Baro Logistics

B2B API Documentation

Web Services Integration Guide for Business Partners

Version: 2.3

Last Updated: April 29, 2026

Table of Contents

Introduction

Welcome to Baro Logistics B2B API documentation. Our API enables business partners to fully integrate logistics operations into their systems.

Available Services

- **Package Tracking** - Track orders by code or phone number
- **Order Creation** - Create new delivery orders programmatically
- **Order Cancellation** - Cancel owned orders in allowed statuses
- **Price Estimation** - Calculate delivery costs before order creation
- **Batch Orders** - Estimate and create batch orders, plus compatibility checks
- **City Management** - Get list of supported cities
- **Terminal Lookup** - Get list of terminals with locations (public endpoint)
- **Pickup Scheduling** - Get available pickup time slots
- **Discount Validation** - Check discount code validity and preview discount
- **Discount Application** - Apply discount code on owned orders

Key Features

- **JWT-based authentication** - Token-based authentication with token refresh
- **IP whitelist security** - Enhanced security for your account
- **Comprehensive order tracking** - Status history and tracking details
- **All service types** - D2D, D2T, T2D, T2T services
- **Real-time price calculation** - Instant delivery cost estimation
- **Batch order** pricing and creation
- **Flexible pickup scheduling**- Available time slots for pickup coordination

Getting Started

Step 1: Request API Credentials

To access Baro's B2B API, you need to obtain your credentials and permissions:

Contact our technical team:
Email: support@baro.ir

You will receive:

- **Username** - Your unique business username
- **Password** - Your secret password (keep this secure!)
- **Permissions** - Specific API access rights based on your needs

Step 2: Provide IP Addresses for Whitelisting

Security Requirement: For both Stage and Production environments, you must provide us with the IP addresses from which your application will make API requests.

Why IP Whitelisting?

- **Enhanced security** - For your account
- **Protection against unauthorized access** - Prevents misuse
- **Compliance** - Security best practices

What to Provide:

- **List of static IP addresses** - For your servers
- **Separate IPs** - For Stage and Production environments
- **Update us** - If your IP addresses change

Example:

```
Stage Environment IPs:
- 192.168.1.100
- 192.168.1.101

Production Environment IPs:
- 203.0.113.50
- 203.0.113.51
```

Important: API requests from non-whitelisted IPs will be blocked. Ensure you provide all IPs that will access the API. Contact us immediately if you need to add or change IP addresses.

Step 3: Choose Your Environment

We provide two environments for development and production:

Environment	Base URL	Purpose
Stage (Testing)	https://barobus.ir/	Development and testing
Production	https://baro.ir/	Live production environment

Important: Always test your integration in the Stage environment before moving to Production.

Step 4: Access and Use the API Documentation (Swagger)

To explore and test the available API endpoints, you can use our interactive API documentation powered by Swagger.

API Documentation (Swagger):

- https://barobus.ir/api/b2b-docs/

How to Access:

1. Open the Swagger documentation link in your browser.
2. Log in using the **Username** and **Password** you received from our technical team in **Step 1**.
3. After authentication, you can:
    - View all available API endpoints
    - Review request/response schemas
    - Test API calls directly from the browser
    - Understand required parameters and permissions

Important Notes:

- Your access level in Swagger depends on the permissions assigned to your account.
- Use the **Stage environment** for testing and validation before moving to Production.
- Ensure your IP address has been whitelisted (Step 2); otherwise, API requests may be blocked.

Base URLs

Stage Environment

```
https://barobus.ir/api/b2b/
```

Production Environment

```
https://baro.ir/api/b2b/
```

All API endpoints should be appended to the base URL.

Example

```
https://barobus.ir/api/b2b/login/
https://barobus.ir/api/b2b/track-order/
```

Authentication

Baro API uses **JWT (JSON Web Tokens)** for authentication.

Authentication Flow

```
1. Login with credentials -> Get access token + refresh token
2. Use access token for API calls
3. Refresh token when expired
```

1. Login

Endpoint: `POST /api/b2b/login/`

Request:

```
{
  "username": "your_business_username",
  "password": "your_secure_password"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
    "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_expires_in": 86400
  },
  "client_info": {
    "username": "your_business_username",
    "business_name": "Your Company Name"
  },
  "message": "Login successful",
  "message_fa": "ورود موفقیت‌آمیز"
}
```

Token Expiry:

- **Access Token**- 1 hour
- **Refresh Token**- 24 hours

Error Responses:

Missing credentials (400):

```
{
  "success": false,
  "error": "username and password are required",
  "error_fa": "نام کاربری و رمز عبور الزامی هستند"
}
```

Invalid credentials (401):

```
{
  "success": false,
  "error": "Invalid credentials",
  "error_fa": "نام کاربری یا رمز عبور نامعتبر است"
}
```

IP not whitelisted (403):

```
{
  "success": false,
  "error": "IP address not allowed",
  "error_fa": "آدرس IP شما مجاز نیست",
  "client_ip": "203.0.113.100"
}
```

2. Using Access Token

Include the access token in the Authorization header for all API requests:

```
Authorization: Bearer <your_access_token>
```

3. Refresh Token

When your access token expires, use the refresh token to get a new one.

Endpoint: `POST /api/b2b/refresh-token/`

Request:

```
{
  "refresh_token": "your-refresh-token"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "access_token": "new-access-token",
    "token_type": "Bearer",
    "expires_in": 3600
  },
  "message": "Token refreshed successfully",
```

```
  "message_fa": "توکن با موفقیت تازه‌سازی شد"
}
```

Error Responses:

Missing refresh_token (400):

```
{
  "success": false,
  "error": "refresh_token is required",
  "error_fa": "refresh_token الزامی است"
}
```

Invalid token type (401):

```
{
  "success": false,
  "error": "Invalid token type",
  "error_fa": "نوع توکن نامعتبر است"
}
```

Client not found or inactive (401):

```
{
  "success": false,
  "error": "Client not found or inactive",
  "error_fa": "کلاینت یافت نشد یا غیرفعال است"
}
```

Token expired or invalid (401):

```
{
  "success": false,
  "error": "Invalid or expired refresh token",
  "error_fa": "توکن نامعتبر یا منقضی شده است",
  "details": "Token is expired"
}
```

API Endpoints

1. Track Order

Track packages by order code or customer phone number.

