domain.com/api/v1/subscriber/payments/add-balance" \ -H "Content-Type: application/json" \ -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..." \ -d '{ "subscriber_id": 12345, // Required: ID of subscriber to add balance to "payment_amount": 500.00, // Required: Amount to add to subscriber balance "payment_method": 1, // Optional: 1=Cash payment method "payment_note": "Monthly balance topup", // Optional: Description for the payment "salesperson_balance_cut_status": 1 // Optional: Deduct from salesperson balance }'
```

Response Format

All responses are returned in JSON format with a consistent structure.

Success Response

HTTP Status: 200 OK

```
{ "status": "success", // Response status: "success" or "error" "message": "Balance added successfully", // Human-readable success message "data": { // Payment transaction details "payment_id": 67890, // Unique payment ID for reference "subscriber_id": 12345, // ID of subscriber who received balance "amount_added": 500.00, // Amount successfully added to balance "new_balance": 1500.00, // Subscriber's new total balance after addition "payment_date": "2024-01-15 14:30:25" // Timestamp when payment was processed } }
```

Error Responses

Validation Errors

HTTP Status: 200 OK (with error status in response body)

Missing Required Fields

```
{ "status": "error", "message": "The subscriber id field is required.The payment amount field is required." } // status: Response status indicating failure // message: Concatenated validation error messages
```

Invalid Payment Amount

```
{ "status": "error", "message": "The payment amount must be a number." } // status: Response status indicating failure // message: Error when payment_amount is not numeric
```

Business Logic Errors

Subscriber Not Found

```
{ "status": "error", "message": "Subscriber Not Found" } // status: Response status indicating failure // message: Error when subscriber_id doesn't exist in database
```

Salesperson Not Found

```
{ "status": "error", "message": "Subscriber Salesperson Not Found" } // status: Response status indicating failure // message: Error when subscriber's assigned salesperson is missing
```

Insufficient Permissions

```
{ "status": "error", "message": "Oops! Insufficient Permission" } // status: Response status indicating failure // message: Error when user lacks permission to manage this subscriber
```

Duplicate Payment Prevention

