API Documentation

Complete API reference for the SUNSHINE (ON Internal API) system.

- API Documentation (will be separated)

API Documentation (will be separated)

Table of Contents

Overview

The ON Internal API provides RESTful endpoints and WebSocket connections for managing various business operations. All endpoints return JSON responses and use standard HTTP methods.

Base URL

```
Development: http://localhost:3601
Production: https://api.yourdomain.com
```

All endpoints are prefixed with `/api` unless otherwise specified.

Authentication

Most endpoints require authentication using JWT (JSON Web Token).

Obtaining a Token

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "user@example.com",
  "password": "your_password"
}
```

Response:

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "username": "user@example.com",
    "name": "John Doe",
```

```
    "role": "admin"
  }
}
```

Using the Token

Include the token in the `x-access-token` header for protected endpoints:

```
GET /api/auth/account
x-access-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Common Headers

Request Headers

```
Content-Type: application/json
x-access-token: <JWT_TOKEN>  # For authenticated endpoints
```

Response Headers

```
Content-Type: application/json
Cross-Origin-Resource-Policy: cross-origin
```

Response Formats

Success Response

```
{
  "success": true,
  "data": {
    // Response data
  },
```

```
  "message": "Operation successful"
}
```

Error Response

```
{
  "success": false,
  "message": "Error description",
  "errors": [
    {
      "field": "fieldName",
      "message": "Field-specific error"
    }
  ]
}
```

Paginated Response

```
{
  "success": true,
  "data": [...],
  "pagination": {
    "page": 1,
    "perPage": 20,
    "total": 150,
    "totalPages": 8
  }
}
```

Error Codes

Status Code	Description
200	Success
201	Created

Status Code	Description
400	Bad Request (validation error)
401	Unauthorized (authentication required)
403	Forbidden (insufficient permissions)
404	Not Found
500	Internal Server Error

API Endpoints

Authentication Endpoints

POST /api/auth/login

Authenticate a user and receive a JWT token.

Request Body:

```
{
  "username": "user@example.com",
  "password": "password123"
}
```

Response:

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "username": "user@example.com",
    "name": "John Doe",
    "role": "admin"
  }
}
```

Status Codes:

- `200`: Login successful
- `401`: Invalid credentials

GET /api/auth/logout

Logout the current user.

Headers:

```
x-access-token: <JWT_TOKEN>
```

Response:

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

GET /api/auth/account

Get the current authenticated user's account information.

Headers:

```
x-access-token: <JWT_TOKEN>
```

Response:

```
{
  "success": true,
  "data": {
    "id": 1,
    "username": "user@example.com",
    "name": "John Doe",
    "email": "user@example.com",
    "role": "admin",
    "permissions": [...]
```

```
  }
}
```

GET /api/auth/verify-jwt

Verify if the JWT token is valid.

Headers:

```
x-access-token: <JWT_TOKEN>
```

Response:

```
{
  "success": true,
  "message": "Token is valid",
  "user": {
    "id": 1,
    "username": "user@example.com"
  }
}
```

User Management

User Endpoints

Base route: `/api/users`

Available Operations:

- Create user
- Update user
- Delete user
- List users
- Get user by ID

Employee Management

Employee management endpoints handle employee data, shifts, positions, and placements.

Base Routes

- `/api/employees` - Employee CRUD operations
- `/api/positions` - Position management
- `/api/shifts` - Shift management

Common Employee Operations

List Employees

```
GET /api/employees
x-access-token: <JWT_TOKEN>

Query Parameters:
- page (optional): Page number
- limit (optional): Items per page
- search (optional): Search term
```

Get Employee by ID

```
GET /api/employees/:id
x-access-token: <JWT_TOKEN>
```

Create Employee

```
POST /api/employees
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "name": "John Doe",
  "email": "john@example.com",
  "position": "Developer",
  "department": "IT",
```

```
  "phone": "+1234567890",
  "joinDate": "2024-01-01"
}
```

Update Employee

```
PUT /api/employees/:id
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "name": "John Doe Updated",
  "position": "Senior Developer"
}
```

Delete Employee

```
DELETE /api/employees/:id
x-access-token: <JWT_TOKEN>
```

Asset Management

Asset management includes configuration, inventory, and tracking.