Endpoint: `GET /api/b2b/track-order/`

Authentication: Required (Bearer Token)

Permission Required: can_track_orders

Query Parameters:

Parameter	Type	Required	Description
order_code	string	Optional*	8-digit order tracking code
phone_number	string	Optional*	Customer phone (09xxxxxxxxx)

Examples:

```
# Track by order code
GET /api/b2b/track-order/?order_code=12345678

# Track by phone number
GET /api/b2b/track-order/?phone_number=09123456789

# Track by both (more specific)
GET /api/b2b/track-order/?order_code=12345678&phone_number=09123456789
```

Success Response (200 OK):


```
{
  "success": true,
  "data": {
    "orders": [
      {
        "order_code": "12345678",
        "status": "IN_TRANSIT_TO_DESTINATION",
        "status_display": "در حال ارسال به مقصد",
        "service_type": "D2D",
        "service_type_display": "درب به درب",
        "origin_city": "تهران",
        "destination_city": "مشهد",
        "sender_name": "علی احمدی",
        "sender_phone": "09123456789",
        "receiver_name": "محمد رضایی",
        "receiver_phone": "09987654321",
        "weight_kg": 2.5,
```

```
        "total_cost": 150000,
        "is_paid": true,
        "delivery_code": "1234",
        "created_at": "2026-01-25T10:30:00Z",
        "updated_at": "2026-01-25T14:20:00Z",
        "delivered_at": null,
        "scheduled_pickup_time": "2026-01-26T09:00:00Z",
        "scheduled_pickup_end_time": "2026-01-26T11:00:00Z",
        "status_history": [
          {
            "status": "PENDING",
            "status_display": "در انتظار پرداخت",
            "notes": null,
            "changed_by": null,
            "created_at": "2026-01-25T10:30:00Z"
          },
          {
            "status": "AWAITING_PICKUP",
            "status_display": "در انتظار تحویل به پیک",
            "notes": "پیک تخصیص داده شد",
            "changed_by": "علی احمدی",
            "created_at": "2026-01-25T11:00:00Z"
          }
        ]
      }
    ],
    "count": 1
  },
  "message": "Found 1 order(s)"
}
```


Notes:

- **Returns up to 50 orders per request**
- **Excludes COMPLETED and CANCELLED orders**
- **delivery_code is only shown for paid orders**
- **terminal_warehouse_doc_url is only shown for whitelisted users**

Error Responses:

Missing parameters (400):

```
{
  "success": false,
  "error": "Either order_code or phone_number is required",
  "error_fa": "حداقل یکی از order_code یا phone_number الزامی است"
}
```

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

No orders found (404):

```
{
  "success": false,
  "error": "No orders found",
  "error_fa": "سفارشی یافت نشد"
}
```

2. Create Order

Create a new delivery order programmatically.

Endpoint: `POST /api/b2b/create-order/`

Authentication: Required (Bearer Token)

Permission Required: can_create_orders

Important: The sender_phone must belong to an existing registered user in the Baro system. This user will be set as the order's customer.

Request Body:

```
{
  "service_type": "D2D",
  "payment_type": "PREPAID",
  "sender_name": "علی احمدی",
  "sender_phone": "09123456789",
  "receiver_name": "محمد رضایی",
```

```
  "receiver_phone": "09987654321",
  "origin_province": "تهران",
  "origin_city": "تهران",
  "origin_address": "خیابان ولیعصر، پلاک 123",
  "origin_postal_code": "1234567890",
  "origin_lat": 35.6892,
  "origin_lng": 51.3890,
  "destination_province": "خراسان رضوی",
  "destination_city": "مشهد",
  "destination_address": "خیابان امام رضا، پلاک 456",
  "destination_postal_code": "9876543210",
  "destination_lat": 36.2974,
  "destination_lng": 59.6067,
  "custom_length_cm": 30,
  "custom_width_cm": 20,
  "custom_height_cm": 15,
  "weight_kg": 2.5,
  "estimated_value": 500000,
  "needs_insurance": true,
  "needs_packaging": false,
  "is_fragile": true,
  "is_unconventional": false,
  "description": "کتاب و لوازم التحریر",
  "scheduled_pickup_time": "2026-02-17T09:00:00Z",
  "scheduled_pickup_end_time": "2026-02-17T11:00:00Z"
}
```

Field Descriptions:

Field	Type	Required	Description
service_type	string	Yes	D2D, D2T, T2D, T2T
payment_type	string	Yes	PREPAID or POSTPAID
sender_name	string	Yes	Sender's full name
sender_phone	string	Yes	Must be registered user
receiver_name	string	Yes	Receiver's full name
receiver_phone	string	Yes	Receiver phone number
origin_city	string/int	Yes	Origin city name or ID
destination_city	string/int	Yes	Destination city name or ID
origin_address	string	Conditional	Required for D2D, D2T
origin_lat	float	Conditional	Required for D2D, D2T
origin_lng	float	Conditional	Required for D2D, D2T

Field	Type	Required	Description
destination_address	string	Conditional	Required for D2D, T2D
destination_lat	float	Conditional	Required for D2D, T2D
destination_lng	float	Conditional	Required for D2D, T2D
weight_kg	float	Yes	Package weight in kg
estimated_value	int	Yes	Package value in Rials
needs_insurance	boolean	Yes	Whether insurance needed
needs_packaging	boolean	Yes	Whether packaging needed
is_fragile	boolean	Yes	Whether package is fragile

Success Response (201 Created):

```
{
  "success": true,
  "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52",
  "order_code": "12345678",
  "amount": 150000,
  "message": "سفارش با موفقیت ثبت شد"
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Customer not found (400):

```
{
  "success": false,
  "error": "No customer account found for sender_phone",
  "error_fa": "کاربر متناظر با شماره فرستنده یافت نشد",
  "details": "sender_phone must belong to an existing registered user profile"
}
```

Validation errors (400):

```
{
  "success": false,
  "error": "Validation error",
  "error_en": "Validation error",
  "errors": {
    "origin_address": ["برای سرویس دربمحور، آدرس مبدا الزامی است."],
    "weight_kg": ["این فیلد الزامی است."],
    "origin_city": ["شهر \"نامعتبر\" یافت نشد"]
  }
}
```

Route not found (400):

```
{
  "success": false,
  "error": "Validation error",
  "errors": {
    "route": "هیچ مسیر فعالی بین این دو شهر وجود ندارد",
    "route_en": "No active route exists between these cities",
    "error_code": "NO_ROUTE_BETWEEN_CITIES"
  }
}
```

3. Estimate Price

Calculate delivery cost before creating an order.

Endpoint: `POST /api/b2b/estimate-price/`

Authentication: Required (Bearer Token)

Permission Required: can_estimate_prices

Uses the same schema as Create Order endpoint. You can send the complete order data to get an accurate price estimate.

Request Body:

```
{
  "service_type": "D2D",
  "origin_city": "تهران",
  "destination_city": "مشهد",
  "custom_length_cm": 30,
  "custom_width_cm": 20,
  "custom_height_cm": 15,
  "weight_kg": 2.5,
  "estimated_value": 500000,
  "needs_insurance": true,
  "needs_packaging": false,
  "is_fragile": true,
```

```
  "is_unconventional": false
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "total_price": 185000,
    "breakdown": {
      "base_price": 50000,
      "weight_price": 65000,
      "distance_price": 40000,
      "insurance_fee": 15000,
      "packaging_fee": 0,
      "subtotal": 170000,
      "tax": 15000,
      "tax_rate": 9
    },
    "message": "قیمت با موفقیت محاسبه شد"
  },
  "message": "Price calculated successfully"
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Validation error (400):

```
{
  "success": false,
  "error": "Validation error",
  "error_en": "Validation error",
  "errors": {
    "weight_kg": ["این فیلد الزامی است."],
    "origin_city": ["شهر \"نامعتبر\" یافت نشد"]
  }
}
```

4. Estimate Batch Price

Estimate total price for multiple orders without creating them.