```
{ "status": "error", "message": "Too Frequent Payments! Please Wait 1 Minute & Try Again." } // status: Response status indicating failure // message: Error when same amount added within 60 seconds
```

Important Notes

Duplicate Payment Prevention

The API prevents duplicate payments by checking if:

- The same payment amount was added within the last 60 seconds
- If a duplicate is detected, the user must wait 1 minute before trying again

Permission System

The API enforces strict permission checks:

- Resellers can only add balance to their own subscribers and their sub-resellers'/retailers' subscribers
- Sub-resellers and retailers have similar hierarchical permission restrictions
- The system validates these permissions before processing any payment

Integration Tips

1. **Always validate subscriber_id:** Ensure the subscriber exists and you have permission to manage them
2. **Handle duplicate prevention:** Implement retry logic with appropriate delays if you receive the "Too Frequent Payments" error
3. **Check response status:** Always check the "status" field in the response to determine if the operation was successful
4. **Store payment references:** Save the returned payment_id for your records and future reference
5. **Implement proper error handling:** Handle all possible error scenarios gracefully in your application

Testing

Before integrating this API into your production system:

- Test with valid subscriber IDs that you have permission to manage
- Test with invalid subscriber IDs to ensure proper error handling
- Test the duplicate payment prevention by making consecutive requests
- Verify that permission checks work correctly for your user type

Subscriber Activation API

This API endpoint allows authorized users to activate subscribers with package assignments, billing generation, payment processing, and RADIUS integration. The API handles complex business logic including invoice generation, payment validation, profit distribution, and subscriber profile updates.

Endpoint Information

POST `/api/v1/subscriber/activation`

Authentication

This API requires Bearer token authentication. You must first obtain a token from the `/api/auth-login` endpoint and include it in the Authorization header of your request.

Required Authentication Steps

1. **Step 1:** Get Bearer token from `POST /api/auth-login` with email/password
2. **Step 2:** Include token in Authorization header: `Authorization: Bearer {token}`
3. **Step 3:** User must be a salesperson, reseller, sub-reseller, or retailer with appropriate permissions

Important Business Rules:

- Only authorized salespersons can activate subscribers assigned to them
- Package accounting validation is performed before activation
- Duplicate invoice prevention: Only one due invoice per hour is allowed
- Subscriber discount cannot exceed salesperson profit (except for admin users)

Request Parameters

All parameters should be sent in the request body as JSON or form data.

Parameter	Type	Required	Description
subscriber_id	Integer	REQUIRED	The unique ID of the subscriber to activate
payment_amount	Numeric	OPTIONAL	Payment amount for activation (if paying during activation)
payment_method	Integer	OPTIONAL	Payment method ID (defaults to 1 for cash, 6 for bKash when payment_amount provided)
cut_subscriber_balance	Boolean/Integer	OPTIONAL	Set to 1 to deduct activation fee from subscriber's existing balance (payment_method becomes 4)

Payment Methods

Understanding payment methods is crucial for proper API integration. Each method has specific behavior and requirements.

Method ID	Payment Type	Description	When Used
1	Cash	Default payment method for manual transactions	When no payment_method specified
4	Subscriber Balance	Deduct from subscriber's existing balance	Automatically set when cut_subscriber_balance = 1
6	bKash	Mobile financial service payment	Default when payment_amount is provided

Activation Process Flow

Step 1: Validation

- Validate subscriber existence and associated salesperson/package data
- Check for existing due invoices (within last hour)
- Verify package accounting permissions for the salesperson
- Validate package accounting rules

Step 2: Payment Processing (if applicable)

- Process payment if payment_amount is provided
- Create ledger entries for payment tracking
- Validate subscriber balance if cut_subscriber_balance is enabled

Step 3: Billing & Invoice Generation

- Calculate package fees, extra fees, and subscriber discounts
- Generate invoice with calculated amounts
- Apply subscriber discounts (cannot exceed salesperson profit)

Step 4: Activation & RADIUS Integration

- Process payment and update invoice status
- Activate subscriber in RADIUS system
- Distribute profits to reseller hierarchy
- Update subscriber profile with new package and expiration

Request Examples

Basic Activation (Generate Invoice Only)

```
{ "subscriber_id": 12345 // Required: ID of subscriber to activate }
```

Activation with Payment

```
{ "subscriber_id": 12345, // Required: ID of subscriber to activate "payment_amount": 1500.00, // Optional: Amount to pay during activation "payment_method": 6 // Optional: 6 = bKash (default when payment_amount provided) }
```

Activation Using Subscriber Balance

```
{ "subscriber_id": 12345, // Required: ID of subscriber to activate "cut_subscriber_balance": 1 // Optional: 1 = Use subscriber's existing balance (payment_method becomes 4) }
```

cURL Example

```
curl -X POST "https://your-domain.com/api/v1/subscriber/activation" \ -H "Content-Type: application/json" \ -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..." \ -d '{ "subscriber_id": 12345, // Required: ID of subscriber to activate "payment_amount": 1500.00, // Optional: Amount to pay during activation "payment_method": 6 // Optional: 6 = bKash payment method }'
```

Response Format

Success Response

HTTP Status: 200 OK

```
{ "status": "success", // Response status: "success" or "error" "message": "Subscriber Activated Successfully.", // Human-readable success message "subscriber_id": 12345, // ID of the activated subscriber "subscriber_username": "john_doe", // Username of the activated subscriber "invoice_data": { // Generated invoice information "id": 67890, // Unique invoice ID for reference "billing_total_amount": 1500.00, // Total invoice amount "billing_due_amount": 0.00, // Remaining due amount (0 if fully paid) "activation_status": 1, // 1 = Activated, 0 = Not activated "invoice_status": 1 // Invoice status (1 = Paid, 6 = Due) }, "subscriber_data": { // Updated subscriber profile information "id": 12345, // Subscriber ID "username": "john_doe", // Subscriber username "profile_status": 2, // 2 = Active profile "package_id": 15, // Assigned package ID "expiration_date": "2024-02-15 14:30:25", // New package expiration date "last_activation_time": "2024-01-15 14:30:25" // Timestamp of activation } }
```

Error Responses

Validation Errors

HTTP Status: 422 Unprocessable Entity

Missing Required Fields

