API Contract and Integration Guide

Version: 1.0

Base URL: `https://api.tms-system.com/v1`

Audience: Frontend Developers, Mobile Developers, Third-Party Integrators

---

1. Authentication & Security

1.1 Authentication Scheme

We use **Bearer Token Authentication** (JWT).

- **Header:** `Authorization: Bearer <access_token>`
- **Token Lifespan:** 60 minutes for Access Tokens, 7 days for Refresh Tokens.

1.2 Authorization

Top-level permissions are enforced via scopes in the JWT.

- `role:admin` - Super Admin access.
- `role:company_admin` - Bound to a specific `company_id`.
- `role:agent` - Restricted to Sales endpoints.
- `role:driver` - Restricted to Manifest and Tracking endpoints.

---

2. Core Endpoints

2.1 Domain: Identity (`/auth`)

- **POST** `/auth/login`
  - *Request:* `{ "email": "user@example.com", "password": "***" }`
  - *Response:* `{ "access_token": "ey...", "refresh_token": "ey...", "user": { ... } }`
- **POST** `/auth/refresh`
  - *Request:* `{ "refresh_token": "ey..." }`

- *Response:* `{ "access_token": "ey..." }` (New short-lived token)

2.2 Domain: Transport Operations (`/company`)

- **GET** `/company/schedules?from=CityA&to=CityB&date=2025-12-25`
  - *Goal:* Public search for trips.
  - *Response:*

```
[
  {
    "id": "sched_123",
    "departure_time": "2025-12-25T08:00:00Z",
    "price": 5000,
    "available_seats": 24,
    "bus_type": "Luxury"
  }
]
```

- **GET** `/company/schedules/{id}/manifest`
  - *Goal:* Driver gets list of passengers (Requires `role:driver` or `role:admin`).

2.3 Domain: Sales (`/ticketing`)

- **POST** `/ticketing/lock-seat`
  - *Goal:* Temporarily hold a seat before payment.
  - *Request:* `{ "schedule_id": "sched_123", "seat_number": "A1" }`
  - *Response:* `{ "lock_token": "lock_abc...", "expires_in": 300 }` (5 minutes)
- **POST** `/ticketing/book`
  - *Goal:* Finalize booking after payment.
  - *Request:*

```
{
  "lock_token": "lock_abc...",
  "passenger": { "name": "John", "phone": "+250..." },
  "payment_ref": "pay_xyz"
}
```

2.4 Domain: Payments (`/payment`)

- **POST** `/payment/initiate`
  - *Request:* `{ "amount": 5000, "provider": "MTN_MOMO", "phone": "+250..." }`
  - *Response:* `{ "transaction_id": "tx_789", "status": "PENDING_USER_ACTION" }`
- **WEBHOOK** `/payment/callback`
  - *Goal:* Provider notifies us of success/failure.

---

3. Standardization

3.1 Error Handling

All errors follow a standard envelope structure (`RFC 7807` style).

HTTP 400 Bad Request

```
{
  "type": "validation_error",
  "message": "Invalid phone number format",
  "field": "passenger.phone"
}
```

HTTP 409 Conflict

```
{
  "type": "resource_conflict",
  "message": "Seat A1 is already booked"
}
```

3.2 Idempotency

Critical mutation endpoints (Billing, Booking) support Idempotency Keys to prevent duplicate processing during network retries.

- **Header:** `Idempotency-Key: <unique-uuid>`
- **Behavior:** If a client sends the same Request + Key twice, the server returns the *cached original response* for the second request, without re-executing the logic.

3.3 Pagination

List endpoints use cursor-based or page-based pagination.

- **Query Params:** `?page=1&page_size=20`
- **Response Envelope:**

```
{
  "data": [ ... ],
  "meta": { "total": 100, "page": 1, "last_page": 5 }
}
```

3.4 API Versioning

- **Strategy:** URI Versioning (`/v1/...`).
- **Deprecation:** We pledge 6 months notice before decommissioning a major version.
- **Header:** `X-API-Version` can be used for minor feature toggles.

---

4. Integration Guidelines for Partners

1. **Do not poll aggressively.** Use Webhooks for status updates (Payment success, Trip cancellation).
2. **Handle 429 Too Many Requests.** Implement exponential backoff if you hit rate limits.
3. **Secure your keys.** Never expose Admin API Keys in client-side code (Browsers/Mobile Apps).