Endpoint: `POST /api/b2b/batch-orders/estimate-price/`

Authentication: Required (Bearer Token)

Permission Required: can_estimate_batch_prices

Request Body:

Each item in `orders` uses the same schema as the Create Order endpoint. Minimum 2 orders are required.

```
{
  "orders": [
    {
      "service_type": "D2D",
      "payment_type": "PREPAID",
      "sender_name": "علی احمدی",
      "sender_phone": "09123456789",
      "receiver_name": "رضا کریمی",
      "receiver_phone": "09987654321",
      "origin_province": "تهران",
      "origin_city": "تهران",
      "origin_address": "ولیعصر، پلاک ۱۰",
      "origin_lat": 35.715,
      "origin_lng": 51.404,
      "destination_province": "اصفهان",
      "destination_city": "اصفهان",
      "destination_address": "چهارباغ، پلاک ۲۲",
      "destination_lat": 32.657,
      "destination_lng": 51.677,
      "custom_length_cm": 30,
      "custom_width_cm": 20,
      "custom_height_cm": 15,
      "weight_kg": 2.5,
      "estimated_value": 800000,
      "needs_insurance": true,
      "needs_packaging": false,
      "is_fragile": false
    },
    {
      "service_type": "D2D",
      "payment_type": "PREPAID",
      "sender_name": "علی احمدی",
      "sender_phone": "09123456789",
      "receiver_name": "سارا حسینی",
```

```
      "receiver_phone": "09351234567",
      "origin_province": "تهران",
      "origin_city": "تهران",
      "origin_address": "ولیعصر، پلاک ۱۰",
      "origin_lat": 35.715,
      "origin_lng": 51.404,
      "destination_province": "شیراز",
      "destination_city": "شیراز",
      "destination_address": "معالی آباد، کوچه ۳",
      "destination_lat": 29.62,
      "destination_lng": 52.53,
      "custom_length_cm": 40,
      "custom_width_cm": 25,
      "custom_height_cm": 18,
      "weight_kg": 3.2,
      "estimated_value": 1200000,
      "needs_insurance": true,
      "needs_packaging": true,
      "is_fragile": true
    }
  ]
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "orders_count": 2,
    "total_estimated_price": 320000,
    "currency": "TOMAN",
    "orders": [
      {
        "index": 0,
        "service_type": "D2D",
        "sender_name": "علی احمدی",
        "receiver_name": "رضا کریمی",
        "estimated_price": 150000
      },
      {
        "index": 1,
        "service_type": "D2D",
        "sender_name": "علی احمدی",
        "receiver_name": "سارا حسینی",
        "estimated_price": 170000
      }
    ]
  }
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Validation error (400):

```
{
  "success": false,
  "error": "اعتبارسنجی یک یا چند سفارش با خطا مواجه شد.",
  "message": "اعتبارسنجی یک یا چند سفارش با خطا مواجه شد.",
  "error_en": "Validation failed for one or more orders.",
  "orders": {
    "1": {
      "origin_city": ["شهر \"نامعتبر\" یافت نشد"]
    }
  }
}
```

Validation error - Origin warehouse mismatch (400):

```
{
  "success": false,
  "error": "همه سفارش‌های گروهی باید از یک انبار مبدا ارسال شوند.",
  "message": "همه سفارش‌های گروهی باید از یک انبار مبدا ارسال شوند.",
  "error_en": "All orders in a batch must have the same origin warehouse."
}
```

5. Create Batch Order

Create a batch order and attach all created orders to it.

Endpoint: `POST /api/b2b/batch-orders/create/`

Authentication: Required (Bearer Token)

Permission Required: can_create_batch_orders

Request Body:

Each item in `orders` uses the same schema as the Create Order endpoint. Minimum 2 orders are required.

```
{
  "pickup_time": "2026-03-16T10:00:00Z",
  "orders": [
    {
      "service_type": "D2D",
      "payment_type": "PREPAID",
      "sender_name": "علی احمدی",
      "sender_phone": "09123456789",
      "receiver_name": "رضا کریمی",
      "receiver_phone": "09987654321",
      "origin_province": "تهران",
      "origin_city": "تهران",
      "origin_address": "ولیعصر، پلاک ۱۰",
      "origin_lat": 35.715,
      "origin_lng": 51.404,
      "destination_province": "اصفهان",
      "destination_city": "اصفهان",
      "destination_address": "چهارباغ، پلاک ۲۲",
      "destination_lat": 32.657,
      "destination_lng": 51.677,
      "custom_length_cm": 30,
      "custom_width_cm": 20,
      "custom_height_cm": 15,
      "weight_kg": 2.5,
      "estimated_value": 800000,
      "needs_insurance": true,
      "needs_packaging": false,
      "is_fragile": false
    },
    {
      "service_type": "D2D",
      "payment_type": "PREPAID",
      "sender_name": "علی احمدی",
      "sender_phone": "09123456789",
      "receiver_name": "سارا حسینی",
      "receiver_phone": "09351234567",
      "origin_province": "تهران",
      "origin_city": "تهران",
      "origin_address": "ولیعصر، پلاک ۱۰",
      "origin_lat": 35.715,
      "origin_lng": 51.404,
      "destination_province": "شیراز",
      "destination_city": "شیراز",
```

```
      "destination_address": "معالی آباد، کوچه ۳",
      "destination_lat": 29.62,
      "destination_lng": 52.53,
      "custom_length_cm": 40,
      "custom_width_cm": 25,
      "custom_height_cm": 18,
      "weight_kg": 3.2,
      "estimated_value": 1200000,
      "needs_insurance": true,
      "needs_packaging": true,
      "is_fragile": true
    }
  ]
}
```

Field Notes:

`pickup_time` (optional): Pickup datetime for the whole batch. If null or omitted, the fastest time possible will be used for all orders in the batch. This behaves the same as single order creation.

`orders`: Array of order payloads (minimum 2 orders required for a batch).

Success Response (201 Created):

```
{
  "success": true,
  "data": {
    "id": "2a15fdb7-6d1e-4e95-8f46-0b2bde4f1f5a",
    "type": "batch_order",
    "batch_code": "123456789012",
    "pickup_time": "2026-03-16T10:00:00Z",
    "origin_warehouse": 5,
    "total_cost": 320000,
    "is_paid": false,
    "orders_count": 2,
    "orders": [
      {
        "id": "d5f0f10a-05f5-4f9e-9d6b-4f1a2f6b3c7a",
        "order_code": "12345678",
        "status": "PENDING",
        "status_display": "در انتظار پرداخت",
        "origin_warehouse": 5,
        "destination_warehouse": 8,
        "total_cost": 150000,
        "is_paid": false,
        "created_at": "2026-03-16T10:02:00Z"
      }
    ]
  }
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Validation error (400):

```
{
  "success": false,
  "error": "اعتبارسنجی یک یا چند سفارش با خطا مواجه شد.",
  "message": "اعتبارسنجی یک یا چند سفارش با خطا مواجه شد.",
  "error_en": "Validation failed for one or more orders.",
  "orders": {
    "0": {
      "scheduled_pickup_time": ["Batch pickup time must match every D2* order pickup time."]
    }
  }
}
```

6. Validate Add Order (Batch)

Validate whether a new order can be added to a batch by comparing origin warehouse assignment.

