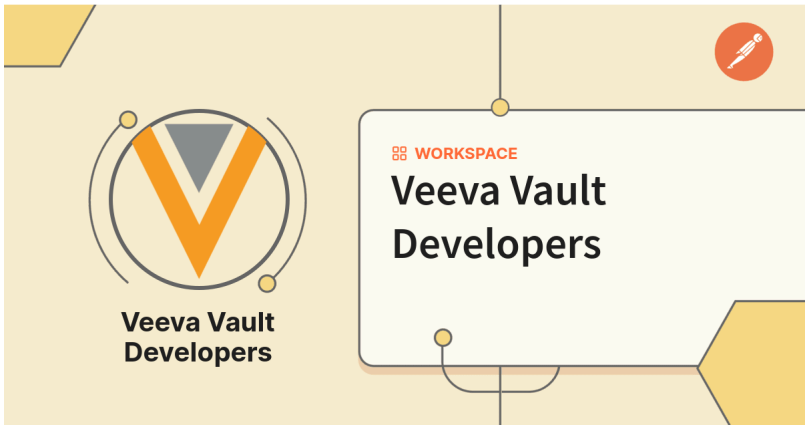
Veeva Ecosystem APIs: Comprehensive Technical Overview

By IntuitionLabs • 4/11/2025 • 35 min read

veeva api integration vault crm network opendata rest soap sdk life-sciences technical



Veeva Ecosystem APIs: Comprehensive Technical Overview

Introduction

Veeva Systems provides a suite of cloud-based applications for the life sciences industry. Key products include **Veeva Vault** (a content and data management platform), **Veeva CRM** (customer relationship management for pharma sales, built on Salesforce), and **Veeva Network** (master data management for healthcare provider data), among others. Each of these products offers robust APIs for integration and extension. This document details all types of APIs available across the Veeva ecosystem – including REST, SOAP, Bulk APIs, and SDKs – explaining their usage, authentication, versioning, and integration scenarios. All examples and information are drawn from official Veeva documentation and developer resources.

Veeva Vault Platform APIs

Overview: Veeva Vault is a cloud content management and business process platform used for managing documents and structured data (objects) in various domains (e.g. clinical, regulatory, quality). Vault provides a unified REST API that allows software to import, update, and retrieve documents or object records programmatically (About Vault API - Veeva Vault Help). The Vault API enforces the same business rules and permissions as the Vault web UI, ensuring that integrations respect security and lifecycle controls (About Vault API - Veeva Vault Help). Vault's APIs enable integrations such as document publishing, data synchronization with other systems, and custom application development on the Vault platform.

Vault REST API (CRUD Operations)

Vault's primary interface is a RESTful API. It exposes endpoints to **create, read, update, and delete** both **documents** (unstructured content with metadata) and **object records** (structured data similar to database records). Common use cases include uploading new documents, updating metadata, retrieving document content, querying records, and more (About Vault API - Veeva Vault Help). All interactions are over HTTPS and require authentication (session token or OAuth, described below). The API accepts and returns JSON by default.

- **Endpoint structure:** Vault URLs include the Vault domain, API version, and resource. For example:

```
https://yourvault.veevavault.com/api/v20.3/objects/documents/{doc_id}
```

would target a specific document (where `v20.3` is the API version). Custom object records are accessed similarly via their object name. Vault releases several API versions per year (e.g. `v21.3`, `v22.1`, etc.), described in **Versioning** below.

- **Example – Retrieving metadata:** For instance, an integration can retrieve Vault object definitions via the API. A GET request to the metadata endpoint (with a valid session) might be:

  ```
  GET /api/v21.3/metadata/objectTypes HTTP/1.1
  Authorization: {SESSION_ID}
  ```

  This returns a JSON list of available object types. For example, it might list standard objects like *Document* or custom objects defined in that Vault, each with a name and description (About the Network API) (About the Network API).

- **Document content:** Document files (like PDFs) can be downloaded via the REST API (e.g. GET on a document's `/versions/{version}/file` resource) and uploaded via a multipart POST. Vault uses a token-based upload for large files – for example, first obtaining an upload URL/token (`POST /objects/documents/tokens`), then PUTting the file bytes. This ensures reliable file transfer and virus scanning.

- **Bulk record operations:** Vault's API supports **bulk operations** to improve efficiency. A single API call can create or update many records at once (up to 500 records in one request) (API Reference). For example, the "Create Multiple Object Records" endpoint allows inserting a batch of new records in one HTTP call (API Reference). Bulk APIs are available for many (though not all) resource types, and using them can drastically reduce API call count. Vault automatically applies appropriate validations to each record in a batch and returns detailed results for each. If a resource doesn't have a bulk API, clients must send one request per record.

Vault Query Language (VQL) and Query API

For retrieving data with complex filters or joins, Vault provides the **Vault Query Language (VQL)** – a SQL-like query language. The REST API offers a **`/query`** endpoint where clients can submit a VQL query string and get JSON results. This is analogous to querying a database or Salesforce's SOQL. For example, a client could POST a query like:

```
SELECT name__v, status__v FROM object_name__c WHERE status__v = 'Active'
```

to `/api/{version}/query`, and Vault will return matching records. VQL supports filtering, projections, and even some joins across documents and related objects (How to Obtain Documents with Attachments Through Vault Query ...). This is useful for reading large sets of data without multiple round trips. The query API returns data in pages if the result set is large, and the client can iterate through page results.

