Link Box AI Platform: API Capabilities and Developer Ecosystem Briefing

Executive Summary

The Link Box AI Platform provides a comprehensive, versioned RESTful API (v1.0) that enables programmatic access to a multifaceted personal data ecosystem. The API is not limited to knowledge management; it integrates functionalities for managing health, finance, and calendar data, positioning it as a central hub for personal information automation. Core capabilities include full CRUD (Create, Read, Update, Delete) operations for knowledge entries, rule-based "Smart Collections" for dynamic organization, and powerful semantic search with granular filtering.The developer experience is a clear priority, demonstrated by robust dual authentication methods (API Keys for personal use, OAuth 2.0 for third-party applications), detailed security best practices, and a tiered rate-limiting system tied to user subscription plans (Free, PRO, Enterprise). The platform architecture encourages efficient integration through real-time event notifications via a secure webhook system, reducing the need for polling. This is complemented by official SDKs for Python and Node.js, community-supported libraries, comprehensive error handling with standard HTTP codes, and dedicated support channels, establishing a robust framework for building custom integrations and workflows.

1. Core Platform Capabilities via API

The Link Box AI API provides extensive control over the platform's primary data domains, enabling sophisticated integrations and automations. The Base URL for all API v1.0 endpoints is https://api.linkbox.ai/v1.

1.1. Knowledge Management

The API offers powerful tools for managing a user's knowledge base, centered around "Entries" and their organization into "Collections."**Entries:**

- **CRUD Operations:** The API supports full Create, Read (single or list), Update, and Delete operations for entries.
- **Content Types:** Entries can be filtered by type, including note, article, photo, and video.
- **Advanced Filtering & Sorting:** When listing entries, the API allows for extensive querying with parameters such as:
- Pagination (limit, offset)
- Sorting (created_at, updated_at, title in asc or desc order)
- Filtering by collection_id or date range (from_date, to_date)**Collections:**
- **Manual Collections:** Users can create standard collections and programmatically add or remove entries.
- **Smart Collections:** A key feature is the ability to create "Smart Collections" that automatically populate based on user-defined rules. The API supports a rich set of rule operators for defining this logic, including:
- **Text:** contains, not_contains, equals, not_equals, starts_with, ends_with
- **Date:** before, after
- **Numeric:** greater_than, less_than

1.2. Semantic Search

The platform exposes a dedicated endpoint for performing semantic searches across the knowledge base. This goes beyond simple keyword matching, allowing for more intuitive and context-aware queries.

- **Search Filtering:** Search results can be refined using a variety of filters:
- type: An array of entry types to include.
- date_range: Pre-defined ranges (last_7_days, last_year) or custom start/end dates.
- collections: An array of collection IDs to search within.
- tags: An array of tags to match.
- platform: Filter by the source platform of the entry.
- min_relevance: A score from 0 to 1 to filter out less relevant results.

1.3. Personal Data Integration

A significant aspect of the API is its ability to interact with personal data beyond knowledge management.

- **Health Data:** The /health endpoints allow applications to:
- Retrieve health summaries (including steps, calories, workouts, sleep) for specific date ranges.
- Log new meals and workouts directly to the user's account.
- **Financial Data:** The /finance endpoints provide programmatic access to financial information:
- List financial accounts.
- Retrieve transactions with filters for account, date range, category, and amount.
- Update existing transaction details.
- Fetch monthly budget summaries.
- **Calendar Data:** The /calendar endpoints enable interaction with scheduling:
- Retrieve calendar events within a specified date range and for specific calendar IDs.
- Create new events programmatically.

2. Developer Experience and API Architecture

The API is designed with RESTful principles, ensuring predictable and standardized interactions. It exclusively uses JSON for requests and responses and requires all connections to be made over HTTPS.

2.1. Authentication Mechanisms

The platform provides two distinct authentication schemes tailored to different use cases, with a strong emphasis on security.

- **API Keys (Personal Use):**
- **Generation:** Keys are generated within the user's settings and are displayed only once for security.
- **Best Practices:** The documentation explicitly recommends treating keys like passwords, storing them in environment variables (LBX_API_KEY), never committing

them to version control, rotating them every 90 days, and creating separate keys for each application.