Endpoint: `POST /api/b2b/batch-orders/validate-add-order/`

Authentication: Required (Bearer Token)

Permission Required: can_validate_batch_add_order

Request Body:

```
{
  "first_origin_city": "Tehran",
  "first_destination_city": "Shiraz",
```

```
  "new_origin_city": "Tehran",
  "new_destination_city": "Isfahan"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "is_valid": true,
  "origin_warehouse_id": 12,
  "origin_warehouse_name": "انبار مرکزی تهران"
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Validation error (400):

```
{
  "success": false,
  "is_valid": false,
  "error": " امکان افزودن این سفارش به سفارش گروهی فعلی وجود ندارد، چون انبار مبدا آن متفاوت است.",
  "message": " امکان افزودن این سفارش به سفارش گروهی فعلی وجود ندارد، چون انبار مبدا آن متفاوت است.",
  "error_en": "New order does not map to the same origin warehouse as the first order.",
  "first_origin_warehouse_id": 12,
  "first_origin_warehouse_name": "انبار مرکزی تهران",
  "new_origin_warehouse_id": 14,
  "new_origin_warehouse_name": "انبار مرکزی مشهد"
}
```

7. Batch Compatible Cities

Get destination cities that keep the same origin warehouse for the given origin city.

Endpoint: `POST /api/b2b/batch-orders/compatible-cities/`

Authentication: Required (Bearer Token)

Permission Required: can_view_batch_compatible_cities

Request Body:

```
{
  "origin_city": "Tehran",
  "destination_city": "Shiraz"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "origin_city": {
    "id": 1,
    "name_fa": "تهران",
    "name_en": "Tehran"
  },
  "base_destination_city": {
    "id": 2,
    "name_fa": "شیراز",
    "name_en": "Shiraz"
  },
  "origin_warehouse": {
    "id": 12,
    "name": "انبار مرکزی تهران",
    "terminal_name": "ترمینال جنوب تهران"
  },
  "compatible_cities": [
    {
      "id": 3,
      "name_fa": "اصفهان",
      "name_en": "Isfahan",
      "province_name_fa": "اصفهان",
      "province_name_en": "Isfahan"
    }
  ],
  "count": 1
}
```

Error Responses:

Authentication required (401):

```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

8. List Cities

Get list of all active cities supported by Baro.

Endpoint: `GET /api/b2b/cities/`

Authentication: Required (Bearer Token)

Permission Required: can_view_cities

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "cities": [
      {
        "id": 1,
        "name_fa": "تهران",
        "name_en": "Tehran",
        "province": {
          "id": 1,
          "name_fa": "تهران",
          "name_en": "Tehran"
        },
        "is_active": true
      },
      {
        "id": 2,
        "name_fa": "مشهد",
        "name_en": "Mashhad",
        "province": {
          "id": 9,
          "name_fa": "خراسان رضوی",
```

```
          "name_en": "Razavi Khorasan"
        },
        "is_active": true
      }
    ],
    "count": 2
  }
}
```


Notes:

- **Response is cached for better performance**
- **Only returns active cities**
- **Cities are ordered alphabetically by Persian name**

Error Responses:

Authentication required (401):


```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```


Permission denied (403):


```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```


9. List Terminals

Get list of all active terminals with their locations and details.

Endpoint: `GET /api/terminals/`

Authentication: Not Required (Public Endpoint)

Success Response (200 OK):


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
```

```
      "terminal_id": "TRM001",
      "name": "ترمینال جنوب تهران",
      "city": 1,
      "city_name": "تهران",
      "province_name": "تهران",
      "address": "تهران، جنوب، خیابان آزادی",
      "phone_number": "02112345678",
      "latitude": 35.6892,
      "longitude": 51.3890,
      "is_active": true,
      "warehouses_count": 2
    },
    {
      "id": 2,
      "terminal_id": "TRM002",
      "name": "ترمینال مرکزی مشهد",
      "city": 2,
      "city_name": "مشهد",
      "province_name": "خراسان رضوی",
      "address": "مشهد، بلوار وکیل آباد",
      "phone_number": "05138765432",
      "latitude": 36.2974,
      "longitude": 59.6067,
      "is_active": true,
      "warehouses_count": 1
    }
  ]
}
```

10. Get Pickup Slots

Get available pickup time slots for order scheduling.

Endpoint: `POST /api/b2b/pickup-slots/`

Authentication: Required (Bearer Token)

Permission Required: can_view_pickup_slots

Request Body:

```
{
  "service_type": "D2D",
  "origin_city": "تهران",
  "destination_city": "مشهد",
  "max_slots": 40
}
```

Field Descriptions:

Field	Type	Required	Description
service_type	string	Yes	D2D, D2T, T2D, T2T
origin_city	string	Yes	Origin city name
destination_city	string	Yes	Destination city name
origin_terminal_id	int	No	Specific origin terminal ID
warehouse_id	int	No	Specific warehouse ID
max_slots	int	No	Max slots to return (default: 40)

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "slots": [
      {
        "date": "2026-02-17",
        "day_name": "دوشنبه",
        "start_datetime": "2026-02-17T09:00:00+03:30",
        "end_datetime": "2026-02-17T11:00:00+03:30",
        "display": "دوشنبه 28 بهمن - 09:00 تا 11:00",
        "is_available": true
      },
      {
```

```
        "date": "2026-02-17",
        "day_name": "دوشنبه",
        "start_datetime": "2026-02-17T11:00:00+03:30",
        "end_datetime": "2026-02-17T13:00:00+03:30",
        "display": "13:00 تا 11:00 - دوشنبه 28 بهمن",
        "is_available": true
      }
    ],
    "warehouse": {
      "id": 1,
      "name": "انبار مرکزی تهران",
      "terminal": {
        "id": 1,
        "name": "ترمینال جنوب تهران"
      }
    }
  },
  "message": "بازه‌های زمانی با موفقیت بازیابی شد"
}
```


Notes:

- **For T2D and T2T services, pickup scheduling is not required**
- **Slots are generated based on warehouse working hours**
- **Times are in Tehran timezone (Asia/Tehran, UTC+3:30)**
- **Use start_datetime and end_datetime for scheduled_pickup_time fields**

Error Responses:

Authentication required (401):


```
{
  "success": false,
  "error": "Authentication required",
  "error_fa": "احراز هویت الزامی است"
}
```


Permission denied (403):


```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```


Invalid max_slots (400):


```
{
  "success": false,
  "error": "max_slots نامعتبر است",
```

```
  "error_en": "Invalid max_slots"
}
```

No route found (400):

```
{
  "success": false,
  "error": "هیچ مسیر فعالی بین این دو شهر وجود ندارد",
  "error_en": "No active route exists between these cities"
}
```

T2D/T2T service (200):

```
{
  "success": true,
  "message": "برای این نوع سرویس نیازی به انتخاب بازه زمانی نیست",
  "message_en": "Pickup slots not needed for this service type",
  "slots": [],
  "service_type": "T2D"
}
```

11. Cancel Order

Cancel one order owned by the B2B client.

Endpoint: `POST /api/b2b/cancel-order/`

Authentication: Required (Bearer Token)

Permission Required: can_cancel_orders

Ownership Rule:

- The order must belong to the same owner phone as the B2B client `contact_phone`.
- If the order belongs to another account, the API returns `404`.