Authentication & Security (Vault)

All Vault API calls must be authenticated. Vault supports multiple **authentication methods**: basic (user credentials), OAuth 2.0/OIDC, and SAML SSO. The simplest method is a login call using Vault credentials:

- **Username/Password Login:** The client sends a POST to the authentication endpoint:

  ```
  POST https://yourvault.veevavault.com/api/{version}/auth
  Content-Type: application/x-www-form-urlencoded

  username=your.username@company.com&password=YourPassword
  ```

  On success, Vault returns a JSON containing a `sessionId` (a long token string) (API Reference) (API Reference). This session ID must be included in subsequent requests. Typically, it's passed in an HTTP header, e.g. `Authorization: {sessionId}` for basic session auth (API Reference). Vault also accepts the more standard `Authorization: Bearer {sessionId}` format (API Reference) – functionally equivalent for Vault API calls.

- **OAuth 2.0 / SSO:** Vault can integrate with enterprise SSO. For OAuth, an access token obtained from an IdP can be used in the `Authorization: Bearer {access_token}` header (API Reference). Vault's `/auth` endpoint also supports an OAuth flow for certain configurations. SAML tokens can likewise be exchanged for a Vault session. These methods allow integrations to avoid handling user passwords directly and leverage central identity management (About Vault API - Veeva Vault Help).

- **Session management:** Once obtained, a Vault session ID remains valid for a configurable duration of inactivity (e.g. 30 minutes) up to a maximum of 48 hours (API Reference) (API Reference). Integrations should reuse the session ID for multiple calls (to avoid re-authenticating frequently) and refresh it when expired. Vault APIs also provide a "keep alive" endpoint to extend sessions programmatically (API Reference).

- **Security & permissions:** All API actions execute with the permissions of the authenticated Vault user. This means record-level security, object permissions, and document role access are enforced. An API user cannot access or modify content beyond what they are allowed in the Vault UI (About Vault API - Veeva Vault Help). All traffic must be over TLS (HTTPS), and Vault's servers validate the session token on each call. Vault also implements **API rate limits** to protect performance – by default a burst of calls above a threshold will be throttled (e.g. requests beyond a certain number per 5-minute window incur a small delay) (About Vault API - Veeva Vault Help). Similarly, too many login attempts in a short time may be temporarily blocked (About Vault API - Veeva Vault Help). These safeguards ensure stability and security for all tenants.

API Versioning and Evolution (Vault)

Veeva Vault's API is versioned to allow continuous improvements while maintaining backward compatibility. Veeva releases Vault updates three times a year (e.g., 22R1, 22R2, 22R3 corresponding to Year.Release). Each major release usually introduces a new API version (e.g., `v22.1`, `v22.2`, etc.). The **latest version** may be labeled "Beta" until the subsequent release, at which point it becomes "GA" (General Availability). Importantly, once an API version is GA, it remains stable – older versions **do not change** even as the platform evolves (API Reference).

This guarantees that integrations built against a given version will continue to work without modification in future Vault releases (API Reference).

For example, if Vault 23R3 introduces API version 23.3 as Beta, Vault 24R1 would promote 23.3 to GA and introduce 24.1 as Beta. Integrators can choose to upgrade to newer API versions to access new features, but aren't forced to until they're ready. Veeva typically supports multiple past versions (often all versions not officially retired). Documentation for each version is available on the Vault Developer Portal, and changes are noted in release notes. This versioning strategy allows Vault's API to evolve (adding new resources, fields, etc.) without breaking existing integrations.

Vault Java SDK (In-Platform SDK)

In addition to web service APIs, Vault provides a **Java SDK** for writing custom code that runs inside the Vault Platform. The Vault Java SDK allows developers to extend Vault's functionality by developing custom **Vault Extensions** in Java (Vault Java SDK Documentation). These extensions can implement business logic such as custom triggers (e.g., execute code when a document is approved), custom user actions (buttons that run logic), data validations, or integrations with external services via callouts.

Key points about the Vault Java SDK:

- Developers write Java code using provided Vault SDK libraries, and package the code as a Vault Package (often a ZIP file with a `vpk` extension). This is deployed to a Vault environment via the admin UI or API.
- The code runs within Vault's cloud environment (sandbox or production Vault), so there are security and execution time limits. The SDK provides an `HTTPService` for safe outbound HTTP calls (e.g., to call an external API) (Veeva vault java sdk external libraries - Stack Overflow), and other services to interact with Vault data.
- Vault Java SDK uses standard Java (currently Java 17 as of recent updates (Vault Java SDK Documentation)), so developers can work in familiar IDEs and debug locally, then deploy to Vault. The custom code is executed by Vault seamlessly as part of the Vault application's logic (Vault Java SDK Documentation).
- This SDK is not an external API, but rather an **internal extension mechanism** – useful for implementing complex business rules or integrations that need to react to Vault events in real-time. For example, one could write a Vault SDK script that triggers whenever a new document is uploaded and automatically posts a summary to an external system's REST API.

Vault Direct Data API

A recent addition to the Vault ecosystem is the **Direct Data API**. This is a special high-performance API designed for **bulk data extraction** from Vault. Traditional Vault REST calls

retrieve records in real-time (with some rate limits), which can be slow if extracting millions of records. Direct Data API provides a way to replicate or export Vault data **up to 100× faster than traditional APIs** (Veeva Adds Direct Data API to Veeva Vault Platform - Contract Pharma). It does so in a transactionally consistent manner, meaning the data extract represents a stable snapshot.

