Blind Navigation & Wayfinding Data-Sharing API Specification

February 2025

Introduction

Purpose

The API's primary role is to **enable seamless data exchange** between various assistive technologies for blind and low-vision users. It aims to create a **common interoperability layer** that connects wayfinding applications, wearables, navigation services, and scene description tools, allowing them to work together in real time.

Scope

- Supports communication between **mobile apps, wearable devices, smart assistants, and cloud-based services**.
- Facilitates **real-time navigation, object recognition, scene descriptions, and location-based assistance**.
- Ensures secure and standardized **data handling, privacy controls, and user authentication**.

Contact: darryl.adams@accessinsights.net

Audience

- **Developers** building navigation and assistive applications.
- **Product managers** defining accessibility strategies.
- **Industry stakeholders** looking to improve **interoperability in assistive tech**.
- **Accessibility researchers and standards bodies** interested in advancing **universal wayfinding solutions**.

Background and Rationale

Assistive technologies for blind and low-vision users have **expanded significantly**, but they often operate in **isolated ecosystems**, limiting their effectiveness when combined.

- **Fragmentation Problem**: Each service (e.g., navigation apps, object recognition tools, and scene description systems) uses its own **proprietary data formats and communication protocols**, preventing seamless integration.
- **Lack of Interoperability**: Users must **switch between multiple apps** manually, leading to inefficiencies and potential safety risks when navigating.
- **Limited Data Sharing**: Technologies that could **complement each other** (e.g., Smart glasses recognizing text and a GPS or wayfinding app providing contextual location guidance) are not currently **designed to share data in real time**.

This API proposes a **standardized, open framework** to enable diverse applications and devices to communicate efficiently, **providing a more unified and accessible experience**.

API Objectives

The API is designed to address the **fragmentation issue** in assistive technologies by ensuring **seamless communication** between devices and applications. The key objectives are:

Interoperability

- Establish a **common data-sharing protocol** that allows diverse applications (e.g., navigation, object recognition, live assistance) to exchange information.
- Support integration with **wearable devices, smartphones, smart assistants, and cloud-based services**.
- Define **standardized data formats** for user location, object recognition, scene descriptions, and navigation instructions.

Scalability

- Architected to handle a **growing ecosystem** of services and devices.

- Supports **varied levels of integration**, from simple API calls to deep service-to-service connectivity.
- Designed for **future expansion** to accommodate **emerging assistive technologies**.

Security and Privacy

- Implements **OAuth 2.0 authentication** for secure access.
- Enforces **end-to-end encryption** (TLS for transit, AES for stored data).
- Incorporates **user-controlled data sharing**, allowing individuals to **opt in or out** of specific services.
- Ensures compliance with **GDPR, CCPA, and other data privacy regulations**.

Real-Time Performance

- Enables **low-latency data exchange** to support live navigation and scene descriptions.
- Uses **websockets** or other real-time communication mechanisms where necessary.
- Optimized for **edge processing** in wearable and mobile devices to minimize delays.

Monetization & Value-Added Services (Optional)

- ### Usage-Based Pricing
    - Offer **tiered access** based on API call volume, response latency, or premium features.
    - Free access for basic services (e.g., location sharing), with paid options for advanced features (e.g., AI-enhanced scene descriptions).
- ### Context-Specific Data Access
    - Companies can **monetize high-value datasets** (e.g., enhanced POI data, accessibility-specific maps, advanced object recognition models).
    - Enable **permissions-based data sharing** where businesses can purchase **aggregated, anonymized insights** while maintaining user privacy.
- ### Revenue-Sharing Model
    - Allow third-party developers and device manufacturers to **earn revenue** by contributing **enhanced data streams** or value-added services (e.g., AI-powered wayfinding).
    - Facilitate partnerships where assistive tech companies **license proprietary datasets** (e.g., high-accuracy indoor navigation maps) via the API.

Contact: darryl.adams@accessinsights.net

- Enterprise Integrations
    - Offer **premium API plans** for enterprise customers, such as **transportation services, smart city initiatives, and commercial wayfinding solutions** that leverage the API for accessibility enhancements.

Technical Specifications

The API is designed using a **RESTful architecture** with **JSON-formatted requests and responses**, ensuring broad compatibility and ease of integration.

Architecture

- **RESTful API** with standardized HTTP methods:
    - GET for retrieving data (e.g., user profile, navigation instructions).
    - POST for submitting data (e.g., sharing location updates).
    - PUT for updating records (e.g., modifying user preferences).
    - DELETE for removing records (e.g., deleting a device pairing).
- **WebSockets** support for real-time data exchange (e.g., live navigation guidance).
- Cloud-hosted, with the option for **edge computing** to reduce latency on mobile and wearable devices.