- **OAuth 2.0 (Third-Party Applications):**
- **Flow:** Utilizes the standard Authorization Code Flow for secure user authorization.
- **Granular Scopes:** Allows applications to request specific permissions, ensuring users only grant necessary access. Available scopes include:
- entries:read, entries:write
- collections:read, collections:write
- health:read, finance:read
- user:read, user:write

2.2. Rate Limiting and Usage Tiers

API access and request volume are governed by the user's subscription plan. This structure links API usage directly to the platform's business model.| Plan | Requests/Hour | Requests/Day || ------ | ------ | ------ || Free | 100 | 1,000 || **PRO** | 1,000 | 10,000 || Enterprise | Custom | Custom |

- **Rate Limit Headers:** Every API response includes headers to help applications manage their request rate programmatically:
- X-RateLimit-Limit: The total request quota for the current window.
- X-RateLimit-Remaining: The number of requests left in the window.
- X-RateLimit-Reset: A Unix timestamp indicating when the quota will reset.
- **Exceeding Limits:** A 429 Too Many Requests HTTP status code is returned when a limit is exceeded. The documentation recommends best practices such as implementing exponential backoff, caching responses, and using webhooks over polling.

3. Real-Time Integration via Webhooks

The platform offers a robust webhook system to enable real-time notifications, allowing applications to react instantly to events without resorting to inefficient polling.

- **Setup:** Webhooks are configured in the developer settings by providing a secure (HTTPS) endpoint URL and subscribing to specific events.
- **Security:** To ensure authenticity, every webhook request is sent with a signature in the X-LinkBox-Signature header. The documentation provides a Python example for verifying this signature using a shared secret.
- **Reliability:** The system has built-in retry logic for failed deliveries. It will attempt to resend a notification three times with exponential backoff (1 min, 10 min, 1 hour). A webhook is automatically disabled after 10 consecutive failures.

Supported Webhook Events

Event,Description
entry.created,A new entry has been added.
entry.updated,An existing entry has been modified.
entry.deleted,An entry has been deleted.
collection.created,A new collection has been created.
collection.updated,A collection has been modified.
health.meal_logged,A meal has been added to the log.

health.workout_logged,A workout has been completed.
finance.transaction_added,A new transaction has been synced.
calendar.event_created,A calendar event has been created.
calendar.event_upcoming,A calendar event is starting soon.

4. Tooling, Support, and Error Handling

The platform provides a comprehensive ecosystem to support developers throughout the integration process.

4.1. SDKs and Libraries

To accelerate development, Link Box AI provides both official and community-maintained SDKs.

- **Official SDKs:**
- Python
- Node.js
- **Community Libraries:**
- Ruby (linkbox-ruby)
- Go (go-linkbox)
- PHP (linkbox-php)

4.2. Error Handling

The API uses standard HTTP status codes for communicating the outcome of a request. Error responses follow a consistent JSON format, providing a message and an error_code.| Code | Meaning | Description || ------ | ------ | ------ || 200 | OK | The request was successful. || 201 | Created | The resource was successfully created. || 204 | No Content | The request was successful, but there is no body. || 400 | Bad Request | The request was malformed or had invalid parameters. || 401 | Unauthorized | The API key is missing, invalid, or expired. || 403 | Forbidden | The authenticated user does not have permission. || 404 | Not Found | The requested resource does not exist. || 429 | Too Many Requests | The rate limit has been exceeded. || 500 | Internal Server Error | An unexpected error occurred on the server. |

4.3. Support and Community Channels

A multi-channel support system is available for developers:

- **Technical Support:** api@bb23llc.com
- **Documentation:** docs.linkbox.ai
- **System Status:** status.linkbox.ai
- **Bug Reports:** GitHub Issues (github.com/bb23llc/linkbox-api)
- **Security Vulnerabilities:** security@bb23llc.com
- **Feature Requests & Roadmap:** community.linkbox.ai/api and roadmap.linkbox.ai