How it works: Instead of making numerous GET calls, a client can use Direct Data API to request an export of an entire object's data (or a subset). Vault then generates the data extract on the server side (respecting all security controls) and typically provides the results in a file or stream. This is often used to feed data warehouses or analytics systems. For example, an organization might use Direct Data API to pull all Vault documents metadata or all object records to an external database for reporting. Indeed, Veeva provides connectors – e.g., a sample **Amazon Redshift connector** – that utilize the Direct Data API to efficiently load Vault data into an S3 data lake and then to Redshift (veeva/Direct-Data-API-connector-for-Amazon-Redshift - GitHub).

Direct Data API is included as part of the Vault Platform license (no extra cost) (Veeva announces Direct Data API included as part ... - Markets Insider). It emphasizes **throughput** over interactivity – ideal for nightly jobs or initial data loads. Security is maintained (only accessible data is exported) and it does not impact the Vault application's performance for end-users (the export is handled asynchronously). This API is particularly relevant for Vault customers needing business intelligence or large-scale data integration.

Veeva CRM APIs

Overview: Veeva CRM is a CRM solution for life sciences field teams, built on the Salesforce platform. Because of this architecture, Veeva CRM supports both **Salesforce's native APIs** and **Veeva-provided specialized APIs**. Integrations can use Salesforce APIs (SOAP, REST, Bulk) to interact with standard and custom objects in the CRM (Accounts, Contacts, Calls, etc.), and they can also use Veeva's REST APIs for features unique to Veeva CRM (for example, the Order Management module, CLM content, etc.). All these APIs allow external systems or custom applications to read/write CRM data and invoke CRM functionality securely.

Salesforce APIs for Veeva CRM

Since Veeva CRM data resides in a Salesforce org, the core **Salesforce APIs** are available by default. This includes:

- **SOAP API** – a WSDL-based web service for creating, retrieving, updating CRM records, executing queries (SOQL), etc. Many integration middleware or ETL tools use the SOAP API to pull data from or push data to Veeva CRM. For example, one could use the Salesforce SOAP API to extract all new Account records from Veeva CRM (since Veeva CRM's Account

object extends the standard Salesforce Account). The SOAP API requires logging in to get a session ID or using OAuth tokens.

- **REST API** – a RESTful HTTP API provided by Salesforce (e.g. endpoints under `/services/data/vXX.X/` in the Salesforce domain). This can accomplish similar tasks as SOAP (CRUD, query, etc.) but in a lighter-weight REST/JSON format.
- **Bulk API** – Salesforce Bulk API v1.0 and v2.0 can be used for high-volume data loading in Veeva CRM. Bulk API v1.0 (which uses REST for data upload but a SOAP login) is commonly used to insert or update large batches of records (thousands or millions) asynchronously. Bulk API v2.0 (all REST) can also be used for inserting, querying, or deleting big data sets. These are useful for initial data loads or batch integrations (for example, loading a million prescription records into a custom object in Veeva CRM).
- **Streaming and Pub/Sub APIs** – Salesforce provides streaming API (PushTopic or CDC – Change Data Capture events) which could be used to get real-time notifications of changes in Veeva CRM data. This can be useful if an external system needs to know immediately when, say, an Account is updated or a Call is completed in CRM. (While not a data API per se, it's part of the integration toolset.)

Authentication: To use Salesforce APIs against Veeva CRM, one would authenticate via Salesforce (using OAuth 2.0 web flow or SOAP login with username, password, and token) and then call the desired APIs. The credentials and permissions of the Salesforce/Veeva CRM user account govern what data can be accessed. Veeva CRM's profiles and permission sets align with Salesforce's security model, so standard API security applies.

Note: Veeva CRM being a managed package on Salesforce means any Veeva-specific objects have API names usually ending in `__c` (for custom objects) or `__vod__c` for some managed package components (Vod is a common namespace in Veeva CRM). All these are accessible via the Salesforce APIs. For example, the Call Reporting object (Call2_vod__c in Veeva) can be queried or updated through the Salesforce REST/SOAP API just like any standard object.

Veeva CRM REST APIs (Veeva-specific)

In addition to the generic Salesforce APIs, Veeva exposes **custom REST endpoints** to handle certain CRM functionalities that involve complex business logic or offline scenarios. One prime example is the **Order Management API** in Veeva CRM, which is a REST API for working with Orders (often used in the context of pharma sample orders or sales orders). This API is part of Veeva CRM's integration endpoints and allows external applications to create and manage Orders with all of Veeva's business rules applied.

- **Order Management REST API:** This API provides access to Veeva CRM's Order objects and related functionality via REST calls. It allows external systems to **create, edit, submit, void, unlock, and delete** orders in CRM, as well as retrieve order details (Order Management Rest API). Critically, it also exposes Veeva's **Pricing Engine** – when an order is created or modified through the API, the pricing rules (customer-specific pricing, discounts, auto-added line items, etc.) are applied automatically, just as they would be in the CRM UI (Order Management Rest API). This ensures the order totals and lines are validated and priced correctly. The Order API methods also honor any custom Apex triggers or logic defined in the org for orders (Veeva CRM APIs for Order Management).

