Hendrick Data API - Technical Planning Document

Executive Summary

This document outlines the comprehensive technical planning for the Hendrick Data API application, a partner data distribution platform designed to provide external business partners with secure, curated access to Hendrick Automotive Group's operational dealership data through isolated, containerized API environments.

Organization Overviews

Hendrick Automotive Group

Hendrick Automotive Group, founded in 1976 and headquartered in Charlotte, North Carolina, is the largest privately held automotive retail organization in the United States. Under the leadership of Chairman and CEO Rick Hendrick, the company operates 94 dealerships across 13 states, representing 130 franchises and 25 manufacturer brands ranging from mass-market vehicles (Toyota, Chevrolet, Honda) to luxury nameplates (BMW, Mercedes-Benz, Lexus). With over 10,000 employees, the organization generated $13.6 billion in revenue in 2024 by selling 206,000+ vehicles and servicing 2.6 million cars and trucks. The company's comprehensive automotive ecosystem includes new and pre-owned sales, financing, parts and accessories, service operations, collision centers, and a proprietary Certified Pre-Owned program, all unified by a customer-centric culture and commitment to operational excellence.

Reynolds & Reynolds

Reynolds & Reynolds, founded in 1866 and headquartered in Dayton, Ohio, is the automotive industry's leading provider of dealer management systems and technology solutions, serving over 70% of U.S. dealerships. The company specializes in integrated software platforms including their flagship ERA and POWER dealer management systems, which handle sales logistics, inventory management, accounting, service operations, and parts management across dealership departments. Their comprehensive Retail Management System (RMS) addresses the evolving customer-retailer relationship with solutions like FOCUS CRM for customer relationship management and docuPAD for interactive sales processes. As one of the three largest vendors in the dealership management software segment, Reynolds & Reynolds continues to expand through strategic acquisitions, including recent purchases of cybersecurity provider Proton Technologies (2022) and used vehicle technology startup AutoVision (2023), reinforcing their position as a comprehensive technology partner for automotive dealerships worldwide.

Business Requirements

Problem Statement

Hendrick Automotive Group receives operational dealership data from Reynolds & Reynolds through two distinct data feeds:

1. **XML Format**: Standardized structured data containing inventory, vehicle sales, and vehicle service information

2. **CSD (Custom Scheduled Download)**: Customized structured data generated from specific Reynolds & Reynolds reports, scheduled for nightly download

This operational data, owned by Hendrick Automotive Group, is currently stored immutably in AWS S3 buckets after being received through existing API endpoints. However, external business partners and vendors who provide services on behalf of Hendrick dealerships require curated access to this operational data to support their business processes.

Business Objectives

The Hendrick Data API application aims to:

1. **Enable Partner Data Distribution**: Provide curated operational data to external business partners through dedicated, secure API interfaces
2. **Implement Multi-Tenant Architecture**: Create isolated data environments where each partner accesses only the dealership data and attributes relevant to their business relationship
3. **Ensure Data Security**: Maintain complete data isolation between partners while providing role-based access to specific dealership operations
4. **Support Scalable Operations**: Enable efficient onboarding of new partners without impacting existing partner data access

Target Users & Use Cases

Primary Users:

- External vendors and business partners providing services to specific Hendrick dealerships
- Partner applications requiring automated access to dealership operational data

Key Use Cases:

- **Acme Corporation**: Requires inventory feed access for Dealership 1 only
- **Betamax Organization**: Needs inventory and service data for Dealerships 3, 4, and 5
- **Future Partners**: Various service providers requiring different data subsets from specific dealership groups

Business Requirements

Functional Requirements

1. **Partner-Specific Data Access**
    - Each partner receives access only to dealerships they service
    - Data filtering based on both dealership scope and attribute permissions
    - Support for different data types per partner (inventory, sales, service)
2. **Isolated Partner Environments**
    - Dedicated API application and database per partner
    - Complete data isolation between partners
    - Standardized endpoints with partner-specific data filtering

3. **Event-Driven Data Pipeline**

    - Messaging service alerts when new data is available in S3
    - Automated data import from immutable S3 storage to partner databases
    - Daily data updates with partner pull frequency of 1-2 times per day

4. **Data Curation & Transformation**

    - Minimal transformation of XML/CSD data to preserve original structure
    - Partner-specific data filtering based on authorized dealerships and data types
    - Support for historical data access based on partner needs

Non-Functional Requirements

1. **Security & Compliance**

    - Partner authentication and authorization
    - Data encryption in transit and at rest
    - Audit logging for all data access
    - Compliance with automotive industry data standards

2. **Performance & Scalability**

    - Support for multiple concurrent partners
    - Daily data processing within acceptable time windows
    - Scalable architecture to accommodate partner growth

3. **Availability & Reliability**

    - High availability for partner API access
    - Reliable data pipeline with error handling
    - Monitoring and alerting for system health

Success Criteria

1. **Partner Onboarding**: Ability to onboard new partners with isolated data environments within defined timeframes
2. **Data Access Control**: Zero data leakage between partners, with each partner accessing only authorized dealership data
3. **Operational Efficiency**: Daily data updates processed successfully with minimal manual intervention
4. **System Performance**: Partner API responses within acceptable latency thresholds
5. **Business Value**: Enhanced partner relationships through reliable, timely access to operational data

Constraints & Assumptions

Constraints:

- Source data is immutable and stored in AWS S3
- Daily data update frequency from Reynolds & Reynolds
- Partners require dedicated, isolated data access

Assumptions:

- Partners will pull data 1-2 times per day
- Messaging service will reliably indicate data availability
- Partner data requirements will remain relatively stable post-implementation

System Architecture & Design

Architecture Overview

The Hendrick Data API employs a multi-tenant, isolated partner architecture where each external business partner receives their own dedicated containerized environment. This approach ensures complete data isolation while maintaining operational simplicity and security.

High-Level Architecture

```
┌─────────────────┐    ┌─────────────────┐
┌──────────────────────────────┐
│ Reynolds &      │    │ Existing        │    │ AWS S3 Bucket
│
│ Reynolds        │───▶│ Hendrick API    │───▶│ (Immutable Storage)
│
│ (XML/CSD Data)  │    │ (Out of Scope)  │    │ - XML Files
(standardized)    │
└─────────────────┘    └─────────────────┘    │ - CSD Files (partner-
specific)  │

└──────────────────────────────┘

                                                          │
                                                          ▼

┌──────────────────────────────┐
                                              │ Messaging Service
│
                                              │ (Fan-out Pattern)
│
                                              │ - Broadcasts data
availability  │
                                              │ - No partner-specific
config    │
└──────────────────────────────┘

                                                          │
                                                          ▼
┌──────────────────────────────────────────────────────────────────────────
─────────────────────┐
│ EC2 Instance (Docker + Traefik)
│
│
│
│  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐
│
```

```
│  │ Acme Container   │  │ Betamax Container│  │ Partner N        │
│
│  │ ├─Config File    │  │ ├─Config File    │  │ ├─Config File    │
│
│  │ ├─Data Processor │  │ ├─Data Processor │  │ ├─Data Processor │
│
│  │ ├─Database       │  │ ├─Database       │  │ ├─Database       │
│
│  │ ├─API Server     │  │ ├─API Server     │  │ ├─API Server     │
│
│  │ └─Traefik Labels │  │ └─Traefik Labels │  │ └─Traefik Labels │
│
│  └──────────────────┘  └──────────────────┘  └──────────────────┘
│
│
│
┌─────────────────────────────────────────────────────────────────────────
────────────────┐ │
│  │ Traefik Reverse Proxy
│ │
│  │ - api.hendrick.com/acme/v1/* → Acme Container
│ │
│  │ - api.hendrick.com/betamax/v1/* → Betamax Container
│ │
│  │ - Automatic service discovery via Docker labels
│ │
│
└─────────────────────────────────────────────────────────────────────────
────────────────┘ │
└─────────────────────────────────────────────────────────────────────────
────────────────────┘
```

Core Architectural Principles

1. **Complete Partner Isolation**: Each partner operates in a dedicated container with its own database, ensuring zero data leakage between partners
2. **Event-Driven Processing**: Fan-out messaging pattern allows partners to self-select relevant data without central configuration management
3. **Standardized API Patterns**: Common endpoint structures (`/v1/inventory`, `/v1/sales`, `/v1/service`) with partner-specific data filtering
4. **Infrastructure Simplicity**: Single EC2 instance with Docker Compose orchestration for operational simplicity and easy recovery

Component Architecture

Partner Container Structure

Each partner container is a self-contained application including:

- **Configuration Management**: Partner-specific config file defining authorized dealerships and data types
- **Message Subscription**: Subscribes to fan-out messaging queue for S3 data availability notifications
- **Data Processing**: Retrieves and transforms XML/CSD data from S3 based on partner authorization
- **Local Database**: Partner-specific database for data storage and querying (SQLite or PostgreSQL)
- **API Server**: RESTful API endpoints for partner data access
- **Traefik Integration**: Automatic service discovery and routing via Docker labels

Messaging Architecture

Fan-Out Pattern Benefits:

- No central configuration required for new partners
- Partners self-filter based on their authorization
- Multiple services can consume the same data notifications
- Decoupled architecture supporting independent partner deployments

Message Structure (detailed schema TBD):

```json
{
  "timestamp": "2025-07-02T10:30:00Z",
  "dealership_uuid": "123e4567-e89b-12d3-a456-426614174000",
  "s3_bucket": "hendrick-data",
  "s3_path": "daily/2025-07-02/dealership-123/inventory.xml",
  "data_type": "inventory",
  "file_name": "structured-filename.xml"
}
```

Data Flow Architecture

1. **Data Ingestion**: Reynolds & Reynolds → Existing Hendrick API → AWS S3 (immutable storage)
2. **Notification**: S3 event triggers message to fan-out queue
3. **Partner Processing**:
   - Partner containers receive notification
   - Filter based on authorized dealerships and data types
   - Retrieve relevant data from S3
   - Transform and store in partner-specific database
4. **Data Access**: Partner applications query via standardized API endpoints