Base Routes

- `/api/asset-management/config` - Asset configuration
- `/api/asset-management/inventory` - Inventory management
- `/api/asset-management/tracking` - Asset tracking

Asset Operations

List Assets

```
GET /api/asset-management/inventory
x-access-token: <JWT_TOKEN>
```

Query Parameters:

- status: active, inactive, maintenance
- category: laptop, phone, furniture, etc.
- assignedTo: employee ID

Create Asset

```
POST /api/asset-management/inventory
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "assetCode": "AST-001",
  "name": "MacBook Pro",
  "category": "laptop",
  "serialNumber": "SN123456",
  "purchaseDate": "2024-01-01",
  "value": 2000
}
```

Assign Asset

```
POST /api/asset-management/inventory/:id/assign
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "employeeId": 123,
  "assignDate": "2024-01-15",
  "notes": "Work laptop"
}
```

Track Asset Location

```
GET /api/asset-management/tracking/:assetId
x-access-token: <JWT_TOKEN>
```

Attendance

Attendance system for tracking employee check-ins and check-outs.

Base Routes

- `/api/attendance` - Attendance operations
- `/api/attendance/locations` - Location management

Attendance Operations

Check In

```
POST /api/attendance/check-in
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "employeeId": 123,
  "location": {
    "latitude": -6.2088,
    "longitude": 106.8456
  },
  "photo": "base64_photo_string",
  "timestamp": "2024-01-15T08:00:00Z"
}
```

Check Out

```
POST /api/attendance/check-out
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "employeeId": 123,
  "timestamp": "2024-01-15T17:00:00Z"
}
```

Get Attendance Records

```
GET /api/attendance
x-access-token: <JWT_TOKEN>
```

Query Parameters:

- employeeId: Filter by employee
- startDate: YYYY-MM-DD
- endDate: YYYY-MM-DD
- status: present, absent, late

Attendance Report

GET /api/attendance/report
x-access-token: <JWT_TOKEN>

Query Parameters:

- month: 1-12
- year: YYYY
- departmentId: Filter by department

Leave Management

Leave application and approval system supporting annual, special, and collective leaves.

Base Routes

- `/api/leave` - Leave applications
- `/api/leave/approval` - Approval workflow

Leave Types

- **Annual Leave**: Regular vacation days
- **Special Leave**: Marriage, childbirth, family emergency, etc.
- **Collective Leave**: Company-wide holidays

Leave Operations

Apply for Leave

```
POST /api/leave/apply
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "employeeId": 123,
  "leaveType": "annual",
  "startDate": "2024-02-01",
  "endDate": "2024-02-05",
  "reason": "Family vacation",
  "documents": ["url_to_document"] // Optional
}
```

Get Leave Balance

```
GET /api/leave/balance/:employeeId
x-access-token: <JWT_TOKEN>
```

Response:

```
{
  "success": true,
  "data": {
    "annualLeave": {
      "total": 12,
      "used": 5,
      "remaining": 7
    },
    "specialLeave": {
      "total": 5,
      "used": 1,
      "remaining": 4
    }
  }
}
```

List Leave Applications

```
GET /api/leave
x-access-token: <JWT_TOKEN>
```

Query Parameters:

- status: pending, approved, rejected
- employeeId: Filter by employee
- startDate, endDate: Date range

Approve/Reject Leave

```
POST /api/leave/approval/:leaveId
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "action": "approve", // or "reject"
  "notes": "Approved for vacation"
}
```

Get Leave History

```
GET /api/leave/history/:employeeId
x-access-token: <JWT_TOKEN>
```

Permit Management

Permit system for temporary absence requests (shorter than leave).

Base Routes

- `/api/permit` - Permit applications
- `/api/permit/approval` - Approval workflow

Permit Operations

Apply for Permit

```
POST /api/permit/apply
x-access-token: <JWT_TOKEN>
```

```
Content-Type: application/json

{
  "employeeId": 123,
  "date": "2024-01-15",
  "startTime": "09:00",
  "endTime": "11:00",
  "reason": "Doctor appointment",
  "type": "sick" // or "personal", "official"
}
```

List Permits

```
GET /api/permit
x-access-token: <JWT_TOKEN>

Query Parameters:
- status: pending, approved, rejected
- employeeId: Filter by employee
- date: Specific date
```

Approve/Reject Permit

```
POST /api/permit/approval/:permitId
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "action": "approve",
  "notes": "Medical appointment approved"
}
```

Fleet Management

Driver and vehicle management system.

Base Routes

- `/api/fleet/drivers` - Driver management
- `/api/fleet/auth` - Driver authentication
- `/api/fleet/vehicles` - Vehicle management

Fleet Operations

Register Driver

```
POST /api/fleet/drivers
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "name": "John Driver",
  "licenseNumber": "DL123456",
  "phone": "+1234567890",
  "vehicleType": "truck",
  "licenseExpiry": "2025-12-31"
}
```

Driver Login

```
POST /api/fleet/auth/login
Content-Type: application/json

{
  "username": "driver123",
  "password": "password"
}
```

List Drivers

```
GET /api/fleet/drivers
x-access-token: <JWT_TOKEN>

Query Parameters:
- status: active, inactive
- vehicleType: truck, van, motorcycle
```

Update Driver Location

```
POST /api/fleet/drivers/:id/location
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "latitude": -6.2088,
  "longitude": 106.8456,
  "timestamp": "2024-01-15T10:30:00Z"
}
```

Ticketing

Issue tracking and ticketing system with real-time updates.