  Example: The base URL for Order Management API calls includes the instance's content server and a versioned API path. For example:

  ```
  https://na1.vod309.com/172.13.10/api/v1/om/
  ```

  is a sample base (where `na1.vod309.com` is the Veeva content server, `172.13.10` indicates the specific CRM release version, and `api/v1/om/` indicates version 1 of the Order Management ("om") API) (Order Management Rest API). An external system could POST an order JSON to `.../orders` endpoint to create an order, or GET `.../orders/{id}` to fetch an order. The responses and requests are in JSON format (Veeva CRM APIs for Order Management). For instance, retrieving an order might return a JSON structure with order header fields and line items. Creating an order involves sending a JSON payload; the API will respond with status and any errors (such as pricing rule violations) in the response.

- **Other CRM APIs:** Veeva's documentation suggests that similar REST APIs may exist for other CRM functions (for example, retrieving CLM content presentations, or working with Meetings/Events, etc.), though the primary one publicly documented is Order Management. These APIs follow the same design principles: they are RESTful, JSON-based, and apply Veeva CRM's business logic on top of Salesforce data (Veeva CRM APIs for Order Management). For example, a **CLM content API** (if available) might allow listing or updating approved content presentations in the CRM media library. Likewise, Veeva **MyInsights** (a custom HTML content framework in CRM) provides a JavaScript API for in-app data access (discussed further below).

- **Guiding principles and compatibility:** The Veeva CRM REST APIs are designed to be intuitive (following REST conventions), secure (always TLS, enforcing user permissions), efficient, and aligned with the underlying CRM functionality (Veeva CRM APIs for Order Management). They leverage the same logic as the CRM UI and Salesforce platform, meaning any validations or Apex logic will execute as part of API calls (Veeva CRM APIs for Order Management). Veeva CRM's REST API is versioned similarly to Vault and Network – new versions may be introduced with CRM releases, but older API versions remain supported so that existing integrations continue working (Veeva CRM APIs for Order Management). In practice, Veeva CRM maintains support for API versions until they are no longer used in production (Veeva CRM APIs for Order Management). This ensures backward compatibility (though new features will only be in the latest version).

Authentication & Authorization (Veeva CRM API)

Accessing Veeva CRM's own REST APIs requires a Veeva CRM user session. Because Veeva CRM is on Salesforce, there are two ways to authenticate: reuse an existing Salesforce session

or perform a login via Veeva's token endpoint.

- **Using Salesforce Session (OAuth JWT):** If an integration already has a Salesforce session ID (for example, obtained via Single Sign-On or OAuth), it can be passed to Veeva CRM's API. Veeva CRM provides a mechanism to accept a Salesforce session and endpoint in the header of API calls (Order Management Rest API). Specifically, the client would include: `sfSession: <Salesforce session id>` and `sfEndpoint: <Salesforce SOAP endpoint URL>` as HTTP headers in the request (Order Management Rest API), along with an `Authorization` header containing the username (and a dummy password if using Basic auth) (Order Management Rest API) (Order Management Rest API). Veeva's server will then use that Salesforce session to authorize the API call. This approach is useful when an integration is running within the Salesforce context or has already authenticated. It avoids storing user passwords and leverages the live session.

- **Direct Login via Veeva API:** If an integration cannot use an existing session, Veeva CRM offers an auth endpoint similar to Vault's. The client can POST the Salesforce username and password to a Veeva authentication URL (for example, `https://{veeva_instance}/auth`) with an API version parameter (Order Management Rest API). If valid, this returns a session token and content server URL (Order Management Rest API) which can be used for subsequent calls. After this initial auth, further API calls can just include the session token and username (with a dummy password) in the Authorization header (Order Management Rest API). Veeva documentation recommends using a dummy password in subsequent calls to prevent exposure of the real password, since the session ID is what actually grants access (Order Management Rest API). This method essentially logs into Salesforce behind the scenes and returns a session for Veeva API use. (Note: This approach might be considered legacy as most modern integrations use OAuth or existing sessions, but it is available for completeness.)

In all cases, the user's profiles and permissions in CRM determine data access. An "integration user" is often created with specific permissions to use the APIs (for example, able to read/write orders, but maybe read-only on some sensitive objects depending on the use case). Calls are made over HTTPS and require the appropriate domain (which can be obtained from Veeva CRM's configuration as shown in the base URL example).

Example – Creating an Order via API

To illustrate, consider an external order entry system that needs to push an order into Veeva CRM. Using the Order Management REST API: the system would first authenticate (either via an existing session or the login call) to get a session token. Then it would send a `POST` request to the Orders resource. For example:

```
POST /api/v1/om/orders  (HTTP/1.1)
Host: na1.vod309.com
Authorization: Basic <username>:<dummyPassword>
sfSession: 00D...!ABC...   (Salesforce session token)
sfEndpoint: https://mydomain.my.salesforce.com/...
Content-Type: application/json

{
```

```
    "AccountId": "001...ABC",
    "Date": "2025-04-10",
    "LineItems": [
        {"ProductId": "a01...XYZ", "Quantity": 5},
        {"ProductId": "a01...QQQ", "Quantity": 3}
    ]
}
```

The above JSON is a simplified representation: in reality the field names would match the API's expected payload (which might use Veeva field API names). The response would include a status (success or failure) and the created Order's ID and details, or validation errors if any. Veeva's pricing engine would have been run – so the response might also include calculated fields like total price, discounts applied, or even additional line items automatically added (e.g., free goods) as per configuration (Order Management Rest API) (Order Management Rest API). If the pricing engine found issues (like an invalid product for that account), the response would convey those errors. This process allows an external system to leverage all of Veeva CRM's complex order logic without directly accessing the Salesforce database – everything is handled through the controlled API.