Cancellable Statuses:

- `PENDING`
- `SCHEDULED_PICKUP`
- `SEARCHING_FOR_SENDER_COURIER`
- `AWAITING_PICKUP`

Request Body:

```
{
  "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Order cancelled successfully",
  "message_fa": "سفارش با موفقیت لغو شد",
  "data": {
    "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52",
    "order_code": "12345678",
    "previous_status": "PENDING",
    "status": "CANCELLED",
    "status_display": "لغو شده"
  }
}
```

Error Responses:

Missing order_id (400):

```
{
  "success": false,
  "error": "order_id is required",
  "error_fa": "فیلد order_id الزامی است"
}
```

Invalid order_id format (400):

```
{
  "success": false,
  "error": "Invalid order_id format",
  "error_fa": "فرمت order_id نامعتبر است"
}
```

Missing order_id (400):

```
{
  "success": false,
  "error": "order_id is required",
  "error_fa": "فیلد order_id الزامی است"
}
```

Already cancelled (400):

```
{
  "success": false,
  "error": "This order is already cancelled",
  "error_fa": "این سفارش قبلاً لغو شده است"
}
```

Status not cancellable (400):

```
{
  "success": false,
  "error": "Order cannot be cancelled at this stage. Please contact support.",
  "error_fa": "در این مرحله امکان لغو سفارش وجود ندارد. لطفاً با پشتیبانی تماس بگیرید.",
```

```
  "current_status": "IN_TRANSIT_TO_DESTINATION",
  "current_status_display": "در حال ارسال به مقصد"
}
```

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

12. Validate Discount Code

Validate a discount code and preview its effect for a route/service before order creation.

Endpoint: `POST /api/b2b/discount/validate/`

Authentication: Required (Bearer Token)

Permission Required: can_estimate_prices

Request Body:

```
{
  "code": "WELCOME50",
  "origin_city": "تهران",
  "destination_city": "شیراز",
  "service_type": "D2D",
  "amount": 150000
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "code": "WELCOME50",
    "discount_type": "PERCENTAGE",
    "discount_value": 50,
    "discount_amount": 27500,
    "original_amount": 55000,
    "final_amount": 27500,
    "max_discount_amount": null,
    "description": "First purchase discount"
  }
}
```

Error Responses:

Permission denied (403):

```
{
  "success": false,
  "error": "Permission denied",
  "error_fa": "دسترسی مجاز نیست"
}
```

Validation error (400):

```
{
  "success": false,
  "error": "Validation error",
  "error_fa": "خطای اعتبارسنجی",
  "errors": {
    "origin_city": [
      "شهر \"نامعتبر\" یافت نشد."
    ]
  }
}
```

Discount invalid (400):

```
{
  "success": false,
  "error": "کد تخفیف یافت نشد",
  "error_en": "Discount code not found",
  "error_code": "CODE_NOT_FOUND"
}
```

13. Apply Discount Code

Apply a discount code to an order owned by the B2B client.

Endpoint: `POST /api/b2b/discount/apply/`

Authentication: Required (Bearer Token)

Permission Required: can_create_orders

Request Body:

```
{
  "code": "WELCOME50",
  "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
    "discount_code": "WELCOME50",
    "discount_amount": 75000,
    "original_amount": 150000,
    "final_amount": 75000,
    "discount_type": "PERCENTAGE",
    "message": "کد تخفیف با موفقیت اعمال شد. مبلغ تخفیف: 75,000 تومان",
    "new_total_cost": 75000
  }
}
```

Error Responses:

Discount invalid (400):

```
{
  "success": false,
  "error": "کد تخفیف یافت نشد",
  "error_en": "Discount code not found",
  "error_code": "CODE_NOT_FOUND"
}
```

Already has discount (400):

```
{
  "success": false,
  "error": "This order already has a discount code. Remove the current discount first.",
  "error_fa": "این سفارش قبلاً کد تخفیف دارد. ابتدا کد فعلی را حذف کنید."
}
```

Order not found for this client (404):

```
{
  "success": false,
  "error": "Order not found for this client",
  "error_fa": "سفارش برای این کلاینت یافت نشد"
}
```

14. Wallet Balance

Get wallet balance linked to B2B client `contact_phone`.

Endpoint: `GET /api/b2b/wallet/balance/`

Authentication: Required (Bearer Token)

Permission Required: None (no extra permission flag)

Ownership Rule: B2B client is mapped to wallet owner using `contact_phone`, `contact_phone` must belong to an existing registered user.

Success Response (200 OK):

```
{
  "success": true,
  "balance": 250000,
  "balance_display": "250,000 تومان",
  "owner_phone": "09123456789"
}
```

Error Responses:

Error Response (400):

```
{
  "success": false,
  "error": " Wallet owner is not registered",
  "error_fa": "برای استفاده از کیف پول لطفا ابتدا در با شماره‌ای که اعلام کردید در سایت بارو ثبت‌نام کنید ."
}
```

15. Wallet Pay Order

Pay one owned order with wallet (full wallet payment or wallet).

Endpoint: `POST /api/b2b/wallet/pay-order/`

Authentication: Required (Bearer Token)

Permission Required: None (no extra permission flag)

Payment Rule: This endpoint is wallet-only.

Request Body:

```
{
  "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "paid_with_wallet": true,
  "wallet_amount": 150000,
  "gateway_amount": 0,
  "new_balance": 130000,
  "order_code": "12345678"
}
```

16. Wallet Pay Batch Order

Pay one owned batch order with wallet.

Endpoint: `POST /api/b2b/wallet/pay-batch-order/`

Authentication: Required (Bearer Token)

Permission Required: None (no extra permission flag)

Payment Rule: This endpoint is wallet-only.

Request Body:

```
{
  "order_id": "9f7ccca2-3f46-4a49-b5cc-30f5e9bf1f52"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "paid_with_wallet": true,
  "wallet_amount": 350000,
  "gateway_amount": 0,
  "new_balance": 50000,
  "batch_code": "B25042901"
}
```

Order Status Reference

The following table lists all possible order statuses:

Status Code	Persian Display	English Display	Description
PENDING	در انتظار پرداخت	Pending Payment	Order created, awaiting payment
SCHEDULED_PICKUP	برنامه‌ریزی شده برای دریافت	Scheduled for Pickup	Pickup time scheduled
SEARCHING_FOR_SENDER_COURIER	در حال جستجوی پیک	Searching for Courier	Finding courier for pickup
AWAITING_PICKUP	در انتظار تحویل به پیک	Awaiting Pickup	Waiting for courier pickup
AWAITING_DROPOFF	در انتظار تحویل به ترمینال	Awaiting Dropoff	Waiting for terminal delivery
DELIVERED_TO_ORIGIN_TERMINAL	تحویل به ترمینال مبدا	Delivered to Origin	Package at origin terminal
TERMINAL_APPROVED	تایید شده توسط ترمینال	Terminal Approved	Terminal verified package
AWAITING_SECURITY_CHECK	در انتظار بررسی امنیت	Awaiting Security	Security check pending
SECURITY_APPROVED	تایید شده توسط امنیت	Security Approved	Security check passed
SECURITY_REJECTED	رد شده توسط امنیت	Security Rejected	Security check failed
LOADED_ON_BUS	بارگیری شده در اتوبوس	Loaded on Bus	Package loaded for transport
IN_TRANSIT_TO_DESTINATION	در حال ارسال به مقصد	In Transit	Package in transit
ARRIVED_AT_DESTINATION	رسیده به ترمینال مقصد	Arrived at Destination	Package at destination terminal
SEARCHING_FOR_RECEIVER_COURIER	در حال جستجوی پیک	Searching for Courier	Finding courier for delivery
READY_FOR_DELIVERY	آماده ارسال به گیرنده	Ready for Delivery	Ready for final delivery

Status Code	Persian Display	English Display	Description
READY_FOR_PICKUP	آماده تحویل در ترمینال	Ready for Pickup	Ready for customer pickup
COMPLETED	تحویل داده شد	Completed	Package delivered successfully
CANCELLED	لغو شده	Cancelled	Order cancelled