Authentication & Security

- OAuth 2.0 for secure, token-based authentication.
- API key-based access for approved enterprise integrations.
- Role-based access control (RBAC) to manage data permissions:
    - **User**: Access to personal data and device interactions.
    - **Service Provider**: Restricted access to shared datasets (e.g., wayfinding apps accessing navigation data).
    - **Enterprise Partner**: Monetized access to anonymized, aggregate data (if opted in).
- End-to-end encryption
    - **TLS/SSL** for data in transit.
    - **AES-256** for data at rest.

- Privacy controls
  - User-consent management for **data sharing**.
  - Compliance with **GDPR, CCPA, and other privacy laws**.

API Endpoints

Category	Endpoint	Function
User Management	`POST /users/register`	Register a new user
	`POST /users/authenticate`	Authenticate a user via OAuth 2.0
	`GET /users/profile`	Retrieve user profile & preferences
Data Exchange	`POST /data/location`	Share real-time location data
	`GET /data/navigation`	Retrieve step-by-step navigation instructions
	`POST /data/object-recognition`	Submit identified objects for processing
	`GET /data/scene-description`	Retrieve AI-generated scene descriptions
Device Communication	`POST /devices/connect`	Pair a wearable device (e.g., smart glasses)
	`GET /devices/status`	Check connected device status
	`DELETE /devices/disconnect`	Unpair a device
Monetization & Analytics	`GET /analytics/usage`	Retrieve API usage statistics
	`POST /monetization/access`	Purchase access to premium datasets
	`GET /monetization/subscriptions`	Manage paid API plans

Data Models

User Profile

```
json

{

  "user_id": "12345",
```

```
  "name": "John Doe",
  "preferences": {
    "navigation_assistance": true,
    "object_recognition": false
  },
  "permissions": {
    "data_sharing": "opt-in",
    "third_party_access": "limited"
  }
}
```

Location Data

```
json
{
  "user_id": "12345",
  "timestamp": "2025-02-27T14:30:00Z",
  "coordinates": {
    "latitude": 37.7749,
    "longitude": -122.4194
  },
  "context": "Crosswalk detected ahead"
}
```

Navigation Instructions

```
json
{
  "route_id": "67890",
  "steps": [
    {"instruction": "Head north for 50 meters", "landmark": "Coffee shop on
the right"},
    {"instruction": "Turn left onto Main Street", "landmark": "Bus stop
ahead"}
```

```
  ]
}
```

Object Recognition

```
json
{
  "device_id": "glasses_001",
  "timestamp": "2025-02-27T14:32:00Z",
  "objects": [
    {"name": "Stop sign", "confidence": 95, "position": "3 o'clock"},
    {"name": "Pedestrian", "confidence": 90, "position": "11 o'clock"}
  ]
}
```

Scene Description

```
json
{
  "description_id": "scene_123",
  "timestamp": "2025-02-27T14:35:00Z",
  "summary": "You are in a park. There are trees, a bench on the left, and a walking path ahead."
}
```

Data Privacy and Security

Ensuring **user privacy and data security** is critical for an API handling sensitive navigation and accessibility data. The following measures will protect user information while maintaining system integrity.

Data Encryption

- **Transport Layer Security (TLS 1.3)** ensures all data transmitted between clients and servers is encrypted.

- **AES-256 encryption** is applied to all data stored at rest, protecting user information in case of a data breach.
- **End-to-end encryption (E2EE)** is available for real-time data exchanges between devices and services.

User Consent & Access Control

- **Explicit Opt-in Model**: Users must **grant explicit permission** before sharing location, object recognition, or scene description data.
- **Granular Permissions**: Users can define which types of data (e.g., location, navigation history, object recognition) are shared and with whom.
- **Revocable Access**: Users can **revoke third-party access** at any time via API or app settings.

Data Anonymization

- **Location Blurring**: To prevent precise tracking, the API can apply **fuzzy location obfuscation** in anonymized datasets.
- **Aggregated Analytics**: Data used for **monetization or research** is de-identified and aggregated to **prevent individual tracking**.
- **On-Device Processing**: Whenever possible, object recognition and scene descriptions are processed **locally on the device** to minimize data transmission.