Veeva CRM JavaScript SDK (MyInsights)

While not a server-to-server API, it's worth noting that Veeva CRM offers a client-side SDK called **MyInsights** for building custom interactive content within the CRM app. MyInsights allows developers to create HTML/JavaScript pages that run inside the Veeva CRM iPad or web app. These pages can call a **JavaScript API (Veeva CRM JS Library)** to retrieve CRM data or perform actions in context. For example, a MyInsights page might query for products and sales figures for a given account and display a custom chart. The MyInsights Data APIs simplify querying Salesforce data from within the page (MyInsights v2.0 - Veeva CRM). This is an **SDK for extending CRM's UI**, distinct from the REST/SOAP APIs, but it uses API calls under the hood to fetch data. Authentication is handled by the fact it runs inside an already logged-in session (no separate login needed). Technical audiences might use MyInsights to create rich dashboards or integrate third-party data within CRM, complementing the server-side integrations described above.

Summary (CRM): In practice, integrators working with Veeva CRM have a rich toolkit: they might use **Salesforce SOAP/REST** APIs for standard data operations and the **Veeva-specific REST endpoints** for specialized operations like Order Management. All of these are secured via Salesforce's robust authentication and honor the configured business rules in Veeva CRM. Veeva's own APIs are versioned and backward compatible (Veeva CRM APIs for Order Management), ensuring that custom applications built against one version (say, an Order API v1.0) will continue to work in future CRM releases. This layered approach (Salesforce + Veeva API) gives flexibility depending on the integration requirements.

Veeva Network API

Overview: Veeva Network is a cloud-based master data management (MDM) application for customer data (HCPs – healthcare professionals, HCOs – healthcare organizations, and related entities). The **Veeva Network API** provides programmatic access to this master data and Network's data management features (About the Network API). It is primarily a RESTful API returning JSON, designed to be simple, powerful, and secure (About the Network API). Common use cases include syncing HCP data from Network into other systems (like CRM or data warehouses), matching external records against the master database, and submitting updates or additions (data change requests) to the master data.

Network REST API Capabilities

Using the Network API, developers can:

- **Retrieve data:** Query and fetch records of HCPs, HCOs, and related sub-objects (like addresses, licenses, affiliations). For example, you can retrieve a profile of a healthcare professional by their Veeva ID or search for professionals meeting certain criteria. There are endpoints for retrieving objects and performing searches.
- **Match records:** An important function of MDM is matching incoming data to existing records. The API provides a *match* operation where you send a record (with identifying info like name, address, etc.), and the Network API returns either a match to an existing entity or a determination that it's new (About the Network API). This is useful when, say, a new doctor is entered in CRM – an integration can call Network's match API to see if that doctor already exists in the master database (to avoid duplicates). The API can match using Veeva IDs or custom keys as well (API Reference - Veeva Network API).
- **Create and update records (with approval workflows):** Instead of directly inserting data, Network uses a **Data Change Request (DCR)** mechanism. Via the API, you can submit a request to add or update an HCP/HCO record (About the Network API). Network will apply business rules and, if configured, route the request to data stewards for approval. Once approved (or if auto-approved), the change becomes part of the master data. For example, if a sales rep discovers a new clinic, an integration could submit a DCR via API to add that clinic. The response will include a DCR ID and status. There are also API calls to check DCR status or results.
- **Merge or delete records:** If applicable, the API can also initiate merges (combining duplicate records) or deactivations. These usually also go through DCRs or specific admin-controlled endpoints.
- **Metadata access:** The Network API provides metadata endpoints (like `/metadata/objectTypes` and `/metadata/fields`) to discover the data model in a given Network instance. This is useful to see what objects and fields exist (including custom fields) and their descriptions. For instance, an API call can list that the instance has objects like HCP, HCO, and custom objects, with each field's API name and type (About the Network API) (About the Network API).

Network's API was built with similar principles to Vault's: it is intuitive, always SSL-secured, and reflects the same business logic as the Network UI (About the Network API). When you use the API to create or change data, all of Network's validation (for example, ensuring required fields are present, or that a license number is valid) will execute, so you cannot bypass data quality rules by using the API.

Historically, Veeva Network offered a SOAP-based "Web Services API" as well (General Service Description) (for example, a WSDL that allowed queries and DCR submissions). In recent years, the REST API has become the primary interface as it is more flexible and easier to adopt. The old web service functions (search, view, DCR) are essentially all achievable via the REST API now, with better performance and JSON formatting.

Authentication & Security (Network)

Authenticating to the Network API is very similar to Vault: the client must first obtain a **session ID** via an authentication call. The Network API's auth endpoint is `/api/{version}/auth`, which expects a POST with form-encoded credentials (username and password) (About the Network API). For example:

```
curl -X POST "https://my.veevanetwork.com/api/v31.0/auth" \
     -H "Content-Type: application/x-www-form-urlencoded" \
     -d "username=api.user@mycompany.com&password=SecretPassword"
```

If the credentials are valid, the response will include a JSON with `responseStatus: SUCCESS` and a `sessionId` (a long token string) (About the Network API). It also lists the Network instance(s) the user has access to (many customers have only one network instance, but if multiple, it's enumerated) (About the Network API). This session ID must be provided in subsequent API calls. Network requires it in the HTTP header `Authorization` (just the token string) for all requests (About the Network API). For example:

```
Authorization: 8E974F39652C0AC7D2458107D433A78583511D365F0BF13387D45569F37CBD468...
Accept: application/json
```

Once authenticated, an integration can call any Network API endpoint allowed by that user's role. Network uses a robust security model: user accounts can be configured as data stewards, read-only users, etc., and the API will enforce those permissions. For instance, if an API user account is set to read-only, any attempt to create or update via API will be denied. All API traffic is over HTTPS and requires the correct domain for the specific Network instance (commonly `my.veevanetwork.com` or similar). Session tokens are typically valid for a certain period of time (similar to Vault, e.g., 30 minutes of inactivity or a few hours max). Integrations should handle token expiration by re-authenticating as needed.