Base Routes

- `/api/tickets` - Ticket management
- WebSocket: `/ws/tickets` - Real-time updates

Ticket Operations

Create Ticket

```
POST /api/tickets
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Network issue in office",
  "description": "WiFi not working on 2nd floor",
  "priority": "high", // low, medium, high, urgent
  "category": "IT",
  "reportedBy": 123,
  "attachments": ["url_to_file"]
}
```

Get Tickets

```
GET /api/tickets
x-access-token: <JWT_TOKEN>

Query Parameters:
- status: open, in_progress, resolved, closed
- priority: low, medium, high, urgent
- assignedTo: employee ID
- category: IT, HR, Admin, etc.
```

Update Ticket

```
PUT /api/tickets/:id
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "status": "in_progress",
  "assignedTo": 456,
  "notes": "Working on the issue"
}
```

Add Comment

```
POST /api/tickets/:id/comments
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "comment": "Issue resolved by resetting the router",
  "internal": false // true for internal notes
}
```

Close Ticket

```
POST /api/tickets/:id/close
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "resolution": "Reset router and reconfigured network",
```

```
  "satisfaction": 5 // 1-5 rating
}
```

Meeting Room

Meeting room booking and scheduling system.

Base Routes

- `/api/meeting-rooms` - Room management
- `/api/meeting-rooms/schedule` - Schedule management

Meeting Room Operations

List Meeting Rooms

```
GET /api/meeting-rooms
x-access-token: <JWT_TOKEN>

Query Parameters:
- capacity: Minimum capacity
- floor: Floor number
- facilities: projector, whiteboard, video_conference
```

Book Meeting Room

```
POST /api/meeting-rooms/schedule
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "roomId": 5,
  "title": "Team Standup",
  "date": "2024-01-15",
  "startTime": "09:00",
  "endTime": "10:00",
  "attendees": [123, 456, 789],
```

```
  "description": "Daily team standup meeting"
}
```

Check Room Availability

```
GET /api/meeting-rooms/:id/availability
x-access-token: <JWT_TOKEN>

Query Parameters:
- date: YYYY-MM-DD
```

Cancel Booking

```
DELETE /api/meeting-rooms/schedule/:id
x-access-token: <JWT_TOKEN>
```

Branding Approval

Approval workflow for branding materials with real-time updates.

Base Routes

- `/api/branding-approval` - Approval management
- WebSocket: `/ws/branding-approval` - Real-time updates

Branding Approval Operations

Submit for Approval

```
POST /api/branding-approval
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "New Logo Design",
  "description": "Updated company logo",
  "category": "logo",
```

```
  "files": ["url_to_design_file"],
  "requestedBy": 123,
  "approvers": [456, 789]
}
```

List Approval Requests

```
GET /api/branding-approval
x-access-token: <JWT_TOKEN>

Query Parameters:
- status: pending, approved, rejected, revision
- requestedBy: employee ID
- category: logo, marketing, social_media
```

Approve/Reject

```
POST /api/branding-approval/:id/review
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "action": "approve", // approve, reject, request_revision
  "feedback": "Looks great, approved!",
  "revisionNotes": "" // Required if action is request_revision
}
```

CMS (Content Management System)

Content management for blog, ads, training, careers, and more.

Base Routes

- `/api/cms/blog` - Blog posts
- `/api/cms/ads` - Advertisements

- `/api/cms/training` - Training materials
- `/api/cms/career` - Career/job postings
- `/api/cms/vacancies` - Vacancy listings
- `/api/cms/department` - Department information
- `/api/cms/droppoint` - Drop point locations

Blog Operations

Create Blog Post

```
POST /api/cms/blog
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Company Update 2024",
  "slug": "company-update-2024",
  "content": "Full blog post content here...",
  "excerpt": "Short description",
  "featuredImage": "url_to_image",
  "author": 123,
  "categories": ["company-news", "updates"],
  "tags": ["2024", "announcement"],
  "status": "draft" // draft, published
}
```

List Blog Posts

```
GET /api/cms/blog
x-access-token: <JWT_TOKEN>

Query Parameters:
- status: draft, published
- author: author ID
- category: category slug
- search: search term
```

Update Blog Post

```
PUT /api/cms/blog/:id
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Updated Title",
  "status": "published"
}
```

