Best Practices: Integrating Veeva Vault with Other Systems

Igor Tsives, Sr. Product Manager
Melissa Aron, R&D Practice Manager
Safe Harbor

The following is intended to outline Veeva’s general product direction. It is intended for informational purposes only and does not form part of any contract. It is not a commitment to deliver any specific functionality and it should not be relied upon in making purchasing decisions. Features and functions described in this document will be released commercially only when and if they become available. The development, release, and timing of the features or functionality of Veeva’s products remains at the sole discretion of Veeva.
Today’s Session

- Learn what our APIs can do from a functional and business perspective
- Get a technical overview of how to use our APIs
- Discuss integration planning and strategies for successful and powerful integrations
Why Integrate?

Integration Benefits

- Extend application features
- Integrate business processes and data
- Simplify user interactions
- Enforce data integrity
Integration Strategies

- Backend-to-Backend
- Application-to-Application
- Connector-Based
Vault API
API Features

- search
- study
- roles
- users
- binders
- relationships
- country
- documents
- product
- query
- site
- workflows
- custom objects
- picklists
- renditions
- versions
- groups
## Pragmatic Design Principles

<table>
<thead>
<tr>
<th>Learnable</th>
<th>Secure</th>
<th>Efficient</th>
<th>Aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Intuitive and easy to learn</td>
<td>▪ Access is secured by SSL</td>
<td>▪ Efficient request processing</td>
<td>▪ Leverages the Vault Platform</td>
</tr>
<tr>
<td></td>
<td>▪ It incorporates elements of REST principles, making access to actions consistent and intuitive</td>
<td>▪ In compliance with the access permissions of the authenticated user</td>
<td>▪ The new API versions consistently reflect new capabilities introduced in new versions of Vault</td>
</tr>
</tbody>
</table>
### Authentication / Authorization

- **Vault session ID which is required on each API request**
  - Request: Authorization Header

1. **Standard Authentication**
   - Utilizes Vault username and password

2. **Security Assertion Markup Language (SAML)**
   - Exchanges SAML Response for a Vault Session ID

3. **Salesforce Delegated Authentication**
   - Consumes Salesforce.com session
## Request / Response

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ RESTful Endpoints</td>
<td>▪ 200 OK</td>
</tr>
<tr>
<td>▪ Secured over HTTPS</td>
<td>▪ Either valid data or an error</td>
</tr>
<tr>
<td>▪ Session ID passed in Authorization Header</td>
<td>▪ Error contains error type and message</td>
</tr>
<tr>
<td>▪ Response format requested through Accept Header</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ XML: application/xml</td>
</tr>
<tr>
<td></td>
<td>▪ JSON: application/json</td>
</tr>
</tbody>
</table>
Metadata

- Each Vault could have different configuration: document types, document fields, custom objects, etc.

- The Metadata APIs allow you to interrogate the Vault to understand what metadata is available to use on a given object.

- The Metadata APIs allow you to Build dynamic integrations which adapt to configuration changes.
Vault Query Language

- Vault Query Language (VQL) – an SQL-like language
- Simple but powerful queries to retrieve Vault data
- Joins: Document-to-Object, Object-to-Object

**Query String Syntax:**
- **Select fields to retrieve (id)**
- **Specify objects to query (documents)**
- **Conditions to qualify the objects being queried (filter for name “Cholecap”)**

**VQL Statement (Query String):**

```
https://mycompany.veevavault.com/api/v10.0/query?q=SELECT id FROM documents WHERE name__v = 'Cholecap'
```
Keyword Searching

- Keyword searching is enabled by VQL FIND operator
- Able to search through content and/or metadata
- Able to use search through current, latest or all versions of documents
- Able to utilize Boolean operators for fine-tuned searches

```sql
SELECT id FROM documents FIND ('indication AND oncology' scope all)
```
Versioning and Compatibility

- Each new Vault release consists of two components:
  - A new release of the Vault Platform and Applications
  - A new version of the Vault API

- Version support is maintained for each API version across releases of Vault.