Service Types

Baro offers four service types based on pickup and delivery locations:

Code	Name (Persian)	Name (English)	Description
D2D	درب به درب	Door to Door	Pickup from sender's address, delivery to receiver's address
D2T	درب به ترمینال	Door to Terminal	Pickup from sender's address, receiver picks up from terminal
T2D	ترمینال به درب	Terminal to Door	Sender drops off at terminal, delivery to receiver's address
T2T	ترمینال به ترمینال	Terminal to Terminal	Sender drops off at terminal, receiver picks up from terminal

Service Type Requirements

Service	Origin Requirements	Destination Requirements
D2D	Address, coordinates (lat/lng)	Address, coordinates (lat/lng)
D2T	Address, coordinates (lat/lng)	Terminal ID or city
T2D	Terminal ID or city	Address, coordinates (lat/lng)
T2T	Terminal ID or city	Terminal ID or city

Notes:

- **For terminal-based services, if you provide only the city, the system will automatically assign the appropriate terminal**
- **You can get the list of available terminals using the public /api/terminals/ endpoint**
- **D2D and D2T services require pickup scheduling**
- **T2D and T2T services don't require pickup scheduling (customer drops off at terminal)**

Error Handling

HTTP Status Codes

Code	Meaning	Description
200	OK	Request successful
400	Bad Request	Invalid request parameters
401	Unauthorized	Invalid or expired token
403	Forbidden	Insufficient permissions
404	Not Found	Resource not found
429	Too Many Requests	Rate limit exceeded
500	Internal Server Error	Server error

Error Response Format

```
{
  "success": false,
  "error": "Error message in English",
  "error_fa": "پیام خطا به فارسی",
  "code": "ERROR_CODE"
}
```

Common Errors

Invalid Credentials (401):

```
{
  "success": false,
  "error": "Invalid username or password",
  "error_fa": "نام کاربری یا رمز عبور نامعتبر است"
}
```

IP Not Whitelisted (403):

```
{
  "success": false,
  "error": "Access denied. Your IP address is not whitelisted.",
  "error_fa": "دسترسی رد شد. آدرس IP شما در لیست مجاز نیست."
}
```

Token Expired (401):

```
{
  "success": false,
  "error": "Token has expired",
  "error_fa": "توکن منقضی شده است"
}
```

Missing Parameters (400):

```
{
  "success": false,
  "error": "At least one of order_code or phone_number is required",
  "error_fa": "حداقل یکی از کد سفارش یا شماره تلفن الزامی است"
}
```

Rate Limit Exceeded (429):

```
{
  "success": false,
  "error": "Rate limit exceeded. Please try again later.",
  "error_fa": "محدودیت تعداد درخواست .لطفاً بعداً تلاش کنید."
}
```

Code Examples

Python Example

```
import requests
from datetime import datetime, timedelta

class BaroAPIClient:
    def __init__(self, username, password, base_url="https://barobus.ir/api/b2b/"):
        self.username = username
        self.password = password
        self.base_url = base_url
        self.access_token = None
        self.refresh_token = None
        self.token_expiry = None

    def login(self):
        """Authenticate and get tokens"""
        response = requests.post(
            f"{self.base_url}/login/",
            json={"username": self.username, "password": self.password}
        )
        response.raise_for_status()
        data = response.json()

        self.access_token = data['data']['access_token']
        self.refresh_token = data['data']['refresh_token']
        self.token_expiry = datetime.now() +
timedelta(seconds=data['data']['expires_in'])

        return data

    def refresh_access_token(self):
        """Refresh the access token"""
        response = requests.post(
            f"{self.base_url}/refresh-token/",
            json={"refresh_token": self.refresh_token}
        )
        response.raise_for_status()
        data = response.json()

        self.access_token = data['data']['access_token']
        self.token_expiry = datetime.now() +
timedelta(seconds=data['data']['expires_in'])

        return data

    def _ensure_authenticated(self):
```

```
        """Ensure we have a valid token"""
        if not self.access_token or datetime.now() >= self.token_expiry:
            if self.refresh_token:
                try:
                    self.refresh_access_token()
                except:
                    self.login()
            else:
                self.login()

    def _get_headers(self):
        """Get authorization headers"""
        self._ensure_authenticated()
        return {"Authorization": f"Bearer {self.access_token}"}

    def track_order(self, order_code=None, phone_number=None):
        """Track order by code or phone number"""
        params = {}
        if order_code:
            params['order_code'] = order_code
        if phone_number:
            params['phone_number'] = phone_number

        if not params:
            raise ValueError("At least one of order_code or phone_number is required")

        response = requests.get(
            f"{self.base_url}/track-order/",
            params=params,
            headers=self._get_headers()
        )
        response.raise_for_status()
        return response.json()

    def create_order(self, order_data):
        """Create a new order"""
        response = requests.post(
            f"{self.base_url}/create-order/",
            json=order_data,
            headers=self._get_headers()
        )
        response.raise_for_status()
        return response.json()

    def estimate_price(self, order_data):
        """Estimate order price"""
        response = requests.post(
            f"{self.base_url}/estimate-price/",
            json=order_data,
            headers=self._get_headers()
```

```
        )
        response.raise_for_status()
        return response.json()

    def get_cities(self):
        """Get list of active cities"""
        response = requests.get(
            f"{self.base_url}/cities/",
            headers=self._get_headers()
        )
        response.raise_for_status()
        return response.json()

    def get_pickup_slots(self, service_type, origin_city, destination_city,
max_slots=40):
        """Get available pickup slots"""
        response = requests.post(
            f"{self.base_url}/pickup-slots/",
            json={
                "service_type": service_type,
                "origin_city": origin_city,
                "destination_city": destination_city,
                "max_slots": max_slots
            },
            headers=self._get_headers()
        )
        response.raise_for_status()
        return response.json()

# Usage Examples
if __name__ == "__main__":
    # Initialize client
    client = BaroAPIClient(
        username="your_business_username",
        password="your_secure_password"
    )

    # Example 1: Track order
    result = client.track_order(order_code="12345678")
    print(f"Found {result['data']['count']} order(s)")

    for order in result['data']['orders']:
        print(f"Order: {order['order_code']}")
        print(f"Status: {order['status_display']}")
        print(f"From: {order['origin_city']} -> To: {order['destination_city']}")
        print("---")

    # Example 2: Get cities
    cities = client.get_cities()
    print(f"Available cities: {cities['data']['count']}")
```

```
# Example 3: Get pickup slots
slots = client.get_pickup_slots(
    service_type="D2D",
    origin_city="تهران",
    destination_city="مشهد"
)
print(f"Available slots: {len(slots['data']['slots'])}")

# Example 4: Estimate price
price_estimate = client.estimate_price({
    "service_type": "D2D",
    "origin_city": "تهران",
    "destination_city": "مشهد",
    "custom_length_cm": 30,
    "custom_width_cm": 20,
    "custom_height_cm": 15,
    "weight_kg": 2.5,
    "estimated_value": 500000,
    "needs_insurance": True,
    "needs_packaging": False,
    "is_fragile": True,
    "is_unconventional": False
})
print(f"Estimated price: {price_estimate['data']['total_price']} Rials")

# Example 5: Create order
order = client.create_order({
    "service_type": "D2D",
    "payment_type": "ONLINE",
    "sender_name": "علی احمدی",
    "sender_phone": "09123456789",
    "receiver_name": "محمد رضایی",
    "receiver_phone": "09987654321",
    "origin_province": "تهران",
    "origin_city": "تهران",
    "origin_address": "خیابان ولیعصر، پلاک 123",
    "origin_lat": 35.6892,
    "origin_lng": 51.3890,
    "destination_province": "خراسان رضوی",
    "destination_city": "مشهد",
    "destination_address": "خیابان امام رضا، پلاک 456",
    "destination_lat": 36.2974,
    "destination_lng": 59.6067,
    "custom_length_cm": 30,
    "custom_width_cm": 20,
    "custom_height_cm": 15,
    "weight_kg": 2.5,
    "estimated_value": 500000,
    "needs_insurance": True,
```

```
        "needs_packaging": False,
        "is_fragile": True,
        "description": "کتاب و لوازم التحریر",
        "scheduled_pickup_time": slots['data']['slots'][0]['start_datetime'],
        "scheduled_pickup_end_time": slots['data']['slots'][0]['end_datetime']
    })
    print(f"Order created: {order['order_code']}")
    print(f"Amount: {order['amount']} Rials")
```