Compliance with Global Privacy Regulations

The API will be designed to comply with major privacy regulations, including:

- **General Data Protection Regulation (GDPR)** – Right to access, correct, and delete personal data.
- **California Consumer Privacy Act (CCPA)** – Users can opt-out of data sales and request data deletion.
- **Health Insurance Portability and Accountability Act (HIPAA)** – If integrated into healthcare-related applications, the API ensures compliance for medical data handling.
- **Web Content Accessibility Guidelines (WCAG 2.1 & 2.2)** – Ensures API documentation and implementation resources are accessible.

Secure API Access & Authentication

- **OAuth 2.0 Authorization**: Ensures secure authentication via industry-standard token-based access.
- **Role-Based Access Control (RBAC)**: Limits API access based on user type (e.g., **individual users, assistive services, enterprise partners**).

- **Automatic Token Expiry & Refresh**: Limits the risk of unauthorized access via expired tokens.

Data Retention & Deletion Policies

- **Short-Term Storage**: Personally identifiable data is **not stored beyond necessary processing time** (e.g., real-time navigation data expires after 24 hours).
- **User-Controlled Deletion**: Users can delete their data via API or app settings.
- **Audit Logs**: Maintain logs of API access requests to enhance security monitoring.

Integration with Existing Services

The API is designed to interoperate with a variety of assistive technologies, navigation platforms, and wearable devices. It provides standardized data exchange protocols to ensure seamless communication between different solutions.

Live Assistance Platforms

Functionality: Services that connect blind and low-vision users with human or AI-based assistance via **real-time video, audio, or text communication**.

Integration Points:

- API support for **initiating assistance requests** from compatible applications.
- Secure **contextual data sharing** (e.g., location, object recognition results).
- Optional callback hooks to store **session metadata for future reference**.

Wearable Assistive Devices

Functionality: Smart glasses, augmented reality (AR) headsets, and mobile-based solutions that provide **object recognition, text reading, and environmental awareness**.

Integration Points:

- API endpoints for **streaming recognized text and object data**.
- Data exchange mechanisms for **augmenting scene descriptions with spatial audio cues**.
- Standardized pairing and **device management protocols**.

Navigation & Wayfinding Services

Functionality: Outdoor and indoor navigation solutions, including GPS-based pedestrian guidance, transit accessibility tools, and indoor mapping platforms.

Integration Points:

- Standardized data format for turn-by-turn pedestrian navigation.
- API compatibility with indoor positioning systems for venue-specific wayfinding.
- Support for real-time obstacle detection and path optimization.

AI-Powered Assistance & Smart Assistants

Functionality: AI-based services that provide speech-to-text transcription, object recognition, and contextual awareness through voice or camera input.

Integration Points:

- Support for voice-activated navigation queries.
- Integration with screen readers and AI-powered scene description tools.
- API endpoints for real-time audio feedback on environmental context.

Public Transit Data Platforms

Functionality:
Real-time transit information systems providing accessible routes, schedules, and service alerts.

Integration Points:

- API support for fetching real-time bus, train, and transit schedule updates.
- Support for multimodal journey planning across different transit networks.

Ride-Share and Taxi Services

Functionality:
Ride-hailing platforms that offer accessible transport solutions, including wheelchair-accessible vehicles (WAVs).

Integration Points:

- API hooks for booking and managing accessible ride options.
- Real-time tracking of ride status and estimated time of arrival (ETA).
- Integration with third-party mobility-as-a-service (MaaS) platforms.

Autonomous Vehicle Services

Functionality:
Self-driving vehicle platforms offering on-demand transport for blind and low-vision users.

Integration Points:

- API access to request and schedule autonomous rides.
- Audio-guided assistance for safe boarding and exiting.
- Secure data exchange for personalized travel preferences and route planning.

Error Handling and Response Codes

The API follows standard HTTP status codes along with custom error messages for clarity.

Standard HTTP Status Codes

Status Code	Meaning	Usage Scenario
200 OK	Success	Request was successful, and the response contains the requested data.
201 Created	Resource Created	Successfully created a new resource (e.g., user profile, device connection).
204 No Content	Success (No Data)	Request was successful, but no content is returned (e.g., deleting a resource).
400 Bad Request	Client Error	The request was malformed or contained invalid parameters.
401 Unauthorized	Authentication Required	The user must authenticate before accessing the resource.
403 Forbidden	Permission Denied	The user does not have the required permissions.
404 Not Found	Resource Missing	The requested resource does not exist.
409 Conflict	Data Conflict	Conflict in request processing (e.g., attempting to register an already existing user).
429 Too Many Requests	Rate Limit Exceeded	The client has exceeded the allowed API request limit.
500 Internal Server Error	API Error	A server-side error occurred.
503 Service Unavailable	API Unreachable	The API is temporarily down for maintenance or overload.