Infrastructure Architecture

AWS Infrastructure

- **Compute**: Single EC2 instance running Docker and Traefik
- **Storage**: EBS volumes for container persistence and partner databases
- **Networking**: Standard VPC with security groups for partner access
- **Messaging**: AWS SQS/SNS for fan-out messaging pattern

Container Orchestration

- **Docker Compose**: Partner deployment and lifecycle management
- **Traefik**: Automatic service discovery and reverse proxy
- **Path-based Routing**: `api.hendrick.com/partner-name/v1/endpoint`

Security Architecture

Data Isolation

- **Physical Isolation**: Separate containers and databases per partner
- **Network Isolation**: Container-level network segmentation
- **Access Control**: Partner-specific authentication and authorization
- **Data Filtering**: Dealership-level authorization enforced at the container level

Authentication & Authorization

- **API Keys**: Partner-specific authentication tokens
- **Request Validation**: Endpoint-level authorization checks
- **Audit Logging**: All data access logged for compliance

Scalability & Performance

Current Architecture Scalability

- **Vertical Scaling**: Single EC2 instance can accommodate multiple low-usage partners
- **Horizontal Migration Path**: Architecture supports migration to multi-instance deployment
- **Database Performance**: SQLite as starting point, PostgreSQL migration path for high-volume partners

Performance Characteristics

- **Data Volume**: Millions of records per partner (edge cases), daily access of thousands
- **Query Patterns**: Date-based queries, basic summaries, simple lookups
- **Response Time**: Standard REST API response times for partner applications

Deployment Architecture

Partner Onboarding Process

1. **Template Generation**: Script creates partner-specific configuration files
2. **Container Deployment**: Docker Compose deployment with Traefik labels
3. **Service Registration**: Automatic registration with Traefik reverse proxy
4. **Data Initialization**: TBD - Historical data loading strategy

Deployment Template Structure

```
templates/
├── docker-compose.yml.template
├── partner-config.yml.template
├── traefik-labels.template
└── deploy-partner.sh
```

Monitoring & Operations

System Monitoring

- **Container Health**: Docker container status and resource usage
- **API Performance**: Response time and error rate monitoring
- **Data Pipeline**: Message processing and S3 synchronization status

Operational Model

- **Deployment**: Kamal-based zero-downtime deployments on EC2 instances
- **Recovery**: System redeployment capability with Kamal for rapid reconstruction
- **Maintenance**: Rolling updates via container replacement

Technology Stack

Core Technologies

- **Containerization**: Docker + Docker Compose
- **Deployment**: Kamal for zero-downtime deployments
- **Reverse Proxy**: Traefik with automatic service discovery
- **Database**: Partner-specific databases (SQLite or PostgreSQL based on scale)
- **Authentication**: Auth0 with JWT token validation
- **Messaging**: AWS SQS/SNS for fan-out notifications (Redis for development)
- **Monitoring**: DataDog for comprehensive observability
- **Secrets Management**: 1Password with shared team vaults
- **Infrastructure**: AWS EC2, EBS, VPC (cloud-agnostic design)

Development & Testing

- **Local Development**: MinIO + Redis for realistic service mocking
- **Test Data**: Anonymized real XML/CSD files for comprehensive testing
- **Integration Testing**: End-to-end single partner validation with Auth0
- **API Testing**: Comprehensive REST API testing frameworks
- **Cloud Testing**: Multi-provider testing (AWS, Digital Ocean, etc.)

Future Considerations

Evolution Paths

- **Multi-Container Architecture**: Separate data processing from API serving if performance requires

- **Multi-Instance Deployment**: Scale to multiple EC2 instances for high availability
- **Managed Services**: Migration to ECS/Fargate for container orchestration
- **Database Scaling**: Migration from SQLite to PostgreSQL for individual partners as data volume requires

Data Transformation Enhancement

- **Configurable Transformation Engine**: Currently marked as TBD, architecture supports pluggable transformation logic
- **Historical Data Loading**: Strategy to be determined for initial partner data seeding
- **Real-time Processing**: Architecture supports migration from daily batch to real-time streaming if required

API Specification

API Overview

The Hendrick Data API provides standardized REST endpoints for external business partners to access their authorized operational data. Each partner receives isolated access through dedicated containerized environments with consistent API patterns while maintaining complete data security.

Base URL Structure

Pattern: `https://api.hendrick.com/{partner}/v1/{endpoint}`

Examples:

- `https://api.hendrick.com/acme/v1/inventory`
- `https://api.hendrick.com/betamax/v1/sales`
- `https://api.hendrick.com/partner-name/v1/service`

Authentication

Method: OAuth 2.0 Client Credentials flow via Auth0 **Token Type**: JWT (JSON Web Token) **Header**: `Authorization: Bearer {jwt_token}`

JWT Token Structure:

```
{
  "sub": "partner_client_id",
  "scope": "hendrick:api:read",
  "partner_id": "acme",
  "iss": "https://hendrick.auth0.com/",
  "exp": 1625097600
}
```

Standard Endpoints

All partners have access to the following standardized endpoints with data filtered to their authorized dealerships:

1. Inventory Data

```
GET /{partner}/v1/inventory
```

2. Sales Data

```
GET /{partner}/v1/sales
```

3. Service Data

```
GET /{partner}/v1/service
```

Query Parameters

Date-Based Filtering

Parameter	Type	Description	Example
`period`	string	Predefined time periods	`today`, `yesterday`, `last_week`, `last_month`
`date`	string	Specific date (YYYY-MM-DD)	`2025-07-02`
`start_date`	string	Date range start (YYYY-MM-DD)	`2025-07-01`
`end_date`	string	Date range end (YYYY-MM-DD)	`2025-07-07`

Pagination Parameters

Parameter	Type	Description	Default	Maximum
`offset`	integer	Number of records to skip	`0`	No limit
`limit`	integer	Number of records to return	`1000`	`10000`

Example Queries

```
GET /acme/v1/inventory
GET /acme/v1/inventory?period=today
GET /acme/v1/inventory?date=2025-07-02
GET /acme/v1/inventory?start_date=2025-07-01&end_date=2025-07-07
GET /acme/v1/inventory?period=today&offset=1000&limit=500
```

Response Format

Success Response Structure

```
{
  "metadata": {
    "count": 1000,
    "total_records": 5000,
    "offset": 0,
    "limit": 1000,
    "has_next": true,
    "next_offset": 1000,
    "query": {
      "period": "today"
    },
    "data_freshness": {
      "last_updated": "2025-07-02T02:30:00Z",
      "data_date": "2025-07-02",
      "status": "current"
    },
    "generated_at": "2025-07-02T10:00:00Z"
  },
  "inventory": [
    {
      "vin": "1HGBH41JXMN109186",
      "dealership_id": "123e4567-e89b-12d3-a456-426614174000",
      "make": "Honda",
      "model": "Accord",
      "year": 2023,
      "status": "available",
      "created_date": "2025-07-02T08:30:00Z"
    }
  ]
}
```

Empty Dataset Response

```
{
  "metadata": {
    "count": 0,
    "total_records": 0,
    "offset": 0,
    "limit": 1000,
    "has_next": false,
    "query": {
      "period": "today"
    },
    "data_freshness": {
      "last_updated": "2025-07-01T23:45:00Z",
      "requested_date": "2025-07-02",
      "status": "processing",
      "message": "Data for 2025-07-02 is being processed. Latest available
data is from 2025-07-01."
    },
    "generated_at": "2025-07-02T06:00:00Z"
  },
  "inventory": []
}
```

Error Responses

Error Response Structure

```
{
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error description",
    "details": {
      "parameter": "parameter_name",
      "provided_value": "invalid_value",
      "expected_format": "expected_format"
    }
  }
}
```

Common Error Examples

Invalid Date Format (400 Bad Request)

```
{
  "error": {
    "code": "INVALID_DATE_FORMAT",
    "message": "Date parameter must be in YYYY-MM-DD format",
    "details": {
      "parameter": "date",
      "provided_value": "2025/07/02",
      "expected_format": "YYYY-MM-DD"
    }
  }
}
```

Pagination Limit Exceeded (400 Bad Request)

```
{
  "error": {
    "code": "INVALID_LIMIT",
    "message": "Limit exceeds maximum allowed value of 10000",
    "details": {
      "parameter": "limit",
      "provided_value": 15000,
      "maximum_allowed": 10000
    }
  }
}
```

Invalid Offset (400 Bad Request)

```
{
  "error": {
    "code": "INVALID_OFFSET",
    "message": "Offset cannot be negative",
    "details": {
      "parameter": "offset",
      "provided_value": -100,
      "minimum_allowed": 0
    }
  }
}
```

Authentication Failed (401 Unauthorized)

```
{
  "error": {
    "code": "AUTHENTICATION_FAILED",
    "message": "Invalid or expired JWT token",
    "details": {
      "token_status": "expired",
      "expires_at": "2025-07-02T10:00:00Z"
    }
  }
}
```

HTTP Status Codes

Status Code	Usage	Description

Status Code	Usage	Description
`200 OK`	Success	Data returned successfully (including empty datasets)
`400 Bad Request`	Client Error	Invalid query parameters, malformed requests
`401 Unauthorized`	Auth Error	Missing, invalid, or expired JWT token
`403 Forbidden`	Auth Error	Valid token but insufficient permissions
`404 Not Found`	Client Error	Invalid endpoint or partner route
`500 Internal Server Error`	Server Error	System errors, database issues

Content Type

Request: `Accept: application/json` **Response**: `Content-Type: application/json`

Note: Only JSON format is supported. Sample code will be provided to partners for client-side conversion to CSV or other formats as needed.