JavaScript/Node.js Example

```
const axios = require('axios');

class BaroAPIClient {
    constructor(username, password, baseUrl = 'https://barobus.ir/api/b2b/') {
        this.username = username;
        this.password = password;
        this.baseUrl = baseUrl;
        this.accessToken = null;
        this.refreshToken = null;
        this.tokenExpiry = null;
    }

    async login() {
        const response = await axios.post(`${this.baseUrl}/login/`, {
            username: this.username,
            password: this.password
        });

        this.accessToken = response.data.data.access_token;
        this.refreshToken = response.data.data.refresh_token;
        this.tokenExpiry = Date.now() + (response.data.data.expires_in * 1000);

        return response.data;
    }

    async refreshAccessToken() {
        const response = await axios.post(`${this.baseUrl}/refresh-token/`, {
            refresh_token: this.refreshToken
        });

        this.accessToken = response.data.data.access_token;
        this.tokenExpiry = Date.now() + (response.data.data.expires_in * 1000);

        return response.data;
    }

    async ensureAuthenticated() {
        if (!this.accessToken || Date.now() >= this.tokenExpiry) {
            if (this.refreshToken) {
```

```
            try {
                await this.refreshAccessToken();
            } catch (error) {
                await this.login();
            }
        } else {
            await this.login();
        }
    }
}

getHeaders() {
    return { Authorization: `Bearer ${this.accessToken}` };
}

async trackOrder(orderCode = null, phoneNumber = null) {
    await this.ensureAuthenticated();

    const params = {};
    if (orderCode) params.order_code = orderCode;
    if (phoneNumber) params.phone_number = phoneNumber;

    if (Object.keys(params).length === 0) {
        throw new Error('At least one of orderCode or phoneNumber is required');
    }

    const response = await axios.get(`${this.baseUrl}/track-order/`, {
        params,
        headers: this.getHeaders()
    });

    return response.data;
}

async createOrder(orderData) {
    await this.ensureAuthenticated();

    const response = await axios.post(`${this.baseUrl}/create-order/`,
        orderData,
        { headers: this.getHeaders() }
    );

    return response.data;
}

async estimatePrice(orderData) {
    await this.ensureAuthenticated();

    const response = await axios.post(`${this.baseUrl}/estimate-price/`,
        orderData,
```

```
            { headers: this.getHeaders() }
        );

        return response.data;
    }

    async getCities() {
        await this.ensureAuthenticated();

        const response = await axios.get(`${this.baseUrl}/cities/`, {
            headers: this.getHeaders()
        });

        return response.data;
    }

    async getPickupSlots(serviceType, originCity, destinationCity, maxSlots = 40) {
        await this.ensureAuthenticated();

        const response = await axios.post(`${this.baseUrl}/pickup-slots/`, {
            service_type: serviceType,
            origin_city: originCity,
            destination_city: destinationCity,
            max_slots: maxSlots
        }, {
            headers: this.getHeaders()
        });

        return response.data;
    }
}

// Usage Examples
(async () => {
    const client = new BaroAPIClient('your_business_username', 'your_secure_password');

    try {
        // Example 1: Track order
        const result = await client.trackOrder('12345678');
        console.log(`Found ${result.data.count} order(s)`);

        result.data.orders.forEach(order => {
            console.log(`Order: ${order.order_code}`);
            console.log(`Status: ${order.status_display}`);
            console.log(`From: ${order.origin_city} -> To: ${order.destination_city}`);
            console.log('---');
        });

        // Example 2: Get cities
        const cities = await client.getCities();
```

```
    console.log(`Available cities: ${cities.data.count}`);

    // Example 3: Get pickup slots
    const slots = await client.getPickupSlots('D2D', 'مشهد', 'تهران');
    console.log(`Available slots: ${slots.data.slots.length}`);

    // Example 4: Estimate price
    const priceEstimate = await client.estimatePrice({
        service_type: 'D2D',
        origin_city: 'تهران',
        destination_city: 'مشهد',
        custom_length_cm: 30,
        custom_width_cm: 20,
        custom_height_cm: 15,
        weight_kg: 2.5,
        estimated_value: 500000,
        needs_insurance: true,
        needs_packaging: false,
        is_fragile: true,
        is_unconventional: false
    });
    console.log(`Estimated price: ${priceEstimate.data.total_price} Rials`);

    // Example 5: Create order
    const order = await client.createOrder({
        service_type: 'D2D',
        payment_type: 'ONLINE',
        sender_name: 'علی احمدی',
        sender_phone: '09123456789',
        receiver_name: 'محمد رضایی',
        receiver_phone: '09987654321',
        origin_province: 'تهران',
        origin_city: 'تهران',
        origin_address: '123 خیابان ولیعصر، پلاک',
        origin_lat: 35.6892,
        origin_lng: 51.3890,
        destination_province: 'خراسان رضوی',
        destination_city: 'مشهد',
        destination_address: '456 خیابان امام رضا، پلاک',
        destination_lat: 36.2974,
        destination_lng: 59.6067,
        custom_length_cm: 30,
        custom_width_cm: 20,
        custom_height_cm: 15,
        weight_kg: 2.5,
        estimated_value: 500000,
        needs_insurance: true,
        needs_packaging: false,
        is_fragile: true,
        description: 'کتاب و لوازم التحریر',
```

```
            scheduled_pickup_time: slots.data.slots[0].start_datetime,
            scheduled_pickup_end_time: slots.data.slots[0].end_datetime
        });
        console.log(`Order created: ${order.order_code}`);
        console.log(`Amount: ${order.amount} Rials`);

    } catch (error) {
        console.error('Error:', error.response?.data || error.message);
    }
})();
```