Custom Error Messages & Examples

Each error response includes a structured JSON message to help developers understand the issue.

Example: 400 Bad Request (Invalid Input)

```
json

{
  "error": {
    "code": 400,
    "message": "Invalid coordinates format. Latitude and longitude must be
numerical values.",
    "details": {
      "parameter": "coordinates",
      "expected_format": "latitude: float, longitude: float"
```

```
    }
  }
}
```

Example: 401 Unauthorized (Invalid Token)

```
json
{
  "error": {
    "code": 401,
    "message": "Invalid API token. Please authenticate using OAuth 2.0.",
    "documentation_url": "https://api.example.com/docs/authentication"
  }
}
```

Example: 403 Forbidden (Access Denied)

```
json
{
  "error": {
    "code": 403,
    "message": "Access denied. Your API key does not have permission to
access this resource."
  }
}
```

Example: 429 Too Many Requests (Rate Limit Exceeded)

```
json
{
  "error": {
    "code": 429,
    "message": "Rate limit exceeded. You have made 500 requests in the
last hour.",
    "retry_after": 3600
  }
}
```

Error Logging & Debugging Support

- **Error Logs**: API automatically logs error occurrences for debugging purposes.
- **Request IDs**: Each API request receives a **unique identifier** in error responses to help track issues.
- **Developer Support**: A diagnostic endpoint (GET /status) allows developers to test API connectivity.

Rate Limiting and Throttling

To ensure fair usage and prevent **API abuse**, the system enforces **rate limits** and **throttling mechanisms** based on user roles, request types, and subscription tiers.

Rate Limiting Policies

The API enforces the following request limits:

User Type	Request Limit	Time Window
Free Users	100 requests	per hour
Registered Users	1,000 requests	per hour
Assistive Services	10,000 requests	per hour
Enterprise Partners	Custom (negotiable)	per hour

- Burst Limit: A temporary allowance of double the rate limit for 5 minutes to handle spikes.
- Rate Decay: If an application exceeds limits, it enters a cooldown period before normal access resumes.
- Fair Use Protection: Requests that abuse system resources (e.g., repeated failed authentication attempts) will trigger an automated cooldown or temporary IP block.

Response Headers for Rate Limits

Each API response includes headers to inform clients about their rate limit status:

```
http

X-RateLimit-Limit: 1000

X-RateLimit-Remaining: 250

X-RateLimit-Reset: 1712236800  (Unix timestamp of reset time)
```

Example: 429 Too Many Requests Response

json

```
{

 "error": {

  "code": 429,
```

```
  "message": "Rate limit exceeded. Please wait before making more requests.",
  "retry_after": 3600  // Time in seconds before reset
 }
}
```

Throttling Strategies

To prevent **server overload** and **ensure equitable access**, the API uses multiple throttling methods:

- **Per-IP Throttling**: Limits excessive requests from a single IP address.
- **Per-User Throttling**: Ensures fair distribution across different user accounts.
- **Adaptive Throttling**: Adjusts dynamically based on server load and API demand.

Premium & Monetized Access

- Enterprise partners can **purchase higher rate limits** as part of premium API plans.
- A **pay-per-use model** is available for services that require **higher data access thresholds**.

Versioning and Updates

To ensure long-term maintainability and backward compatibility, the API follows a structured versioning and update policy.

Versioning Strategy

The API follows **semantic versioning** (MAJOR.MINOR.PATCH), where:

- **MAJOR** – Breaking changes (e.g., removing an endpoint, changing response structure).
- **MINOR** – Backward-compatible feature additions (e.g., new endpoints, optional parameters).
- **PATCH** – Bug fixes and minor performance improvements.

The **version number is included in the API URL**:

```
http
https://api.example.com/v1/navigation
```

Deprecation Policies

- Older API versions will remain **supported for at least 12–18 months** before deprecation.
- Clients will receive **advanced notifications** of deprecated features via:
- API response headers (X-Deprecated-Version: true, X-Retirement-Date: YYYY-MM-DD).
- Email alerts for registered developers.

- API status dashboard updates.

Example: Deprecation Warning in Response Header

```
http

X-Deprecated-Version: true

X-Retirement-Date: 2026-06-01
```

Update Release Process

Stable Releases (v1, v2, etc.) – Publicly available and recommended for production use.

Beta Versions (v1.1-beta or v2-beta) – Early access for developers to test new features.