- The API is backward compatible in that an application created to work with a given API version will continue to work with that same API version in future Vault releases.

- Vault supports older versions of the API until they are no longer in use by production customers.
Integration Examples and Approach
Single Sign-on and User Provisioning

**Single Sign-on (SAML v2.0)** (Enabled via Configuration)

- User remember only one set of credentials (corporate)
- Identity Provider (IdP) is used to verify a user’s identity
- Assumes that the user exists in Vault
- Supports SP-Initiated and IdP Initiated flows

**User Provisioning** (Custom Integration)

- Reduces burden on admins to keep users up to date
- Users are sourced from a corporate LDAP or other user directory and pushed into Vault using the Vault API

Very Different Beasts!
User Synchronization

- Periodic extract of users from an external user directory

- A unique ID from the source (employee ID, email) used as federated ID in Vault

Features

- Create/disable users according to group membership or properties in source
- Set user type by group membership
- Set Vault group membership by source group membership
- Synchronize user properties
- Handle name updates
Planning for User Synchronizing

- What will be the federated ID of each user?
- What user type will new, incoming users be?
- Do all users get added into Vault? Only some?
- Into which Vault will they go?
  - For multi-Vault implementations
- Field level data mappings between source system and Vault
- Into which security groups will users be added?
  - How does the integration know?
Example LDAP Synchronization Architecture

- **Active Directory Federation Services**
- **Active Directory**
- **Windows Server**
- **LDAP**
- **Synchronization Process**
  - LDAP API Client
  - Vault API Client
- **Java**
- **Windows or Linux Server**
- **SSO Requests/Responses**
- **Vault**
- **REST API**

- Pull LDAP User Information
- Create, Update, Disable Vault Users
Master Data Integrations

- Vault objects allow you to model the data that provides business context for your applications
  - E.g., Product, Study, Site, Department
  - Object to object relationships provide additional context

- Vault Objects and picklist values frequently sourced from external systems
  - Master Data Hub
  - Data warehouse
  - CTMS systems

- Typically a one way push of data from the source into Vault
Planning the Master Data Integration

- Mapping from source to VOF objects and picklists
- Define transformations where source values do not match Vault values
- External ID is used to link to the primary key of source data
- Can end up with very complex behavior where objects have relationships (e.g. Product, Study, Study-Country and Site in eTMF)
  - Make sure you fully understand the relationships (Parent-Child vs. reference relationships) and implement accordingly
Implementation Approaches

- **Almost all integrations are**
  - Run as scheduled tasks
  - On a customer server inside any firewall
  - Access to local applications and the internet (Vault)

- **Less ideal approaches**
  - Hosted externally
    - If the source system is inside the firewall, can lead to access problems unless carefully managed
  - Hosted as a job in another system (e.g. Documentum)
    - Only advantages are single location for admin and password-less admin session.
    - Loading of libraries is very proprietary
    - Log files generally accessible by administrators – very slow debugging
Custom Actions

- **Custom mash-up style integrations**
  - Generate formatted printable views of document fields
  - 3\textsuperscript{rd} party reporting / analytics / data visualization

- **Hook for custom UI**
  - Invoke a custom page displayed within Vault UI from action menu
Lessons Learned — Plan for Success!

- **Unusual environments cause problems**
  - Documentum Java Method Server
  - SAP Business Integration Suite

- **Communication and internal IT take time**
  - Firewall
  - Email server
  - Local application access

- **Development and test approach are key**
  - Access to development/test instances of local application
  - Access to log files by developers, without intermediary
Lessons Learned — Plan for Success!

- Account for UTF character encoding
  - Specific test cases to include foreign characters, commas, apostrophes, etc.

- Realistic test data, even in development

- Make this a close match to production in terms of content and volume
  - Dirty or unexpected data are an overwhelming cause of errors and delays
Complete the session survey with the Veeva app

Thank you

Search “Veeva” in app store
Password: rdsummit
Thank You

Igor Tsives
igor.Tsives@veeva.com

Melissa Aron
Melissa.Aron@veeva.com