Example – Retrieving a Record and Submitting a Change

Data retrieval: Suppose you want to get details of a specific HCP by ID. After obtaining a session, you could call:

```
GET /api/v35.0/objects/HCP/{hcp_id}
Authorization: {sessionId}
Accept: application/json
```

The response would be a JSON object containing all the fields of that HCP (name, address, specialty, status, etc.), provided the requesting user has access. You could also perform a search by name: e.g., `GET /api/v35.0/search?objectType=HCP&searchTerm=Dr%20Alice%20Smith` to find matches.

Match example: If you have a new record (say from a third-party list) and you want to see if it exists in Network, you might use the **match API**. The request could be a POST to `/api/v35.0/match` with a JSON payload of the entity data (like name, address to match on). The response will include a match score and either identify an existing entity or indicate no match (and possibly provide an ID for a new pending record) (Veeva Network API). This helps in integration scenarios to prevent duplicate creation.

Submitting a DCR: To update a record, you don't directly PUT to an HCP. Instead, you POST a change request. For example:

```
POST /api/v35.0/change_requests
Authorization: {sessionId}
Content-Type: application/json

{
  "entity_type": "HCP",
  "entity_vid__v": "12345",          ← ID of the HCP to update (or omit for new)
  "fields": {
    "last_name__v": "Smith",
    "primary_specialty__v": "Cardiology"
  },
  "metadata": {
    "source": "CRM System",
    "note": "Auto-update specialty based on recent data"
  }
}
```

The above JSON (illustrative) requests to change Dr. Smith's primary specialty to Cardiology. The API would respond with a `change_request_id` and status. This request would then appear in Network for approval if required. If auto-approved, the change might be applied immediately and the status would eventually be updated to SUCCESS. The Network API allows checking the status of a change request by ID (`GET /change_requests/{id}`) and even listing pending change requests. This mechanism ensures data quality by involving data stewards when needed, even for API-driven changes (API Reference) (API Reference).

Bulk Data Export and Integration

Veeva Network is often used as a central hub of customer data that needs to flow into other systems (like CRM, data lakes, etc.). The Network API supports **bulk export jobs** via a concept of **Subscriptions**. In Network, you can define target subscriptions that, for example, export all HCPs matching certain criteria to a file (CSV) on an FTP. Through the API, you can initiate these export jobs and retrieve the results programmatically (About the Network API). The sequence might be: call an endpoint to run a subscription job, poll for job completion, then download the exported file (which could be obtained via API or from a secure FTP location configured in Network). This allows full automation of data extract processes. There are corresponding API calls for **third-party source subscriptions** (bringing data *into* Network in bulk). For instance, if a company has a data feed of physicians from a vendor, an integration can use the API to send that file into Network (often via an FTP drop and then triggering a load job via API) (About the Network API).

Network's design acknowledges that some integrations will use file-based transfer for large volumes (hence the FTP/subscription approach), whereas others will use real-time APIs for incremental updates. The API covers both: real-time CRUD/Match, and asynchronous bulk jobs.

API Versioning and Updates (Network)

Similar to Vault, Veeva Network versioning aims for backward compatibility. Each release of Veeva Network (which occur a few times a year) comes with a new API version if there were enhancements. For example, "Network version 22.0 included the GA (general availability) of the version 21.0 API" (About the Network API). This implies the API version lags slightly behind the product versioning. Network maintains support for each API version across future releases, meaning an integration built on v21.0 of the API will continue to function on Network 23.0, 24.0, etc. (About the Network API). New API versions are introduced to expose new capabilities, but those new features won't be available in older version endpoints (About the Network API). Eventually, when an API version is considered obsolete and no clients use it, Veeva may deprecate it, but this is managed over long timeframes to give customers time to migrate. In practice, you'll specify the version in the URL (e.g. `/api/v30.0/…`) and can keep using that until you choose to update your integration to a newer version.

Veeva OpenData API

Overview: Veeva OpenData is a global reference data service providing up-to-date information on HCPs, HCOs, and affiliations across regions. While Veeva Network is typically a customer's private master database (often seeded or updated by OpenData), the **Veeva OpenData API** allows direct interaction with the OpenData service. It is also a REST API (JSON), focusing on retrieving reference data and submitting change requests to the OpenData stewards. Essentially,

if your organization subscribes to Veeva OpenData, you can use this API to query that dataset and propose updates.