```
{ "status": "error", "message": ["The subscriber id field is required."] } // status: Response status indicating failure // message: Array of validation error messages
```

Business Logic Errors

Subscriber Not Found

```
{ "status": "error", "message": "Subscriber Not Found" } // status: Response status indicating failure // message: Error message when subscriber_id doesn't exist
```

Invalid Salesperson or Package Data

```
{ "status": "error", "message": "Invalid Salesperson or Package Data" } // status: Response status indicating failure // message: Error when salesperson/package validation fails
```

Existing Due Invoice

```
{ "status": "error", "message": "Due Invoice Already Exist (Wait 1 Hour to Generate New One Or Pay On Due Invoice)", "invoice_id": 67890, "due_amount": 1500.00, "subscriber_id": 12345 } // status: Response status indicating failure // message: Error message with instructions // invoice_id: ID of existing due invoice // due_amount: Amount due on existing invoice // subscriber_id: Subscriber ID for reference
```

Package Validation Failed

```
{ "status": "error", "message": "Package Validation Failed" } // status: Response status indicating failure // message: Error when package accounting validation fails
```

Excessive Subscriber Discount

```
{ "status": "error", "message": "Too Much Subscriber Discount" } // status: Response status indicating failure // message: Error when discount exceeds salesperson profit
```

Insufficient Subscriber Balance

```
{ "status": "error", "message": "Insufficient Subscriber Balance Required (1500.00)" } // status: Response status indicating failure // message: Error with required amount when balance is insufficient
```

Important Business Rules

Invoice Generation Rules

- **One Hour Rule:** Only one due invoice can be generated per subscriber per hour
- **Package Validation:** Salesperson must have accounting permissions for the subscriber's package
- **Discount Limits:** Subscriber discount cannot exceed salesperson profit (admin users exempt)

Payment Processing

- **Prepaid Billing:** Full payment required before activation
- **Balance Validation:** Subscriber balance must cover total activation fee when using balance payment
- **Profit Distribution:** Salesperson profits are distributed through the reseller hierarchy

RADIUS Integration

- **Automatic Activation:** Subscriber is activated in RADIUS upon successful payment
- **Package Assignment:** New package settings are applied to subscriber profile
- **Expiration Management:** New expiration date is calculated and set

Integration Guidelines

1. **Validate Subscriber Status:** Ensure subscriber exists and is assigned to your salesperson account
2. **Check Package Permissions:** Verify you have accounting permissions for the subscriber's package
3. **Handle Due Invoices:** Check for existing due invoices and handle the 1-hour waiting period
4. **Payment Validation:** Ensure payment amount covers the full activation fee for successful activation
5. **Balance Management:** When using subscriber balance, verify sufficient funds before making the request
6. **Error Handling:** Implement comprehensive error handling for all business logic scenarios
7. **Invoice Tracking:** Store returned invoice data for future reference and payment tracking

Testing Scenarios

Test Cases to Implement

- Basic activation without payment (invoice generation only)
- Activation with full payment (complete activation flow)
- Activation using subscriber balance
- Handling existing due invoices
- Testing insufficient balance scenarios
- Validating package permission restrictions
- Testing discount limit validations

Common Integration Patterns

Two-Step Activation Process

```
// Step 1: Generate invoice POST /api/v1/subscriber/activation { "subscriber_id": 12345 } // Required: ID of subscriber to activate // Step 2: Process payments (if invoice generated successfully) curl -X POST "https://your-domain.com/api/v1/subscriber/payments/add-balance" \ -H "Content-Type: application/json" \ -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..." \ -d '{ "subscriber_id": 12345, // Required: Same subscriber ID "payment_amount": 1500.00, // Required: Amount to pay "payment_method": 6 // Optional: 6 = bKash payment }'
```

One-Step Activation with Payment

```
// Complete activation in single request curl -X POST "https://your-domain.com/api/v1/subscriber/activation" \ -H "Content-Type: application/json" \ -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..." \ -d '{ "subscriber_id": 12345, // Required: ID of subscriber to activate "payment_amount": 1500.00, // Optional: Amount to pay during activation "payment_method": 6 // Optional: 6 = bKash payment method }'
```