Rate Limiting

Rate limiting is implemented for security purposes to protect against malicious access patterns and potential abuse. While partners typically access data 1-2 times per day, adaptive rate limiting provides protection against:

- Suspicious access patterns and bulk data extraction attempts
- Brute force attacks and credential abuse
- Resource exhaustion and denial of service attacks
- Unauthorized automated access attempts

Data Schema Definitions

Metadata Schema

```
{
  "count": "integer - Records in current response",
  "total_records": "integer - Total matching records",
  "offset": "integer - Current offset position",
  "limit": "integer - Current limit setting",
  "has_next": "boolean - More records available",
  "next_offset": "integer - Next offset for pagination",
  "query": "object - Query parameters used",
  "data_freshness": {
    "last_updated": "string - ISO 8601 timestamp of last data update",
    "data_date": "string - Date of the data (YYYY-MM-DD)",
    "status": "string - current|processing|stale"
  },
  "generated_at": "string - ISO 8601 timestamp of response generation"
}
```

Common Data Fields

All endpoints share common fields for consistency:

```
{
  "dealership_id": "string - UUID of the dealership",
  "created_date": "string - ISO 8601 timestamp",
  "updated_date": "string - ISO 8601 timestamp",
  "record_type": "string - inventory|sales|service"
}
```

Note: Specific data schemas for inventory, sales, and service records will be defined based on XML/CSD data structure analysis and partner requirements.

API Versioning

- **Current Version**: v1
- **Versioning Strategy**: URL-based versioning (`/v1/`)
- **Backward Compatibility**: Major version changes will maintain parallel versions
- **Deprecation Policy**: Minimum 6-month notice for version deprecation

Testing & Integration

Sample Client Code

Sample integration code will be provided for:

- JWT token authentication
- Pagination handling
- Error handling
- JSON to CSV conversion
- Rate limiting best practices

Development Environment

- Test endpoints available with sample data
- Auth0 test credentials for integration testing
- Postman collection for API exploration

Data Architecture

Overview

The Hendrick Data API employs a standardized database architecture across all partner containers, ensuring consistent troubleshooting and operational management while maintaining complete data isolation between partners. Each partner receives their own dedicated database with identical schema, populated only with data they are authorized to access.

Database Technology Strategy

Starting Point: SQLite for operational simplicity and complete isolation **Evolution Path**: PostgreSQL migration for individual partners as data volume requires **Decision Factors**: Partner data volume, query complexity, and performance requirements

Standardized Database Schema

All partner containers implement identical database schemas regardless of their data authorization. This standardization ensures:

- Consistent troubleshooting across all partners
- Uniform operational procedures
- Simplified deployment and maintenance
- Easy identification of data availability issues

Core Data Tables

Inventory Data Table

```
CREATE TABLE inventory_data (
  id INTEGER PRIMARY KEY,
  dealership_id TEXT NOT NULL,
  data_date DATE NOT NULL,
  json_data JSON NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_inventory_dealership_date (dealership_id, data_date),
  INDEX idx_inventory_date (data_date)
);
```

Sales Data Table

```
CREATE TABLE sales_data (
  id INTEGER PRIMARY KEY,
  dealership_id TEXT NOT NULL,
  data_date DATE NOT NULL,
  json_data JSON NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_sales_dealership_date (dealership_id, data_date),
  INDEX idx_sales_date (data_date)
);
```

Service Data Table

```sql
CREATE TABLE service_data (
  id INTEGER PRIMARY KEY,
  dealership_id TEXT NOT NULL,
  data_date DATE NOT NULL,
  json_data JSON NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_service_dealership_date (dealership_id, data_date),
  INDEX idx_service_date (data_date)
);
```

Dealership Information Table

```sql
CREATE TABLE dealership_info (
  dealership_id TEXT PRIMARY KEY,
  name TEXT,
  address TEXT,
  phone TEXT,
  json_data JSON,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Data Freshness Tracking Table

```sql
CREATE TABLE data_freshness (
  data_type TEXT NOT NULL,            -- 'inventory', 'sales', 'service'
  data_date DATE NOT NULL,            -- Date of the data
  last_updated TIMESTAMP NOT NULL,    -- When processing completed
  record_count INTEGER DEFAULT 0,     -- Number of records processed
  status TEXT DEFAULT 'current',      -- 'current', 'processing', 'stale'
  json_data JSON,                     -- Additional metadata if needed
  PRIMARY KEY (data_type, data_date),
  INDEX idx_freshness_type_updated (data_type, last_updated)
);
```

Data Transformation Strategy

Minimal Transformation Approach

Principle: Preserve maximum information with minimal processing to ensure data fidelity and troubleshooting capability.

Transformation Process:

1. **Source Data**: XML/CSD files from Reynolds & Reynolds
2. **Conversion**: Direct XML/CSD to JSON conversion with minimal structural changes
3. **Preservation**: All original field names, values, and hierarchical structure maintained
4. **Extraction**: Only essential fields extracted for database indexing (dealership_id, data_date)

5. **Storage**: Complete original data stored in `json_data` column using standard naming convention

Example Data Flow:

```
XML Input:
<vehicle>
  <vin>1HGBH41JXMN109186</vin>
  <dealership_id>123e4567-e89b-12d3-a456-426614174000</dealership_id>
  <make>Honda</make>
  <model>Accord</model>
  <complex_nested_data>...</complex_nested_data>
</vehicle>

JSON Storage (json_data column):
{
  "vin": "1HGBH41JXMN109186",
  "dealership_id": "123e4567-e89b-12d3-a456-426614174000",
  "make": "Honda",
  "model": "Accord",
  "complex_nested_data": { ... }
}

Database Record:
id: 1
dealership_id: "123e4567-e89b-12d3-a456-426614174000" (extracted for
indexing)
data_date: "2025-07-02" (extracted for indexing)
json_data: {complete original data as JSON}
created_at: "2025-07-02T10:30:00Z"
```

Partner Authorization Model

Configuration-Based Authorization

Partner Config File Structure:

```
# partner-config.yml
partner:
  id: "acme"
  name: "Acme Corporation"

authorization:
  data_types: ["inventory", "dealership_info"]
  dealerships:
    - "123e4567-e89b-12d3-a456-426614174000"
    - "456e7890-e12b-34c5-d678-901234567890"

processing:
  populate_tables: ["inventory_data", "dealership_info"]
  ignore_tables: ["sales_data", "service_data"]
```

Data Population Logic

All Tables Created: Every partner container creates all standard tables **Selective Population**: Only authorized data types are populated during processing **Empty Table Behavior**: Unauthorized tables remain empty but exist for consistency **API Response**: Unauthorized endpoints return empty datasets with proper metadata

Database Initialization

Migration-Based Schema Management

Migration Script Structure:

```
migrations/
├── 001_initial_schema.sql     -- Core tables creation
├── 002_add_indexes.sql        -- Performance indexes
├── 003_add_data_freshness.sql -- Data tracking table
└── 999_seed_data.sql          -- Optional test data
```

Container Startup Process:

1. Check current database schema version
2. Execute pending migrations in sequential order
3. Log migration results for audit trail
4. Validate schema integrity
5. Start API service only after successful migration

Migration Tracking Table:

```sql
CREATE TABLE schema_migrations (
  version TEXT PRIMARY KEY,
  applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  description TEXT
);
```

Data Processing Pipeline

S3 to Database Flow

Processing Sequence:

1. **Message Reception**: Container receives S3 data availability notification
2. **Authorization Check**: Validate data type and dealership against partner config
3. **Data Retrieval**: Download authorized data files from S3
4. **Transformation**: Convert XML/CSD to JSON with minimal processing

5. **Database Insert**: Store data in appropriate tables with extracted index fields
6. **Freshness Update**: Update data_freshness table with processing results
7. **Cleanup**: Remove temporary files and log processing completion

Error Handling:

- Invalid data format: Log error, skip record, continue processing
- Database constraints: Log violation, attempt retry with data cleanup
- S3 access issues: Log error, schedule retry with exponential backoff
- Partial processing: Update data_freshness with partial results and error status

Data Retention Strategy

Current Approach: Indefinite retention of all historical data **Rationale**:

- Data can be reloaded from S3 if needed
- Partner requirements may vary for historical access
- Storage costs are manageable initially
- Simplifies operational procedures

Future Considerations:

- Implement configurable retention policies per partner
- Archive older data to compressed formats
- Tiered storage for recent vs. historical data
- Partner-specific retention requirements

Performance Optimization

Indexing Strategy

Primary Indexes:

- `(dealership_id, data_date)`: Supports partner's primary query patterns
- `(data_date)`: Supports date-range queries across dealerships
- `(data_type, last_updated)`: Supports data freshness queries

Query Optimization:

- Date-based queries leverage primary indexes
- JSON column searches minimize full-table scans where possible
- Pagination offset queries optimized for sequential access patterns

Database Scaling Considerations

SQLite Limitations:

- Concurrent write limitations
- File size practical limits
- Complex query performance

PostgreSQL Migration Triggers:

- Database file size exceeds 10GB
- Concurrent access requirements increase
- Complex analytical queries needed
- Partner requires advanced database features

Data Consistency and Integrity

Consistency Model

Within Partner: Strong consistency within each partner's isolated database **Across Partners**: No consistency requirements (complete isolation) **Source of Truth**: S3 immutable storage remains authoritative

Data Validation

Source Data Validation:

- XML/CSD format validation before processing
- Required field presence verification
- Dealership ID validation against authorized list

Database Constraints:

- NOT NULL constraints on critical fields
- Foreign key relationships where applicable
- Unique constraints on business keys where identified

Backup and Recovery

Backup Strategy:

- Database files included in container volume backups
- S3 source data serves as authoritative backup
- Migration scripts version controlled for schema recovery

Recovery Procedures:

1. **Container Loss**: Redeploy container and reload data from S3
2. **Data Corruption**: Drop database, run migrations, reload from S3
3. **Partial Data Loss**: Identify missing date ranges, reload specific periods
4. **Schema Issues**: Roll back migrations, fix, re-apply

Monitoring and Observability

Database Metrics

Performance Monitoring:

- Query response times by endpoint

- Database file size growth trends
- Index usage statistics
- Failed query patterns

Data Quality Monitoring:

- Record count validation against source files
- Data freshness lag monitoring
- Processing error rates by data type
- Schema migration success rates

Alerting Thresholds:

- Database size approaching limits
- Data processing delays exceeding SLA
- High error rates in data transformation
- Schema migration failures

Technical Implementation Plan

Implementation Strategy

The Hendrick Data API will be implemented using a phased approach that prioritizes rapid validation of core concepts while building toward a scalable, production-ready system. The implementation leverages Claude-assisted development for accelerated delivery timelines and focuses on proving single-partner functionality before scaling to multiple partners.

Development Approach

Claude-Assisted Development: Utilizing AI-assisted development tools to accelerate code generation, configuration management, and documentation creation, enabling aggressive timelines while maintaining code quality.

Cloud Portability Focus: Building with containerized, cloud-agnostic technologies to enable testing and deployment across multiple infrastructure providers (AWS, Digital Ocean, etc.).

Security-First Integration: Real authentication and security components integrated from Phase 1 to ensure production-ready security practices from the beginning.

Implementation Phases

Phase 1: Single Partner MVP (1-2 weeks)

Objective: Prove end-to-end functionality with realistic mock services and a single test partner.

Core Components:

- Single partner container with complete isolation
- Auth0 JWT authentication integration
- Traefik reverse proxy with automatic service discovery
- Complete API endpoints (inventory, sales, service)

- Realistic data processing pipeline with mocked external services

Technology Stack:

- **Authentication**: Real Auth0 integration with JWT validation
- **Container Orchestration**: Docker Compose for local development
- **Reverse Proxy**: Traefik with automatic service discovery
- **Database**: SQLite with migration scripts and standardized schema
- **Mock S3**: MinIO for realistic object storage simulation
- **Mock Messaging**: Redis for pub/sub message simulation
- **Test Data**: Anonymized real XML/CSD data for realistic testing

Key Deliverables:

1. **Development Environment Setup**
    - Docker Compose configuration with all services
    - 1Password integration for secrets management
    - Shared development vault for team access
    - Local development workflow documentation
2. **Partner Container Architecture**
    - Standardized database schema with migration system
    - Partner configuration file structure
    - Data processing pipeline (XML/CSD → JSON)
    - API server with authentication middleware
3. **API Implementation**
    - RESTful endpoints with standardized URL structure
    - Pagination with offset/limit parameters
    - Data freshness tracking and metadata responses
    - Comprehensive error handling with detailed messages
    - Authentication integration with Auth0 JWT validation
4. **Data Pipeline**
    - Mock S3 integration with MinIO
    - Mock messaging system with Redis
    - XML/CSD to JSON transformation with minimal processing
    - Data freshness tracking and status management
    - Partner authorization and data filtering logic
5. **Testing Framework**
    - API endpoint testing with realistic data scenarios
    - Authentication flow validation
    - Data isolation verification
    - Error handling and edge case testing
    - Performance baseline measurement

Success Criteria:

- Single partner can authenticate and access authorized data only
- All API endpoints return proper responses with metadata
- Data processing pipeline handles mock S3 events correctly
- Complete isolation demonstrated (partner only sees their data)
- System handles typical query patterns (date-based, pagination)

Phase 2: Production Infrastructure (2-3 weeks)

Objective: Deploy single partner to production environment with real AWS services and monitoring.

Migration Strategy: Direct migration from Phase 1 mocked services to production equivalents while maintaining identical application behavior.

Infrastructure Changes:

- **MinIO → AWS S3**: Replace mock object storage with production S3 buckets
- **Redis → AWS SQS/SNS**: Replace mock messaging with production message queues
- **Local Files → Real Data**: Replace anonymized test data with actual XML/CSD feeds
- **Development Auth0 → Production Auth0**: Migrate to production Auth0 tenant and credentials

Production Components:

- **Deployment**: Kamal for zero-downtime deployments and container orchestration
- **Monitoring**: DataDog integration for comprehensive observability
- **Infrastructure**: Single EC2 instance with Docker containerization
- **Security**: Production VPC, security groups, and SSL/TLS certificates
- **Secrets Management**: 1Password integration for production credentials

Key Deliverables:

1. **Production Environment Setup**
    - AWS account and VPC configuration
    - EC2 instance provisioning and security hardening
    - Production Auth0 tenant configuration
    - SSL/TLS certificate management
    - Domain configuration for api.hendrick.com
2. **Kamal Deployment Configuration**
    - Kamal deployment scripts and configuration
    - Container build and deployment pipeline
    - Health check and rollback procedures
    - Environment-specific configuration management
    - Secrets integration with 1Password
3. **AWS Services Integration**
    - S3 bucket configuration for XML/CSD data storage

   - SQS/SNS setup for data availability notifications
   - IAM roles and permissions for container access
   - CloudWatch integration for basic AWS monitoring
   - Security group and network access controls

4. **DataDog Monitoring Setup**

   - Container metrics and health monitoring
   - API endpoint performance tracking
   - Data processing pipeline monitoring
   - Custom business metrics (records processed, data freshness)
   - Alert configuration for critical issues
   - Dashboard creation for operational visibility

5. **Operational Procedures**

   - Deployment runbook and procedures
   - Monitoring and alerting configuration
   - Backup and recovery procedures
   - Log aggregation and retention policies
   - Security incident response procedures

Success Criteria:

- Single partner successfully deployed and accessible in production
- Real S3 and messaging integration working correctly
- DataDog monitoring providing comprehensive visibility
- Zero-downtime deployments working with Kamal
- All security controls functioning properly

Phase 3: Partner Onboarding Automation (1-2 weeks)

Objective: Create streamlined process for onboarding new partners without manual configuration.

Automation Components:

- Partner configuration templates
- Automated container deployment scripts
- Documentation and integration guides
- Configuration validation tools

Key Deliverables:

1. **Partner Onboarding Templates**

   - Partner configuration file templates
   - Docker Compose template generation
   - Kamal deployment configuration templates
   - Auth0 client configuration automation
   - Documentation template generation

2. **Deployment Automation Scripts**
    - Partner container generation scripts
    - Configuration validation tools
    - Automated testing for new partner deployments
    - Template customization and deployment tools
    - Integration testing automation
3. **Documentation and Guides**
    - Partner integration guide and API documentation
    - Sample client code for common programming languages
    - Authentication and error handling examples
    - Data format documentation and examples
    - Troubleshooting guides and common issues
4. **Operational Tools**
    - Partner management dashboard (optional)
    - Configuration validation tools
    - Health check and monitoring setup automation
    - Log analysis and debugging tools
    - Performance monitoring setup automation

Success Criteria:

- New partner can be onboarded with minimal manual intervention
- Partner deployment process is documented and repeatable
- Integration testing validates new partner functionality
- Operational tools support ongoing partner management

Phase 4: Multi-Partner Validation (1-2 weeks)

Objective: Validate architecture scalability and partner isolation with multiple production partners.

Validation Components:

- Second partner deployment with different authorization scope
- Concurrent operation testing
- Performance and resource utilization analysis
- Security validation and penetration testing

Key Deliverables:

1. **Second Partner Deployment**
    - Deploy second partner with different data authorization
    - Validate complete data isolation between partners
    - Test concurrent API access and data processing
    - Performance impact analysis with multiple partners

2. **Scalability Testing**
   - Resource utilization monitoring with multiple partners
   - API performance under concurrent load
   - Data processing efficiency with multiple data streams
   - Container resource optimization and tuning
3. **Security Validation**
   - Partner isolation security testing
   - Authentication and authorization validation
   - Data access control verification
   - Security audit and vulnerability assessment
4. **Operational Validation**
   - Multi-partner monitoring and alerting
   - Deployment and rollback procedures with multiple partners
   - Backup and recovery testing
   - Disaster recovery procedures validation

Success Criteria:

- Multiple partners operating concurrently without interference
- Complete data isolation maintained under load
- Performance within acceptable parameters
- Security controls validated for multi-tenant operation

Technology Stack Summary

Development Tools

- **AI Assistant**: Claude for accelerated development
- **Version Control**: Git with GitHub for collaboration
- **Secrets Management**: 1Password with shared team vaults
- **Documentation**: Markdown with automated generation tools

Core Technologies

- **Containerization**: Docker with Docker Compose for development
- **Reverse Proxy**: Traefik with automatic service discovery
- **Database**: SQLite (starting point) with PostgreSQL migration path
- **Authentication**: Auth0 with JWT token validation
- **Deployment**: Kamal for production deployments

AWS Production Services

- **Compute**: EC2 instances with Docker containerization
- **Storage**: S3 for immutable data storage
- **Messaging**: SQS/SNS for event-driven data pipeline

- **Networking**: VPC with security groups and load balancing
- **Monitoring**: CloudWatch for basic AWS service monitoring

Development/Mock Services

- **Mock Object Storage**: MinIO for S3 simulation
- **Mock Messaging**: Redis for pub/sub simulation
- **Test Data**: Anonymized real XML/CSD data
- **Local Development**: Docker Compose with service simulation

Monitoring and Operations

- **Application Monitoring**: DataDog for comprehensive observability
- **Log Management**: DataDog log aggregation and analysis
- **Performance Monitoring**: DataDog APM for API and container metrics
- **Alerting**: DataDog alerts for operational and business metrics

Risk Mitigation Strategies

Technical Risks

1. **Database Scalability**: SQLite limitations mitigated by PostgreSQL migration path
2. **Container Resource Limits**: Monitoring and resource optimization from Phase 2
3. **Authentication Integration**: Real Auth0 integration from Phase 1 reduces risk
4. **Data Processing Complexity**: Minimal transformation approach reduces complexity

Operational Risks

1. **Deployment Issues**: Kamal provides rollback capabilities and zero-downtime deployments
2. **Monitoring Gaps**: DataDog integration from Phase 2 provides comprehensive visibility
3. **Security Vulnerabilities**: Security-first approach with Auth0 and proper isolation
4. **Partner Onboarding Complexity**: Automation and templates reduce manual errors

Business Risks

1. **Cloud Provider Lock-in**: Cloud-agnostic design enables provider flexibility
2. **Scalability Limitations**: Phased approach validates architecture at each step
3. **Data Quality Issues**: Real data testing from Phase 1 identifies issues early
4. **Partner Integration Challenges**: Comprehensive documentation and sample code

Success Metrics and KPIs

Technical Metrics

- **API Response Time**: < 500ms for 95% of requests
- **System Uptime**: > 99.5% availability
- **Data Processing Latency**: Daily data available within 2 hours of S3 upload
- **Container Resource Utilization**: < 80% CPU and memory usage
- **Database Performance**: Query response time < 100ms for typical partner queries

Operational Metrics

- **Partner Onboarding Time**: < 4 hours from request to active API access
- **Deployment Success Rate**: > 99% successful deployments with zero downtime
- **Security Incident Rate**: Zero data leakage or unauthorized access incidents
- **Mean Time to Resolution**: < 2 hours for critical issues

Business Metrics

- **Partner Satisfaction**: Measured through API usage patterns and feedback
- **Data Freshness**: Partners receive data within agreed SLA timeframes
- **System Reliability**: No partner service interruptions during business hours
- **Cost Efficiency**: Total system cost per partner within budget parameters

Development Resources and Timeline

Phase 1: 1-2 weeks

- **Week 1**: Foundation setup and core component development
- **Week 2**: API implementation and testing/refinement

Phase 2: 2-3 weeks

- **Week 1**: AWS infrastructure setup and service integration
- **Week 2**: Production deployment and monitoring configuration
- **Week 3**: Operational validation and performance tuning

Phase 3: 1-2 weeks

- **Week 1**: Automation tool development and template creation
- **Week 2**: Documentation and integration guide development

Phase 4: 1-2 weeks

- **Week 1**: Second partner deployment and concurrent testing
- **Week 2**: Performance validation and security testing

Total Estimated Timeline: 5-9 weeks with aggressive Claude-assisted development approach

Next Steps

1. **Immediate Actions**:
    - Set up development environment with 1Password integration
    - Create shared vault for team credential access
    - Initialize git repository with initial project structure
2. **Phase 1 Kickoff**:
    - Implement Docker Compose development environment

  - Set up Auth0 development tenant and test credentials
  - Create anonymized test data set from real XML/CSD samples

3. **Stakeholder Communication**:

  - Regular progress updates and demo sessions
  - Partner feedback integration throughout development
  - Business stakeholder review at end of each phase

Security & Compliance

Security Overview

The Hendrick Data API implements comprehensive security controls to protect Confidential-classified operational data containing personally identifiable information (PII) and business-sensitive automotive dealership information. The security architecture follows defense-in-depth principles with multiple layers of protection, comprehensive audit logging, and strict access controls.

Data Classification

Classification Level: **Confidential**

All operational data within the Hendrick Data API is classified as Confidential due to the presence of:

- Customer personally identifiable information (PII) in sales and service records
- Business-sensitive operational data across all data types
- Financial and pricing information
- Dealership operational details

Security Requirements per DATA_CLASSIFICATION.md:

- Strong authentication and multi-factor authentication (MFA)
- TLS 1.3 encryption in transit
- AES-256 encryption at rest with managed keys
- Comprehensive audit logging and monitoring
- Quarterly access reviews and certifications
- Data loss prevention measures
- Incident response procedures

Authentication and Authorization Architecture

Auth0 Integration

Primary Authentication: OAuth 2.0 Client Credentials flow via Auth0 **Token Type**: JSON Web Tokens (JWT) with comprehensive validation **Secure Credential Provisioning**: Hendrick provides secure client credentials to partners (no self-registration)

JWT Token Structure and Validation:

```
```

```
{
  "iss": "https://hendrick.auth0.com/",
  "sub": "partner_client_id",
  "aud": "hendrick-data-api",
  "scope": "hendrick:api:read",
  "partner_id": "acme",
  "exp": 1625097600,
  "iat": 1625094000,
  "jti": "unique-token-id"
}
```

Token Security Measures:

- **Token Rotation**: Automatic token rotation every 24 hours
- **Token Revocation**: Immediate revocation capability for compromised credentials
- **Scope Validation**: Strict validation of token scope and partner identity
- **Expiration Enforcement**: Short-lived tokens with automatic refresh
- **Replay Protection**: JWT ID (jti) validation to prevent token replay attacks
- **Secure Provisioning**: Hendrick securely provisions and manages all partner credentials

Authorization Model

Partner Isolation: Complete data isolation implemented through:

- Dedicated database per partner container
- Partner-specific data filtering at the database level
- Container-level network isolation
- Independent authentication tokens per partner

Access Control Matrix:

Partner	Authorized Dealerships	Data Types	Database Tables
Acme Corp	Dealership 1	Inventory	inventory_data, dealership_info
Betamax	Dealerships 3,4,5	Sales, Service	sales_data, service_data, dealership_info
Future Partners	Variable	Variable	Configurable per partner

Authorization Validation Process:

1. JWT token signature verification against Auth0 public keys
2. Token expiration and not-before-time validation
3. Audience and issuer claim verification
4. Partner ID extraction and validation
5. Container-level authorization check
6. Database-level data filtering enforcement

Encryption and Data Protection

Encryption in Transit

External Communications:

- **TLS 1.3** for all API endpoints with perfect forward secrecy
- **Certificate Pinning** for critical Auth0 communications
- **HSTS (HTTP Strict Transport Security)** headers enforced
- **Certificate Transparency** monitoring for SSL/TLS certificates

Internal Communications:

- **TLS 1.3** for container-to-container communications
- **Encrypted service mesh** for sensitive internal data flows
- **VPC-level encryption** for AWS service communications

Encryption at Rest

Database Encryption:

- **AES-256** encryption for all partner databases
- **Transparent Data Encryption (TDE)** for database files
- **Encrypted backups** with separate key management
- **Field-level encryption** for PII data elements

Key Management:

- **1Password** for development and staging environment secrets
- **AWS KMS** for production encryption key management
- **Key rotation** policies with automated rotation every 90 days
- **Hierarchical key structure** with master and data encryption keys

Audit Data Encryption:

- **AES-256** encryption for audit response archives
- **Separate encryption keys** for audit data isolation
- **Immutable audit logs** with cryptographic integrity verification

Network Security

Infrastructure Security

AWS VPC Configuration:

- **Private subnets** for all application components
- **Network Access Control Lists (NACLs)** for subnet-level filtering
- **Security Groups** implementing least-privilege access
- **VPC Flow Logs** for network traffic monitoring and analysis

Container Security:

- **Docker security best practices** with non-root container execution
- **Resource limits** and security contexts for all containers
- **Network policies** restricting inter-container communication
- **Container image scanning** for vulnerabilities and malware

API Security

Rate Limiting and DDoS Protection:

- **Adaptive rate limiting** based on partner usage patterns
- **IP-based rate limiting** with geographic restrictions if needed
- **Request size limits** to prevent resource exhaustion attacks
- **Connection throttling** and timeout enforcement

Input Validation and Sanitization:

- **Strict input validation** for all query parameters
- **SQL injection prevention** through parameterized queries
- **Cross-site scripting (XSS) protection** for any web interfaces
- **Parameter tampering detection** and prevention

API Security Headers:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'
```

Comprehensive Audit Logging

Real-Time Audit Logging

Authentication Events:

```json
{
  "event_type": "authentication",
  "timestamp": "2025-07-02T10:30:00Z",
  "partner_id": "acme",
  "jwt_subject": "acme_client_id",
  "ip_address": "192.168.1.100",
  "user_agent": "Partner-API-Client/1.0",
  "auth_result": "success|failure",
  "failure_reason": "invalid_token|expired_token|invalid_signature",
  "session_id": "unique-session-identifier"
}
```

API Access Events:

```
{
  "event_type": "api_access",
  "timestamp": "2025-07-02T10:30:15Z",
  "partner_id": "acme",
  "session_id": "unique-session-identifier",
  "endpoint": "/acme/v1/inventory",
  "http_method": "GET",
  "query_parameters": {
    "date": "2025-07-02",
    "offset": 0,
    "limit": 1000
  },
  "response_status": 200,
  "response_size_bytes": 2048576,
  "processing_time_ms": 245,
  "records_returned": 856,
  "contains_pii": true,
  "dealerships_accessed": ["uuid1", "uuid2"]
}
```

Security Events:

```
{
  "event_type": "security_event",
  "timestamp": "2025-07-02T10:30:30Z",
  "severity": "high|medium|low",
  "event_category":
"suspicious_activity|failed_authentication|rate_limit_exceeded",
  "partner_id": "acme",
  "details": {
    "failed_attempts": 5,
    "time_window": "5_minutes",
    "blocked_duration": "15_minutes"
  },
  "automated_response": "account_temporarily_locked"
}
```

Response Archive System

Coordinated Audit Pipeline: The audit system implements a coordinated pipeline to eliminate race conditions and ensure perfect audit accuracy:

1. **Partner Request Processing**:
    - Process partner API request normally
    - Return response immediately (no delay)
    - Queue audit job with request details
2. **Database Update Coordination**:

    - S3 data availability message received
    - Check for pending audit jobs
    - If audits pending: Queue database update
    - If no audits: Process update immediately

3. **Audit Job Execution**:

    - Execute identical query with same parameters
    - Capture complete response data
    - Encrypt and store to S3 audit storage
    - Mark audit job as complete

4. **Database Update Release**:

    - When all audits complete: Release queued database updates
    - Process fresh data for subsequent partner requests
    - Maintain audit integrity without race conditions

Audit Storage Architecture:

```
S3 Audit Bucket Structure:
audit-responses/
├── 2025/07/02/
│   ├── partner-acme/
│   │   ├── inventory-143000-uuid.json.enc
│   │   └── sales-143030-uuid.json.enc
│   ├── partner-betamax/
│   │   └── service-143045-uuid.json.enc
│   └── audit-metadata/
│       └── daily-summary-20250702.json
```