Advertisement Operations

Create Ad

```
POST /api/cms/ads
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Summer Sale",
  "image": "url_to_image",
  "link": "https://sale.example.com",
  "placement": "homepage_banner", // homepage_banner, sidebar, popup
  "startDate": "2024-06-01",
  "endDate": "2024-08-31",
  "active": true
}
```

Career/Vacancy Operations

Create Job Posting

```
POST /api/cms/vacancies
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "title": "Senior Software Engineer",
  "department": "Engineering",
```

```
  "location": "Jakarta, Indonesia",
  "type": "full-time", // full-time, part-time, contract
  "description": "Job description...",
  "requirements": ["5+ years experience", "Node.js expertise"],
  "salary": "Competitive",
  "closingDate": "2024-02-28",
  "status": "open"
}
```

File Upload

File upload and management endpoints.

Upload Temporary File

```
POST /api/upload/temp
x-access-token: <JWT_TOKEN>
Content-Type: multipart/form-data

Body: file (form data)
```

Response:

```
{
  "success": true,
  "name": "1234567890-filename.pdf",
  "url": "http://localhost:3601/api/temp/1234567890-filename.pdf",
  "status": "done"
}
```

Download Temporary File

```
GET /api/temp/:filename

Query Parameters:
- s=download (optional): Force download instead of display
```

Note: Temporary files are automatically deleted after first download or after 24 hours.

Download Static File

```
GET /api/static/:filename
```

Sales

Sales tracking and management.

Base Routes

- `/api/sales` - Sales operations

Sales Operations

Create Sales Record

```
POST /api/sales
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "customerId": 123,
  "items": [
    {
      "productId": 456,
      "quantity": 2,
      "price": 100
    }
  ],
  "total": 200,
  "salesPerson": 789,
  "date": "2024-01-15"
}
```

Get Sales Report

```
GET /api/sales/report
x-access-token: <JWT_TOKEN>

Query Parameters:
- startDate: YYYY-MM-DD
- endDate: YYYY-MM-DD
- salesPerson: employee ID
- customerId: customer ID
```

Freelance

Freelancer management system.

Base Routes

- `/api/freelance` - Freelancer operations

Freelance Operations

Register Freelancer

```
POST /api/freelance
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "name": "Jane Freelancer",
  "email": "jane@example.com",
  "skills": ["design", "video editing"],
  "hourlyRate": 50,
  "portfolio": "https://portfolio.example.com"
}
```

Assign Project

```
POST /api/freelance/:id/projects
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "projectName": "Website Redesign",
  "description": "Redesign company website",
  "deadline": "2024-03-01",
  "budget": 5000
}
```

Onapps

Application-specific features and voucher management.

Base Routes

- `/api/onapps` - General onapps operations
- `/api/onapps/vouchers` - Voucher management

Voucher Operations

Create Voucher

```
POST /api/onapps/vouchers
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "code": "SAVE20",
  "description": "20% off all orders",
  "discountType": "percentage", // percentage or fixed
  "discountValue": 20,
  "minPurchase": 100,
  "maxDiscount": 50,
  "validFrom": "2024-01-01",
```

```
  "validUntil": "2024-12-31",
  "usageLimit": 1000,
  "active": true
}
```

Validate Voucher

```
POST /api/onapps/vouchers/validate
x-access-token: <JWT_TOKEN>
Content-Type: application/json

{
  "code": "SAVE20",
  "orderAmount": 150
}
```

Response:

```
{
  "success": true,
  "valid": true,
  "discount": 30,
  "finalAmount": 120,
  "message": "Voucher applied successfully"
}
```

WebSocket Endpoints

Branding Approval WebSocket

Connection:

```
const ws = new WebSocket('ws://localhost:3601/ws/branding-approval');

ws.onopen = () => {
  console.log('Connected to branding approval updates');
};
```

```
ws.onmessage = (event) => {
  const data = JSON.parse(event.data);
  console.log('Approval update:', data);
};
```

Message Format:

```
{
  "type": "approval_update",
  "data": {
    "requestId": 123,
    "status": "approved",
    "approver": "John Doe",
    "timestamp": "2024-01-15T10:30:00Z"
  }
}
```

Ticketing WebSocket

Connection:

```
const ws = new WebSocket('ws://localhost:3601/ws/tickets');

ws.onmessage = (event) => {
  const data = JSON.parse(event.data);
  // Handle ticket updates
};
```

Rate Limiting

(Future implementation)

API rate limits:

- 100 requests per 15 minutes per IP address
- 1000 requests per hour for authenticated users

Deprecation Notice

(None currently)

API Version: 1.0.0
Last Updated: 2026-02-04

For additional support or questions about the API, please contact the development team.