Key capabilities of the OpenData API include:

- **Retrieve entities:** You can retrieve HCP or HCO records from the OpenData database by ID. The data returned includes all standardized fields (such as name, specialty, license, address, status, etc.). This is useful to get the latest info on a doctor or institution without storing it locally. You can also search or query entities with certain criteria (the API supports queries with filters, e.g., all cardiologists in a certain city).
- **Create change requests:** Unlike a typical CRUD, since OpenData is a managed dataset, you generally cannot directly insert or edit data. Instead, you **submit change requests** (similar to Network's DCR concept) to suggest an addition or correction to the data. For example, if a sales rep finds a doctor's address is wrong, an integration (or CRM itself) can send a change request via the OpenData API to update the address (API Reference). Veeva's data steward team will review and either accept or reject the change. If accepted, the change will reflect in OpenData and propagate to subscribers. The API returns a unique change request ID for tracking (API Reference) (API Reference).
- **Data model metadata:** The OpenData API also provides metadata endpoints to understand the structure (available fields, field groups, etc.) of the OpenData entities. This is very similar to Network's metadata and allows dynamic integrations that adapt to new fields.

Authentication: The OpenData API uses a similar authentication mechanism (obtaining a session token and using it in the `Authorization` header) (API Reference). Typically, clients will use a dedicated OpenData API user (provided as part of the OpenData subscription) to log in. For example:

```
curl -X POST "https://opendataexplorer.veeva.com/api/opendata/v1.0/auth" \
     -d "username=odata_user&password=****"
```

This returns a JSON with a `sessionId` (API Reference), which must be used in subsequent calls. All calls go to the base path `https://opendataexplorer.veeva.com/api/opendata/v1.0/...`. The session works similarly to Vault/Network: include `Authorization: {sessionId}` in headers (API Reference). Sessions have a validity period after which you'd re-authenticate.

Example – Adding an HCP via OpenData Change Request:
Suppose your organization wants to add a new physician to OpenData (one that isn't currently in the dataset). Using the API, you would submit a change request like so:

```
curl -X POST -H "Authorization: {sessionId}" -H "Content-Type: application/json" \
     -d @new_hcp.json \
     "https://opendataexplorer.veeva.com/api/opendata/v1.0/change_request"
```

Where `new_hcp.json` contains something like:

```
{
  "entity_type": "HCP",
  "fields": {
    "first_name__v": "John",
    "last_name__v": "Doe",
    "primary_address__v": {
       "street__v": "123 Main St", "city__v": "Boston", "country__v": "US", ...
    },
    "specialty__v": "Oncology",
    "...": "..."
  },
  "metadata": {
    "creator": "MyCRMSystem",
    "note": "New physician discovered in territory",
    "source": "Field Team"
  }
}
```

The API would respond with a success status and a `change_request_id` acknowledging the submission (API Reference). In the background, this request goes to the appropriate OpenData regional data steward team for validation. The API client can later retrieve the status of this request by calling:

```
GET /api/opendata/v1.0/change_requests/{change_request_id}
```

If approved, the new HCP will be added to OpenData and given a permanent Veeva ID. The next OpenData data refresh to your Veeva Network or CRM will include this new record. If rejected, the status might indicate the reason (e.g., already exists or insufficient information).

Use cases: The OpenData API is often used in tandem with Veeva Network or Veeva CRM. For instance, a CRM system can be configured such that if a user attempts to add a new HCP that isn't in Network, the system could call the OpenData API to fetch potential matches from the global dataset. If a match is found, the user can import that data; if not, the system might trigger an OpenData change request to have the person added. Similarly, if certain fields are missing (like a license number), an automated process could query OpenData via API to fill in the data.

In summary, the OpenData API extends the power of Veeva Network by tapping into the global repository, with appropriate governance through change requests. It ensures that any additions or edits go through Veeva's stewardship process, maintaining the quality of the master data.

Integration Scenarios and Examples

The following are integration scenarios that demonstrate how the above APIs work together in real-world use cases:

- **Vault to CRM (Content Integration):** Pharmaceutical companies often use **Vault PromoMats** (a Vault for promotional materials) to manage content and **Veeva CRM** to deliver content to sales reps on iPads (CLM – Closed Loop Marketing). An integration can be built where approved content in Vault is published to CRM. Using the Vault REST API, a script can **export approved documents** (e.g., detail aids, slide decks) and related metadata from Vault, then, using the Salesforce API, **upload those documents into Veeva CRM's Media library** (as attachments or content records). This ensures that when reps open their CRM app, they have the latest approved materials. Vault's API might provide a document download URL (About Vault API - Veeva Vault Help), and CRM's API (or even a specialized Vault-to-CRM connector) can push the file to the appropriate Salesforce object (e.g., a CRM "Presentation" record). Authentication would involve a Vault session for the export and a Salesforce session for the import. Veeva provides out-of-the-box integration for PromoMats-to-CRM, but it's essentially leveraging these APIs under the hood.

- **CRM to Network (Customer Master Sync):** In the field, reps may collect new HCP information in CRM (for example, adding a new doctor during a call). Rather than keeping that data only in CRM, companies use Veeva Network to maintain a **clean master list of HCPs**. An integration can take new or updated Accounts from Veeva CRM and push them to Network via the Network API. For instance, a nightly job could query CRM (using Salesforce Bulk API) for any Accounts created/modified that day marked as "professionals", then for each, call the Network API to either match or create a change request. If a match is found, Network returns the existing profile (and the integration might update the CRM record with the Network ID). If no match, the integration submits a new HCP DCR to Network (About the Network API). Conversely, data steward-approved changes in Network can be **pushed back to CRM**: e.g., if an address is updated in Network, an integration can use the CRM API to find the corresponding Account and update the address. Veeva facilitates some of this with a managed package, but custom middleware can be used as well. This two-way sync ensures CRM users have the latest data and any new insights they gather get centralized.