Audit Data Retention:

- **7-year retention** period for all audit data
- **Immutable storage** with versioning and legal hold capabilities
- **Automated lifecycle management** with cost-optimized storage tiers
- **Secure deletion** procedures after retention expiration

Threat Model and Risk Assessment

Primary Threat Vectors

1. Partner Credential Compromise

- **Risk Level**: High
- **Mitigation**: Token rotation, MFA, behavioral monitoring
- **Detection**: Unusual access patterns, geographic anomalies
- **Response**: Immediate token revocation, account lockout

2. API Endpoint Attacks

- **Risk Level**: Medium
- **Attack Types**: Injection attacks, parameter tampering, enumeration
- **Mitigation**: Input validation, parameterized queries, rate limiting
- **Detection**: Anomalous query patterns, error rate spikes

3. Container Security Breaches

- **Risk Level**: Medium
- **Attack Types**: Container escape, privilege escalation
- **Mitigation**: Non-root containers, resource limits, security contexts
- **Detection**: Runtime security monitoring, file integrity monitoring

4. Data Exfiltration

- **Risk Level**: High
- **Attack Types**: Bulk data downloads, credential abuse
- **Mitigation**: Rate limiting, behavioral analysis, data loss prevention
- **Detection**: Unusual download volumes, off-hours access

5. Man-in-the-Middle Attacks

- **Risk Level**: Low
- **Mitigation**: TLS 1.3, certificate pinning, HSTS
- **Detection**: Certificate transparency monitoring

Security Monitoring and Detection

Real-Time Monitoring:

- **Authentication anomalies**: Failed login attempts, unusual locations
- **Access pattern analysis**: Volume spikes, off-hours access, bulk downloads
- **API behavior monitoring**: Error rates, response times, query complexity
- **Network traffic analysis**: Unusual traffic patterns, potential DDoS

Automated Response Capabilities:

- **Account lockout** for repeated authentication failures
- **Rate limiting** for suspicious access patterns
- **Token revocation** for compromised credentials
- **IP blocking** for malicious traffic sources

Compliance and Regulatory Requirements

Automotive Industry Standards

Data Protection Compliance:

- **GDPR Article 32**: Technical and organizational security measures
- **CCPA**: California Consumer Privacy Act requirements for PII protection
- **SOX**: Sarbanes-Oxley financial data protection requirements
- **Automotive Industry Standards**: Compliance with industry-specific data handling requirements

Technical Compliance Measures:

- **Data minimization**: Partners receive only authorized data subsets
- **Purpose limitation**: Data access restricted to legitimate business purposes
- **Storage limitation**: Automated data retention and deletion policies
- **Integrity and confidentiality**: Encryption and access controls

Audit and Compliance Reporting

Compliance Monitoring:

- **Quarterly access reviews** for all partner accounts
- **Annual security assessments** and penetration testing
- **Continuous compliance monitoring** with automated reporting
- **Regular vulnerability assessments** and remediation tracking

Audit Trail Requirements:

- **Immutable audit logs** with cryptographic integrity verification
- **Complete data lineage** tracking from source to partner consumption
- **Regulatory reporting** capabilities for compliance audits
- **Data subject rights** support for GDPR/CCPA requests

Incident Response and Security Operations

Security Incident Classification

Severity Levels:

- **Critical**: Data breach, system compromise, widespread service disruption
- **High**: Suspected unauthorized access, security control failures
- **Medium**: Anomalous behavior, failed security events
- **Low**: Policy violations, informational security events

Incident Response Procedures:

1. **Detection and Analysis** (0-30 minutes)

   - Automated alert generation and escalation
   - Initial impact assessment and classification
   - Stakeholder notification and team mobilization

2. **Containment and Eradication** (30 minutes - 4 hours)

   - Immediate threat containment measures
   - Evidence preservation and forensic analysis
   - Root cause identification and elimination

3. **Recovery and Lessons Learned** (4-24 hours)

   - System restoration and validation
   - Post-incident review and documentation

  - Security control improvements and updates

Business Continuity and Disaster Recovery

Recovery Time Objectives (RTO):

- **Critical Systems**: 2 hours maximum downtime
- **Partner API Services**: 1 hour maximum downtime
- **Data Processing Pipeline**: 4 hours maximum downtime

Recovery Point Objectives (RPO):

- **Audit Data**: Zero data loss (real-time replication)
- **Partner Configuration**: 15 minutes maximum data loss
- **Operational Data**: 1 hour maximum data loss (aligned with daily updates)

Backup and Recovery Procedures:

- **Automated daily backups** of all partner databases
- **Cross-region backup replication** for disaster recovery
- **Regular recovery testing** and validation procedures
- **Emergency contact procedures** and communication plans

Security Training and Awareness

Team Security Requirements

Development Team Training:

- **Secure coding practices** and vulnerability prevention
- **Container security** and deployment best practices
- **Incident response procedures** and escalation protocols
- **Privacy and data protection** awareness training

Operational Team Training:

- **Security monitoring** and alert response procedures
- **Access management** and partner onboarding security
- **Backup and recovery** procedures and testing
- **Compliance requirements** and reporting procedures

Partner Security Requirements

Partner Onboarding Security:

- **Security assessment** of partner technical capabilities
- **Secure integration** guidance and best practices
- **Credential management** training and procedures
- **Incident reporting** requirements and contact procedures

Ongoing Partner Communication:

- **Security bulletins** for important updates and threats
- **Best practices guidance** for API integration security
- **Quarterly security reviews** and assessments
- **Emergency contact procedures** for security incidents

Security Metrics and KPIs

Technical Security Metrics

Authentication and Access:

- **Authentication success rate**: > 99.9%
- **Failed authentication rate**: < 0.1% of total attempts
- **Token rotation compliance**: 100% within 24-hour window
- **Secure credential provisioning**: 100% of partners use Hendrick-provided credentials

API Security Performance:

- **API response time impact from security controls**: < 50ms
- **Rate limiting false positive rate**: < 0.01%
- **Security header compliance**: 100% of responses
- **Input validation error rate**: Tracked and monitored

Audit and Monitoring:

- **Audit log completeness**: 100% of security events captured
- **Audit storage integrity**: 100% cryptographic verification success
- **Security alert response time**: < 15 minutes average
- **Compliance reporting accuracy**: 100% automated report generation

Security Incident Metrics

Incident Response Performance:

- **Mean Time to Detection (MTTD)**: < 5 minutes for critical incidents
- **Mean Time to Response (MTTR)**: < 30 minutes for critical incidents
- **Incident resolution time**: < 4 hours for high-severity incidents
- **False positive rate**: < 5% for automated security alerts

Security Effectiveness:

- **Data breach incidents**: Zero tolerance target
- **Partner credential compromise rate**: < 0.1% annually
- **Security control effectiveness**: 100% of tests passed
- **Vulnerability remediation time**: < 30 days for high-severity issues

Continuous Security Improvement

Security Testing and Validation

Regular Security Assessments:

- **Monthly vulnerability scans** with automated remediation tracking
- **Quarterly penetration testing** by third-party security firms
- **Annual security architecture reviews** and updates
- **Continuous security monitoring** with threat intelligence integration

Code Security Practices:

- **Static application security testing (SAST)** in CI/CD pipeline
- **Dynamic application security testing (DAST)** for API endpoints
- **Dependency vulnerability scanning** with automated updates
- **Container image security scanning** before deployment

Security Architecture Evolution

Threat Landscape Monitoring:

- **Continuous threat intelligence** monitoring and analysis
- **Industry security trend** analysis and adaptation
- **Regulatory requirement updates** and compliance adjustments
- **Technology security updates** and patch management

Security Control Enhancement:

- **Zero-trust architecture** implementation roadmap
- **Advanced threat detection** and response capabilities
- **Machine learning-based** anomaly detection enhancement
- **Automated security response** and remediation capabilities

Operations & Maintenance

Operations Overview

The Hendrick Data API operations model prioritizes simplicity, reliability, and comprehensive monitoring while maintaining the security and isolation requirements for Confidential data. This section provides runbook-style procedures for day-to-day operations, troubleshooting, and system maintenance.

Monitoring and Observability

Essential Monitoring Strategy

DataDog Integration: Comprehensive monitoring focused on essential metrics without operational complexity

Core Monitoring Areas:

- **Container Health**: Resource usage, uptime, and health check status
- **API Performance**: Response times, error rates, and throughput
- **Authentication Metrics**: Success/failure rates and security events
- **Database Performance**: Query performance, connection health, and storage utilization
- **Data Freshness**: Critical business metric for operational data currency

Health Endpoint Monitoring

Standardized Health Endpoint: `GET /{partner}/v1/health`

Authentication: Unauthenticated for monitoring simplicity (health data is not confidential)

Health Response Structure:

```
{
  "status": "healthy|degraded|unhealthy",
  "timestamp": "2025-07-02T10:30:00Z",
  "data_freshness": {
    "inventory": {
      "last_updated": "2025-07-02T02:30:00Z",
      "data_date": "2025-07-02",
      "records": 1500,
      "status": "current|stale|missing"
    },
    "sales": {
      "last_updated": "2025-07-02T02:35:00Z",
      "data_date": "2025-07-02",
      "records": 856,
      "status": "current"
    },
    "service": null
  },
  "system_health": {
    "database": "healthy|unhealthy",
    "auth": "healthy|unhealthy",
    "container_uptime": "2d 14h 23m",
    "memory_usage_percent": 45,
    "cpu_usage_percent": 23
  }
}
```

DataDog Health Monitoring:

- **Check Frequency**: Every 5 minutes for all partner health endpoints
- **Data Freshness Alerts**: Alert if any data type status shows "stale" (> 24 hours old)
- **System Health Alerts**: Alert if overall status is "unhealthy" or any component shows "unhealthy"
- **Trend Monitoring**: Track record count trends for anomaly detection

Alert Configuration

Critical Alerts (Immediate Response Required):