cURL Example

```
# 1. Login
curl -X POST https://barobus.ir/api/b2b/login/ \
  -H "Content-Type: application/json" \
  -d '{
    "username": "your_business_username",
    "password": "your_secure_password"
  }'

# Save the access_token from response
export TOKEN="YOUR_ACCESS_TOKEN_HERE"

# 2. Track Order by order code
curl -X GET "https://barobus.ir/api/b2b/track-order/?order_code=12345678" \
  -H "Authorization: Bearer $TOKEN"

# 3. Track by Phone Number
curl -X GET "https://barobus.ir/api/b2b/track-order/?phone_number=09123456789" \
  -H "Authorization: Bearer $TOKEN"

# 4. Get Cities
curl -X GET "https://barobus.ir/api/b2b/cities/" \
  -H "Authorization: Bearer $TOKEN"

# 5. Get Pickup Slots
curl -X POST "https://barobus.ir/api/b2b/pickup-slots/" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "service_type": "D2D",
    "origin_city": "تهران",
    "destination_city": "مشهد",
    "max_slots": 40
  }'

# 6. Estimate Price
curl -X POST "https://barobus.ir/api/b2b/estimate-price/" \
  -H "Authorization: Bearer $TOKEN" \
```

```
  -H "Content-Type: application/json" \
  -d '{
    "service_type": "D2D",
    "origin_city": "تهران",
    "destination_city": "مشهد",
    "custom_length_cm": 30,
    "custom_width_cm": 20,
    "custom_height_cm": 15,
    "weight_kg": 2.5,
    "estimated_value": 500000,
    "needs_insurance": true,
    "needs_packaging": false,
    "is_fragile": true,
    "is_unconventional": false
  }'

# 7. Create Order
curl -X POST "https://barobus.ir/api/b2b/create-order/" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "service_type": "D2D",
    "payment_type": "ONLINE",
    "sender_name": "علی احمدی",
    "sender_phone": "09123456789",
    "receiver_name": "محمد رضایی",
    "receiver_phone": "09987654321",
    "origin_province": "تهران",
    "origin_city": "تهران",
    "origin_address": "خیابان ولیعصر، پلاک 123",
    "origin_lat": 35.6892,
    "origin_lng": 51.3890,
    "destination_province": "خراسان رضوی",
    "destination_city": "مشهد",
    "destination_address": "خیابان امام رضا، پلاک 456",
    "destination_lat": 36.2974,
    "destination_lng": 59.6067,
    "custom_length_cm": 30,
    "custom_width_cm": 20,
    "custom_height_cm": 15,
    "weight_kg": 2.5,
    "estimated_value": 500000,
    "needs_insurance": true,
    "needs_packaging": false,
    "is_fragile": true,
    "description": "کتاب و لوازم التحریر",
    "scheduled_pickup_time": "2026-02-17T09:00:00+03:30",
    "scheduled_pickup_end_time": "2026-02-17T11:00:00+03:30"
  }'
```

```
# 8. Refresh Token
curl -X POST "https://barobus.ir/api/b2b/refresh-token/" \
  -H "Content-Type: application/json" \
  -d '{
    "refresh_token": "YOUR_REFRESH_TOKEN_HERE"
  }'
```

Best Practices

1. Security

- **Never expose** - your password in client-side code
- **Store credentials** - securely (environment variables, secrets manager)
- **Use HTTPS** - for all API calls
- **Change passwords** - periodically
- **Implement** - proper error handling
- **Ensure requests come from whitelisted IPs only**
- **Notify us immediately** - if your IP addresses change

2. IP Whitelist Management

- **Maintain a list** - of your current whitelisted IPs
- **Use static IP addresses** - for your servers
- **Test from whitelisted IPs** - before going live
- **Contact** support@baro.ir - to update your IP whitelist
- **Keep separate IPs** - for Stage and Production environments

3. Token Management

- **Cache tokens** - to avoid unnecessary login requests
- **Implement** - automatic token refresh
- **Handle token expiration** - gracefully
- **Store tokens securely** - (encrypted storage)
- **Access tokens expire in 1 hour**
- **Refresh tokens expire in 24 hours**

4. Error Handling

- **Implement retry logic** - with exponential backoff
- **Log errors** - for debugging
- **Handle network failures**- gracefully
- **Provide meaningful error messages**- to users
- **Check both**- success field and HTTP status code
- **Parse error_fa**- for Persian error messages

5. Rate Limiting

- **Respect rate limits** - (default: 600 requests/hour)
- **Implement** - request queuing if needed
- **Cache responses** - when appropriate (especially cities list)
- **Monitor** - your API usage
- **Contact us** - if you need higher rate limits

6. Order Creation Workflow

Recommended flow for creating orders:

0. Call /cities/ to get available cities (cache this)
1. Call /estimate-price/ to show price to user before order creation
2. Call /pickup-slots/ to get available pickup times (for D2D/D2T services)
3. Let user select pickup slot
4. Call /create-order/ with all required data
5. Handle payment (if payment_type is ONLINE)
6. Call /track-order/ to monitor order status

7. Data Validation

- **Validate phone numbers** - before sending (format: 09xxxxxxxxx)
- **Ensure sender_phone** - belongs to a registered user
- **Validate coordinates** - are within Iran's boundaries

- **Check package dimensions** - against limits
- **Verify city names match exactly** - (case-insensitive)
- **Use ISO 8601 format** - for datetime fields

8. Performance Optimization

- **Cache cities list** - (updates infrequently)
- **Reuse access tokens** - until expiration
- **Batch operations** - when possible
- **Use appropriate max_slots value** - for pickup slots
- **Implement connection pooling** - for HTTP requests

9. Testing

- **Always test in Stage environment first**
- **Test all service types** - (D2D, D2T, T2D, T2T)
- **Test error scenarios** - (invalid data, expired tokens, etc.)
- **Verify IP whitelist** - before production deployment
- **Test token refresh mechanism**
- **Validate order creation** - with real user accounts

Support

Technical Support

For API access, technical questions, or integration support:

Email: support@baro.ir

Business Inquiries

For partnership opportunities or business questions:

Email: info@baro.ir

Website: https://baro.ir

Response Time

- **Critical Issues** - Within 2 hours
- **General Support** - Within 24 hours
- **Feature Requests** - Within 3 business days

License & Terms

By using Baro's API, you agree to our Terms of Service and Privacy Policy.

Changelog

Version 2.3 (April 29, 2026)

- **Added B2B wallet endpoints**
  - /wallet/balance/
  - /wallet/pay-order/
  - /wallet/pay-batch-order/
- **Documented wallet ownership** mapping via B2B contact_phone

Version 2.2 (April 20, 2026)

- **Added discount endpoints** - (/discount/validate/ & /discount/apply/)
- **Added order and batch order cancellation endpoint**

Version 2.1 (March 16, 2026)

- **Added batch order estimation endpoints** - (/batch-orders/estimate-price/)
- **Added batch order creation endpoints** - (/batch-orders/create/)
- **Added batch order validation endpoints** - (/batch-orders/validate-add-order/)
- **Added compatible cities endpoints** - (/batch-orders/compatible-cities/)
- **Added batch permissions for B2B clients**

Version 2.0 (February 16, 2026)

- **Added order creation endpoint**- (/create-order/)
- **Added price estimation endpoint**- (/estimate-price/)
- **Added cities list endpoint**- (/cities/)
- **Added pickup slots endpoint**- (/pickup-slots/)
- **Enhanced tracking endpoint**- with more order details
- **Added support for all service types**- (D2D, D2T, T2D, T2T)
- **Added comprehensive field validation**
- **Added pickup scheduling support**
- **Updated documentation**- with complete examples

- **Added service types reference section**

Version 1.0 (January 28, 2026)

- **Initial release**
- **Package tracking API (B2B)**
- **JWT authentication**
- **Stage and Production environments**

Need help? Contact our technical team at support@baro.ir

Baro Logistics

B2B API Documentation

support@baro.ir · https://baro.ir