- **Network & OpenData – Enrich CRM:** For customers subscribing to Veeva OpenData without a full Network deployment, CRM can directly leverage OpenData API. For example, a CRM system can call the OpenData API to **search for an HCP** when a user enters a new contact. The API might return a shortlist of possible matches from the OpenData master (with rich info like specialties, affiliations). The user can select one, and the CRM can then store the OpenData ID, or even initiate an OpenData download to get the full details. If the HCP isn't found, the CRM could automatically create an OpenData change request via API to add this HCP (API Reference). This way, the rep's new contact eventually becomes part of the global reference dataset. Such an integration requires coordinating three APIs: CRM's (to get user input and save data), OpenData's (to search/add data), and possibly Network's (if Network is receiving the OpenData feed). Authentication would involve a service account for OpenData API and the current user's session for CRM.

- **Analytics and Data Warehouse Integration:** Many companies extract data from Veeva systems into a central data lake or warehouse for analytics. Using the **Vault Direct Data API**, a team can nightly pull all newly approved documents and their properties from Vault to an AWS S3 bucket, then load into a Redshift warehouse for reporting on content usage. Similarly, using the **Network API bulk export**, an integration can schedule a job to export the entire list of active HCPs to a CSV, then combine that with sales data from CRM. Veeva **Nitro** is a product that helps with this by providing an out-of-the-box data warehouse and tools, but even without Nitro, these APIs allow building a custom pipeline. For example, a Python script could use the Vault API to get data in chunks (or via Direct Data API for efficiency) and then push to a database. All such data extracts must respect API limits and use appropriate filters (or the designed bulk mechanisms) to avoid hitting rate limits.

- **Mobile Offline Integration:** Veeva CRM's mobile app allows offline use and sync, which is mostly internal to the app. However, some customers extend CRM with mobile workflows. For instance, an external mobile app for sample inventory could call Veeva CRM's APIs to log sample usage. Using the Veeva CRM REST API, the app can authenticate via an integration user and create a Samples transaction record in CRM, which will later sync to the rep's device. In this scenario, understanding the **Veeva API's business logic** is crucial – e.g., the Samples object might require certain fields. The app benefits from the API ensuring that, say, sample allocations are decremented and compliance rules are applied as if the rep entered it in CRM directly.

These scenarios highlight that the **APIs across Veeva Vault, CRM, and Network/OpenData are complementary**. Vault's APIs excel at content and regulated documents integration, CRM's at customer interactions and sales data, and Network/OpenData at master data quality. By combining them, organizations achieve end-to-end integration: from managed content, to field execution, to master data management and analytics.

Conclusion

The Veeva ecosystem offers a rich set of APIs enabling integration and extension of its products to meet complex business needs. **REST APIs** are the norm across Vault, CRM, and Network, providing intuitive JSON-based access to data and functions (with **Bulk** endpoints available for efficiency). **SOAP APIs** historically existed (Salesforce SOAP for CRM, legacy web services for Network) and are still usable for certain integration patterns, but REST is now dominant for new projects. Each API employs robust **authentication and security** controls – from session tokens to OAuth – ensuring that data access is authorized and encrypted. Veeva's commitment to **backward compatibility and versioning** means integrators can rely on stable interfaces even as the platforms evolve (API Reference) (Veeva CRM APIs for Order Management).

For developers, the availability of **SDKs** like the Vault Java SDK and the CRM MyInsights library provides additional flexibility to customize functionality within the platforms themselves, going beyond what remote APIs can do.

In summary, whether you need to integrate a document repository with external systems, synchronize customer data between Veeva and a CRM/ERP, or build custom applications on top

of Veeva's cloud, the ecosystem's APIs provide the necessary tools. By leveraging these APIs – with proper authentication, using the appropriate endpoints for the use case (REST for real-time, bulk or Direct Data API for volume, etc.), and understanding the data model – technical teams can create seamless integrations and innovative solutions that maximize the value of Veeva's products in the enterprise landscape. All development should refer to Veeva's official documentation and developer portals for the latest details and examples (About Vault API - Veeva Vault Help) (Veeva CRM APIs for Order Management), ensuring alignment with Veeva's best practices and any product-specific considerations.

DISCLAIMER

The information contained in this document is provided for educational and informational purposes only. We make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability, or availability of the information contained herein.

Any reliance you place on such information is strictly at your own risk. In no event will IntuitionLabs.ai or its representatives be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from the use of information presented in this document.

This document may contain content generated with the assistance of artificial intelligence technologies. Despite our quality control measures, AI-generated content may contain errors, omissions, or inaccuracies. Readers are advised to independently verify any critical information before acting upon it.

All product names, logos, brands, trademarks, and registered trademarks mentioned in this document are the property of their respective owners. All company, product, and service names used in this document are for identification purposes only. Use of these names, logos, trademarks, and brands does not imply endorsement by the respective trademark holders.

IntuitionLabs.ai is an innovative AI consulting firm specializing in software, CRM, and Veeva solutions for the pharmaceutical industry. Founded in 2023 by Adrien Laurent and based in San Jose, California, we leverage artificial intelligence to enhance business processes and strategic decision-making for our clients.

This document does not constitute professional or legal advice. For specific guidance related to your business needs, please consult with appropriate qualified professionals.