- Container failures or health check failures
- API endpoints returning 500 errors consistently (> 10% error rate for 5+ minutes)
- Authentication system failures (Auth0 connectivity issues)
- Database connection failures

- Data freshness alerts (data > 24 hours old)

Warning Alerts (Monitor and Investigate):

- High API response times (> 1 second average for 10+ minutes)
- High container resource usage (> 80% CPU or memory for 15+ minutes)
- Authentication failure rate spikes (> 5% failure rate)
- Unusual partner access patterns (volume spikes > 300% of baseline)

Alert Delivery:

- **Email alerts** to operations team for all severity levels
- **Subject line severity prefixes**: [CRITICAL], [WARNING], [INFO]
- **Alert consolidation**: Group similar alerts within 15-minute windows

Monitoring Dashboards

Primary Operations Dashboard:

- **System Overview**: All partner container health status
- **API Performance**: Response times and error rates across all partners
- **Data Freshness**: Last update timestamps and status for all data types
- **Resource Utilization**: CPU, memory, and storage usage trends

Security Dashboard:

- **Authentication Events**: Success/failure rates and geographic patterns
- **Access Patterns**: Partner usage trends and anomaly detection
- **Security Alerts**: Failed authentication attempts and suspicious activity

Partner-Specific Dashboards:

- **Individual Partner Health**: Detailed health metrics per partner
- **Data Processing**: S3 sync status and processing latency for partner data
- **Usage Analytics**: API call patterns and data consumption trends

Deployment Operations

Kamal Deployment Procedures

Standard Deployment Process:

```
# 1. Pre-deployment validation
kamal config validate
kamal health check

# 2. Deploy with zero downtime
kamal deploy

# 3. Post-deployment verification
```

```
kamal status
kamal logs --tail=100
```

Rollback Procedures:

```
# Quick rollback to previous version
kamal rollback

# Verify rollback success
kamal status
kamal health check
```

Deployment Validation:

1. **Health Check Verification**: All partner health endpoints return "healthy"
2. **Authentication Testing**: Verify Auth0 connectivity and JWT validation
3. **API Functionality**: Test sample API calls for each data type
4. **DataDog Integration**: Confirm metrics and alerts are functioning
5. **Partner Notification**: Inform partners of maintenance window completion

Container Management

Container Lifecycle Operations:

Restart Individual Partner Container:

```
# Restart specific partner container
kamal app restart partner-{partner-name}

# Verify restart success
kamal status partner-{partner-name}
kamal logs partner-{partner-name} --tail=50
```

Scale Container Resources (if needed):

```
# Update container resource limits
kamal config set partner-{partner-name} --memory=2gb --cpu=1000m

# Apply changes
kamal deploy partner-{partner-name}
```

Container Log Access:

```
# View recent logs
kamal logs partner-{partner-name} --tail=100

# Follow live logs
kamal logs partner-{partner-name} --follow

# Search logs for specific events
kamal logs partner-{partner-name} --grep="ERROR"
```

Partner Operations

Partner Onboarding Procedures

Partner Onboarding Checklist *(To be developed during Phase 3)*:

1. **Business Authorization**
   - [ ] Partner agreement and data access authorization
   - [ ] Dealership scope and data type permissions defined
   - [ ] Security and compliance requirements reviewed
2. **Technical Setup**
   - [ ] Partner configuration file created
   - [ ] Auth0 client credentials provisioned
   - [ ] Container deployment and testing
   - [ ] Health endpoint validation
3. **Integration Testing**
   - [ ] Authentication flow verification
   - [ ] API endpoint testing with partner credentials
   - [ ] Data access scope validation
   - [ ] Performance and security testing
4. **Production Deployment**
   - [ ] Production container deployment
   - [ ] DataDog monitoring configuration
   - [ ] Partner credentials delivered securely
   - [ ] Go-live confirmation and monitoring
5. **Documentation and Training**
   - [ ] Partner integration documentation provided
   - [ ] Technical contact information exchanged
   - [ ] Incident escalation procedures established
   - [ ] Quarterly review schedule established

Partner Credential Management

Credential Provisioning Process:

1. **Generate Secure Credentials**:
    - Create Auth0 client credentials
    - Generate unique partner identifier
    - Configure JWT token settings (expiration, scope)
2. **Secure Delivery**:
    - Use 1Password secure sharing for credential delivery
    - Provide integration documentation and sample code
    - Establish secure communication channels
3. **Credential Rotation**:
    - **Scheduled Rotation**: Every 90 days
    - **Emergency Rotation**: Immediate rotation for security incidents
    - **Partner Notification**: 7-day advance notice for scheduled rotations

Credential Rotation Procedure:

```
# Generate new credentials in Auth0
# Update partner configuration with new credentials
# Deploy updated configuration
kamal deploy partner-{partner-name}

# Notify partner of credential change
# Verify successful authentication with new credentials
# Deactivate old credentials after confirmation
```

Partner Offboarding Procedures

Partner Offboarding Checklist:

1. **Access Revocation**
    - [ ] Revoke Auth0 client credentials immediately
    - [ ] Remove partner container from production
    - [ ] Verify API access termination
2. **Data Cleanup**
    - [ ] Export partner data if required for business continuity
    - [ ] Securely delete partner database and audit logs per retention policy
    - [ ] Update DataDog monitoring configuration
3. **Documentation Update**
    - [ ] Update partner access matrix documentation
    - [ ] Archive partner integration documentation

  - [ ] Update security and compliance records

Data Pipeline Operations

Data Processing Monitoring

S3 Data Pipeline Health:

- **Daily Processing Windows**: Monitor expected data arrival times
- **Processing Success Rates**: Track successful XML/CSD data processing
- **Data Quality Validation**: Monitor record counts and data integrity
- **Pipeline Latency**: Track time from S3 arrival to partner database availability

Data Processing Troubleshooting *(Procedures to be developed during implementation)*:

Scenario: Daily Data Not Arriving:

1. **Check S3 Data Availability**:
   - Verify new data files in S3 bucket
   - Check S3 event notifications are functioning
   - Validate file naming conventions and structure
2. **Check Messaging Pipeline**:
   - Verify SQS/SNS message delivery
   - Check message queue health and backlogs
   - Validate message format and content
3. **Check Container Processing**:
   - Review container logs for processing errors
   - Verify database connectivity and health
   - Check partner authorization and data filtering

Scenario: Stale Data Alerts:

1. **Identify Scope of Issue**:
   - Check if issue affects single partner or all partners
   - Identify which data types are affected
   - Determine last successful processing time
2. **Investigate Root Cause**:
   - Review S3 data availability and timing
   - Check for processing errors in container logs
   - Verify database update completion
3. **Resolution Actions**:
   - Trigger manual data processing if needed
   - Restart affected containers if processing is stuck

    - Coordinate with Reynolds & Reynolds if source data issues

Database Operations

Database Health Monitoring:

- **Connection Pool Health**: Monitor database connection availability
- **Query Performance**: Track query response times and slow queries
- **Storage Utilization**: Monitor database file sizes and growth trends
- **Backup Verification**: Validate daily backup completion and integrity

Database Maintenance Procedures:

Daily Database Health Check:

```
# Check database connectivity for all partners
for partner in partners_list; do
    kamal exec $partner "sqlite3 /app/data.db '.databases'"
done

# Verify backup completion
check_backup_status --date=$(date +%Y-%m-%d)

# Review database sizes and growth
monitor_database_sizes --alert-threshold=10GB
```

Database Migration Procedures:

```
# Run database migrations for specific partner
kamal exec partner-{partner-name} "/app/scripts/run-migrations.sh"

# Verify migration success
kamal exec partner-{partner-name} "/app/scripts/verify-schema.sh"

# Update migration tracking
update_migration_status partner-{partner-name} --version=latest
```

Database Backup and Recovery:

Backup Procedures:

- **Automated Daily Backups**: Included in container volume backups
- **Cross-Region Replication**: S3 source data serves as authoritative backup
- **Backup Verification**: Daily verification of backup integrity and restorability

Recovery Procedures:

1. **Complete System Recovery**:

```
# Redeploy entire system
kamal deploy --force

# Reload data from S3 for date range
reload_partner_data --partner=all --date-range="2025-07-01:2025-07-02"
```

2. **Single Partner Recovery**:

```
# Redeploy partner container
kamal deploy partner-{partner-name} --force

# Reload partner-specific data
reload_partner_data --partner={partner-name} --date-range="2025-07-01:2025-07-02"
```

3. **Partial Data Recovery**:

```
# Reload specific data types for date range
reload_partner_data --partner={partner-name} --data-type=inventory --date="2025-07-02"
```

Security Operations

Security Incident Response

Security Incident Classification and Response Times:

Critical Security Incidents (Response within 30 minutes):

- Suspected data breach or unauthorized data access
- Partner credential compromise
- System compromise or container security breach
- Authentication system failures affecting multiple partners

High Priority Security Incidents (Response within 2 hours):

- Suspicious partner access patterns
- Failed authentication rate spikes
- Individual partner credential issues
- Security control failures

Medium Priority Security Incidents (Response within 24 hours):

- Policy violations
- Audit log anomalies
- Non-critical security alerts

Security Incident Response Procedures:

1. **Incident Detection and Initial Response** (0-30 minutes):

   ```
   # Immediate containment actions
   # Revoke partner credentials if compromise suspected
   revoke_partner_credentials {partner-id}

   # Block suspicious IP addresses
   block_ip_address {suspicious-ip}

   # Preserve evidence
   capture_audit_logs --start-time={incident-time} --duration=4h
   ```

2. **Investigation and Analysis** (30 minutes - 4 hours):

   - Review DataDog security alerts and patterns
   - Analyze audit logs and access patterns
   - Coordinate with Auth0 for authentication analysis
   - Document findings and evidence

3. **Resolution and Recovery** (4-24 hours):

   - Implement fixes based on root cause analysis
   - Restore normal operations with enhanced monitoring
   - Update security controls and procedures as needed
   - Conduct post-incident review and documentation

Access Review Procedures

Quarterly Partner Access Reviews:

