# Credential Vault Requirements

# Table of Contents

- [Overview](#overview)

- [About this Document](#about)

- [Environments](#env)

- [Core Concepts](#coreconcepts)

- [Access Restrictions](#access)

- [Non-Functional Requirements](#nfr)

- [Performance](#performance)

- [High Availability](#availability)

- [Support](#support)

- [Contract API Backwards Compatibility](#compatible)

- [Monitoring Support](#monitoring)

- [Heartbeat](#heartbeat)

- [Service Identity](#identity)

- [Metrics](#metrics)

- [Health](#health)

- [Artifacts](#artifacts)

- [Server API](#serverapi)

- [Server UI](#serverui)

- [Adoption Documentation](#documentation)

# <a name="overview">Overview</a>

BNSF requires a credential vault that supports storing protected information meant

for applications to consume for connectivity, access, encryption, or decryption.

Users (actors) of this service include: systems, administrators, auditors,

Jenkins, and application engineers. The primary goal is for applications to obtain

protected configuration values at service start-up.

For the vault to be usable:

- Modification of the values must be tightly controlled

- Support auditing

- Have an Application Programming Interfact (API) for system integration

The value options must include:

- String values

- Private keys (TLS client certificates and crypto private keys)

# <a name="about">About this Document</a>

This document is meant to describe the required credential vault from the Platform

team's perspective. These are the identified requirements as a collaborative

effort towards modernization. This document is an effort to direct vault

owners and engineers to an "adoption ready" state. It in no way guarantees

correctness or completeness of the vault service. It is also not a commitment

for Transporation System Modernization (TSM) adoption.

# <a name="env">Environments</a>

Information stored within the vault should be considered environment specific.

The vault may either:

- Have a deployment per environment identified (recommended)

- Works with PaaS provisioning solutions

- Offers N environments within the system and scale per environment

- Have an environment key built into the Rest API and client contract

- This implies that both publishing and requesting to a value must be keyed on

application and environment

# <a name="coreconcepts">Core Concepts</a>

- Protected resources are stored encrypted

- If the database is compromised/evaluated, privileged resources are not exposed in the clear.

- The vault cannot decrypt the values (only the client can).

- Protected resource editing is available to credential administrators and application engineers for their assigned application(s).

- The API is agnostic of the consumer (Java Service, Single-Page-Application, Jenkins).

- Any access, edit, and delete is auditable.

- Only automated consumers, with the private key in hand, can decrypt their protected resource value.

- Use industry standard encryption and decryption so that we can support .NET, Java, and more.

- App IDs correlate to Troux App IDs

# <a name="access">Access Restrictions</a>

- Only runtime applications may decrypt

- Decryption keys are not available to the vault.

- Values are still only rendered to runtime systems with appropriate credentials.

- Only runtime applications may retrieve resource values.

- Just because you are privileged to edit values, or delete values, does not grant you privilege to retrieve.

- Only assigned credential administrators may edit values across all applications.

- Application administrators may edit values for their given application(s).

# <a name="nfr">Non-Functional Requirements</a>

## <a name="performance">Performance</a>

- Response times for retrieving values must be sub-second.

- Key retrieval for all application keys must be sub-second.

- Value retrieval of protected resource must support 10 requests per second sustained for 30 seconds.

- Estimate for batch SWARM stack start-ups (PaaS deployments).

- Protected resource editing should be sub-second per edit response time.

\_This service is an ideal candidate for a DB2 dependency with distributed

localized Redis or alternative caching solution. Response times will be

greatly reduced if caches are distributed for localized API deployments for the

Kansas and Fort Worth deployments.\_

## <a name="availability">Availability</a>

Driving Concerns:

- Vault services are required for scaling new instances and will block scaling solutions when down.

- Vault services are required for service restarts and will block restarts when down.

- Vault services are required for updating deployments and will block restarts when down.

Availability Requirements

- Vault API should retain functionality in the event of DB2 outage (caching).

- Vault API should scale as required based on usage (CPU/RAM/Network).

- Vault should be locally available (Fort Worth and Kansas) to combat network segregation/outages.

## <a name="support">Support</a>

After-hours support and SLAs must be defined. Outages impact: scaling, restarts,

and application updates. Response times for outages should be sub-hour

including after-hours support.

## <a name="compatible">Contract API Backwards Compatibility</a>

Contract APIs should be versioned in order to guarantee backwards compatibility.

Breaking API changes should trigger a new version in the API context URL.

# <a name="monitoring">Monitoring Support</a>

## <a name="heartbeat">Heartbeat</a>

Heartbeat API for service ready check (no cost). May be designated as service

identity URL.

## <a name="identity">Service Identity</a>

API service identity URL response should include:

- Version

- Service Name

- Host Machine ID, may be anonymized GUID

## <a name="metrics">Metrics</a>

API metrics should include:

- CPU usage

- DB pool usage

- Audit events triggered

\_Should be a minimal cost request\_

## <a name="health">Health</a>

API health should include:

- DB availability

- Cache availability

- Last successful request/operation

- Last failure request/operation

# <a name="artifacts">Artifacts</a>

## <a name="serverapi">Server API</a>

A REST API offering CRUD operations for managing protected resources meeting the

documented User Stories.

## <a name="serverui">Server UI</a>

A server UI offering human interaction capabilities for manual execution of the

auditing stories and protected resource editing.

## <a name="documentation">Adoption Documentation</a>

The list of desired documentation includes:

- Sample demo POSTMAN requests (demo functionality)

- REST API documentation

- Wizard/flow usage documentation

- In-UI help documentation