Experimental Features – Enabled via opt-in flags (?enable_experimental=true).

Migration Guidelines

Detailed migration guides will be provided when breaking changes occur.

Testing and Validation

To ensure **reliability, compatibility, and ease of integration**, the API provides a structured testing and validation framework for developers.

Sandbox Environment

A dedicated **sandbox API** allows developers to test integrations **without affecting live data**.

Simulated responses help developers **validate workflows before production deployment**.

Accessible via:

```
http

https://sandbox.api.example.com/v1/
```

API Validation Tools

- **Request Validator**: Ensures API requests comply with required formats and parameters.
- **Response Inspector**: Helps developers verify **data consistency** in API responses.
- **Mock Data Generator**: Allows testing with **realistic but dummy data** (e.g., navigation scenarios, object recognition results).

Example: API Validation Response

```
json

{

  "validation": {

    "status": "error",
```

```
    "message": "Missing required parameter: coordinates",
    "expected_format": "latitude: float, longitude: float"
  }
}
```

Automated Testing

- **Unit Tests**: Validate core API functions (e.g., authentication, data exchange).
- **Integration Tests**: Verify interactions between different services (e.g., navigation + object recognition).
- **Performance Testing**: Ensures API can handle high traffic **without performance degradation**.

Developer-Focused Features

- **Interactive API Explorer**: Web-based tool for testing API endpoints **without writing code**.
- **Postman Collection**: Pre-configured API requests for developers to test in **Postman or similar tools**.
- **CI/CD Compatibility**: API validation can be integrated into **continuous integration (CI) pipelines**.

Documentation and Support

To ensure smooth adoption and integration, the API includes **comprehensive documentation and developer support resources**.

Comprehensive Developer Documentation

API Reference Guide:

- Detailed explanations of endpoints, request parameters, response formats, and error handling.
- Code examples in multiple programming languages (e.g., Python, JavaScript, Java).

Quick Start Guide

- Step-by-step instructions for setting up authentication and making the first API call.

Use Case Tutorials

- Guides for **common integrations** (e.g., implementing navigation assistance, accessing scene descriptions).

Changelog

- Tracks updates, new features, and deprecations.

Developer Portal

A **centralized hub** where developers can:

- **Generate API keys** and manage authentication credentials.
- **Access sandbox testing tools**.
- **View API usage analytics** (e.g., request logs, rate limits).

Community & Support

- Discussion Forum: A space for developers to ask questions, share best practices, and troubleshoot issues.
- Live API Status Page: Real-time updates on downtime, maintenance, and incident reports.

Support Tiers:

- **Basic Support**: Free tier with forum access and community-driven assistance.

Interactive Tools

- API Explorer: Allows developers to test API calls directly from the documentation.
- Code Snippet Generator: Generates ready-to-use API request code based on selected parameters.

Future Enhancements

To ensure the API remains adaptable and forward-thinking, future enhancements will focus on AI-driven capabilities, expanded device support, and user-driven improvements.

AI Integration

Machine Learning for Predictive Assistance

- AI-powered **route recommendations** based on user behavior and preferences.
- Dynamic **hazard detection** (e.g., detecting sudden obstacles from crowd-sourced data).

Context-Aware Scene Descriptions

- AI-enhanced scene analysis for more natural, detailed audio descriptions.
- Personalization based on user preferences and environmental factors.

Expanded Device & Service Compatibility

Wearables & Smart Glasses

- Future support for next-gen AR devices that integrate real-time AI-assisted navigation.

Integration with Smart Cities & Public Infrastructure

- Standardized data exchange with transportation systems, traffic signals, and IoT accessibility devices.
- APIs for **real-time transit assistance** and dynamic pedestrian routing.

Enhanced Privacy & Security

Decentralized Identity Management

- Blockchain-based user authentication for **enhanced data security**.

Federated Learning for Personalization

- AI models that improve **without sharing raw user data**, preserving privacy.

User Feedback & Community Involvement

Crowdsourced Data Contributions

- Users can **report accessibility obstacles** (e.g., broken sidewalks, construction zones) to improve navigation.

Open-Source SDK Contributions

- Community-driven improvements to **enhance API flexibility** for diverse applications.

Standardization & Industry Adoption

Collaboration with Accessibility Standards Bodies

- Work with **W3C, CTA, and assistive tech organizations** to formalize industry-wide adoption.

Open API Governance

- Establish a **working group** to guide API evolution based on real-world feedback.

//end of document

Contact: darryl.adams@accessinsights.net