1. **Review Partner Authorization Matrix**:

   - Verify partner dealership access scope
   - Validate data type permissions
   - Confirm business justification for access

2. **Audit Partner Activity**:

   - Review partner usage patterns and volumes
   - Identify unusual access patterns or anomalies
   - Validate compliance with partner agreements

3. **Update Access Controls**:

   - Revoke access for terminated partnerships
   - Update permissions for changed business relationships
   - Document all access changes and justifications

Performance and Capacity Management

Performance Monitoring

Key Performance Indicators:

- **API Response Time**: Target < 500ms for 95% of requests
- **System Uptime**: Target > 99.5% availability
- **Data Processing Latency**: Target < 2 hours from S3 to partner availability
- **Container Resource Utilization**: Target < 80% CPU and memory usage

Performance Optimization Procedures:

Database Performance Tuning:

```
# Analyze slow queries
analyze_slow_queries --partner={partner-name} --threshold=1000ms

# Update database indexes based on query patterns
optimize_database_indexes --partner={partner-name}

# Monitor query performance improvements
monitor_query_performance --baseline-period=7d
```

Container Resource Optimization:

```
# Monitor resource usage trends
monitor_container_resources --period=30d --alert-threshold=80%

# Scale container resources if needed
scale_container_resources --partner={partner-name} --memory=4GB --
cpu=2000m

# Validate performance improvements
validate_performance_improvement --partner={partner-name} --duration=24h
```

Capacity Planning

Growth Monitoring:

- **Partner Growth**: Track number of active partners and onboarding rate
- **Data Volume Growth**: Monitor data volume trends per partner and overall
- **Resource Usage Trends**: Track CPU, memory, and storage utilization patterns
- **API Usage Growth**: Monitor API call volumes and patterns

Capacity Planning Process:

1. **Monthly Resource Review**:

   - Analyze resource utilization trends
   - Project capacity needs for next 6 months
   - Identify potential bottlenecks or constraints

2. **Quarterly Capacity Assessment**:

   - Review partner growth projections
   - Assess infrastructure scaling requirements
   - Plan for technology migrations (SQLite to PostgreSQL)

3. **Annual Architecture Review**:

   - Evaluate overall system architecture scalability
   - Plan for major infrastructure improvements
   - Update disaster recovery and business continuity plans

Disaster Recovery and Business Continuity

Recovery Objectives

Recovery Time Objectives (RTO):

- **Critical Systems**: 2 hours maximum downtime
- **Partner API Services**: 1 hour maximum downtime
- **Data Processing Pipeline**: 4 hours maximum downtime

Recovery Point Objectives (RPO):

- **Audit Data**: Zero data loss (real-time replication)
- **Partner Configuration**: 15 minutes maximum data loss
- **Operational Data**: 1 hour maximum data loss (aligned with daily updates)

Disaster Recovery Procedures

Complete System Failure Recovery:

1. **Assessment and Communication** (0-30 minutes):

   - Assess scope and impact of system failure
   - Activate incident response team
   - Communicate with partners about service disruption

2. **Infrastructure Recovery** (30 minutes - 2 hours):

   ```
   # Deploy system to new infrastructure
   kamal deploy --force --new-environment

   # Verify base infrastructure health
   kamal status --all-services
   ```

3. **Data Recovery** (2-4 hours):

   ```
   # Reload all partner data from S3 for recent period
   reload_all_partner_data --date-range="last-7-days"

   # Verify data integrity and completeness
   verify_data_integrity --all-partners
   ```

4. **Service Restoration** (4-6 hours):
   - Validate all partner health endpoints
   - Test authentication and API functionality
   - Restore DataDog monitoring and alerting
   - Communicate service restoration to partners

Single Partner Recovery:

```
# Assess partner-specific issue
assess_partner_issue --partner={partner-name}

# Redeploy partner container
kamal deploy partner-{partner-name} --force

# Reload partner data if needed
reload_partner_data --partner={partner-name} --date-range="last-3-days"

# Verify partner service restoration
test_partner_api --partner={partner-name} --full-test-suite
```

Maintenance Windows and Updates

Planned Maintenance Procedures

Monthly System Updates:

- **Security Updates**: Apply container image security updates
- **Dependency Updates**: Update application dependencies and libraries
- **Configuration Updates**: Apply configuration changes and optimizations

Maintenance Window Schedule:

- **Frequency**: Second Sunday of each month, 2:00 AM - 6:00 AM EST
- **Partner Notification**: 7-day advance notice
- **Rollback Plan**: Immediate rollback capability if issues occur

Maintenance Procedure Checklist:

1. **Pre-Maintenance Preparation**:

  - [ ] Partner notification sent (7 days prior)
  - [ ] Backup verification completed
  - [ ] Rollback procedures tested and verified
  - [ ] Change documentation prepared

2. **Maintenance Window Execution**:

  - [ ] System health baseline captured
  - [ ] Updates applied using Kamal deployment
  - [ ] Post-update health verification
  - [ ] Performance validation completed

3. **Post-Maintenance Validation**:

  - [ ] All partner health endpoints healthy
  - [ ] API functionality testing completed
  - [ ] DataDog monitoring restored
  - [ ] Partner notification of completion sent

Emergency Maintenance Procedures

Critical Security Updates (Outside normal maintenance windows):

1. **Immediate Assessment**:

  - Evaluate criticality and risk
  - Determine if emergency maintenance is required
  - Prepare rollback procedures

2. **Emergency Deployment**:

```
# Deploy critical security update
kamal deploy --emergency --security-update

# Immediate health verification
kamal health check --all-partners
```

3. **Post-Emergency Validation**:

  - Verify security update effectiveness
  - Monitor system stability for 24 hours
  - Document emergency change and lessons learned

Documentation and Knowledge Management

Operational Documentation Repository

Documentation Structure:

```
operations-wiki/
├── runbooks/
│   ├── partner-onboarding.md
│   ├── incident-response.md
│   ├── data-pipeline-troubleshooting.md
│   └── security-procedures.md
├── procedures/
│   ├── deployment-procedures.md
│   ├── backup-recovery.md
│   ├── monitoring-setup.md
│   └── capacity-planning.md
├── troubleshooting/
│   ├── common-issues.md
│   ├── error-codes.md
│   ├── diagnostic-commands.md
│   └── escalation-procedures.md
└── reference/
    ├── architecture-diagrams.md
    ├── configuration-reference.md
    ├── api-endpoints.md
    └── security-procedures.md
```

Documentation Maintenance:

- **Update Frequency**: Monthly review and updates
- **Version Control**: All documentation versioned in Git
- **Review Process**: Quarterly documentation accuracy review
- **Access Control**: Team access with appropriate permissions

Training and Knowledge Transfer

Operations Team Training Requirements:

- **System Architecture**: Understanding of partner isolation and data flow
- **Monitoring and Alerting**: DataDog dashboard usage and alert response
- **Incident Response**: Security incident procedures and escalation
- **Partner Management**: Onboarding, offboarding, and support procedures

Knowledge Transfer Procedures:

- **New Team Member Onboarding**: Structured training program with hands-on exercises
- **Cross-Training**: Multiple team members trained on critical procedures
- **Documentation Updates**: Continuous improvement based on operational experience
- **Regular Training Reviews**: Quarterly skills assessment and training updates

Continuous Improvement

Operational Metrics and KPIs

System Reliability Metrics:

- **Uptime**: Monthly system availability percentage
- **MTTR**: Mean time to resolution for incidents
- **MTBF**: Mean time between failures
- **Change Success Rate**: Percentage of successful deployments

Process Efficiency Metrics:

- **Partner Onboarding Time**: Average time from request to production
- **Incident Response Time**: Average response time by severity level
- **Documentation Accuracy**: Quarterly documentation review scores
- **Training Effectiveness**: Team competency assessment results

Monthly Operations Review

Review Process:

1. **Metrics Analysis**: Review operational metrics and trends
2. **Incident Review**: Analyze incidents and identify improvement opportunities
3. **Process Evaluation**: Assess effectiveness of operational procedures
4. **Improvement Planning**: Identify and prioritize operational improvements

Continuous Improvement Actions:

- **Process Optimization**: Streamline procedures based on operational experience
- **Automation Enhancement**: Identify manual tasks for automation
- **Documentation Updates**: Update procedures based on lessons learned
- **Training Updates**: Enhance training based on operational challenges

This comprehensive Operations & Maintenance framework provides the foundation for reliable, secure, and efficient operation of the Hendrick Data API while supporting continuous improvement and growth.

Document created: July 2, 2025 Last updated: July 2, 2025

Document Status

Current Completion Status:

- ✅ **Executive Summary**: Complete
- ✅ **Organization Overviews**: Complete
- ✅ **Business Requirements**: Complete
- ✅ **System Architecture & Design**: Complete
- ✅ **API Specification**: Complete
- ✅ **Data Architecture**: Complete
- ✅ **Technical Implementation Plan**: Complete
- ✅ **Security & Compliance**: Complete
- ✅ **Operations & Maintenance**: Complete

Key Architectural Decisions Made:

- Multi-tenant isolated partner containers with dedicated databases

- Fan-out messaging pattern for S3 data availability notifications
- Docker Compose + Traefik for container orchestration and routing
- Kamal deployment for zero-downtime production deployments
- Database per partner (SQLite as starting option, PostgreSQL for scalability)
- Path-based API routing: `api.hendrick.com/partner/v1/endpoint`
- Auth0 JWT authentication with 1Password secrets management
- DataDog monitoring and observability from production deployment
- Cloud-agnostic design with multi-provider testing capability
- Claude-assisted development for accelerated timelines (5-9 weeks total)

Items Requiring Further Definition:

- Historical data loading strategy for new partner onboarding (to be determined during implementation)
- Detailed message schema for S3 notifications (to be determined during Phase 2 AWS integration)
- Specific data schemas for inventory, sales, and service records (to be determined based on XML/CSD analysis)
- Sample client integration code and documentation (API documentation will be auto-generated using FastAPI built-in tools)