Design Document for Distributor Data Extraction Ingestion - VBD

Nathan Lunceford

2025-03-05

Table of contents

1 Core Business Use Cases

1.1 Extract Data from ERP Systems via API

The system must be able to connect to various ERP systems (e.g., Epicor, Sage) via their APIs, authenticate, and extract required data types. This includes:

- Authenticating with proper credentials (API keys, tokens, client certificates)
- Building appropriate requests for each ERP system
- Handling pagination and batching for large datasets
- Managing rate limits and connection timeouts
- Capturing and processing response data
- Implementing resilience patterns for API communication

1.2 Extract Data from ERP Databases

For ERP systems that allow or require direct database access, the system must:

- Connect securely to various database types
- Execute appropriate queries for different data types
- Handle connection pooling and resource management
- Apply appropriate filtering for incremental extraction
- Process result sets efficiently
- Release database resources properly

1.3 Transform Extracted Data

All extracted data must undergo initial transformation to:

- Standardize column names according to our global data dictionary
- Validate data against defined schemas and quality rules

- Convert to Parquet format for efficient storage and processing
- Apply optional compression for reduced storage needs
- Identify and mask sensitive information when required
- Preserve data lineage information

1.4 Secure Credential Management

The system must securely manage credentials for various ERP systems:

- Retrieve credentials from HashiCorp Vault using least privilege access
- Support different credential types (API keys, database credentials, certificates)
- Handle credential rotation and expiration
- Ensure credentials are never logged or persisted outside secure storage
- Apply proper authentication mechanisms for each ERP type

1.5 Track Data Lineage

For audit and compliance purposes, the system must:

- Track the source, extraction time, and parameters of each data extraction
- Record transformation details and any quality issues
- Link extracted data to its final storage location
- Provide metrics on extraction volume and timing
- Enable traceability for data governance

1.6 Monitor and Report System Health

The system must provide comprehensive monitoring:

- Report on successful and failed extractions
- Expose metrics for performance and throughput
- Alert on critical failures or data quality issues
- Log detailed information for troubleshooting
- Provide health check endpoints for operational status

2 Workflow Documentation

2.1 API Data Extraction Workflow

1. **Initiation**:
    - System receives extraction command with parameters (ERP type, client ID, data type)
    - Validate parameters and check authorization
2. **Preparation**:
    - Retrieve ERP configuration from FerretDB
    - Obtain appropriate credentials from HashiCorp Vault
    - Determine extraction mode (full or incremental)
    - Initialize data lineage tracking
3. **Request Construction**:
    - Build appropriate API request with authentication
    - Apply ERP-specific headers and parameters
    - Configure timeouts and retry settings
4. **Execution**:
    - Send request to ERP API endpoint
    - Handle pagination for large datasets
    - Apply circuit breaker pattern for resilience
    - Process response data into standardized format
5. **Transformation**:
    - Map column names to standard dictionary
    - Validate data against quality rules
    - Mask sensitive information if required
    - Convert to Parquet format
6. **Storage**:
    - Upload transformed data to pre-dropzone S3 bucket
    - Organize by client, ERP type, and data type
    - Apply appropriate permissions
7. **Completion**:
    - Update data lineage with completion status
    - Log extraction metrics and results
    - Report success or failure

2.2 Database Data Extraction Workflow

1. **Initiation**:
    - System receives extraction command with parameters
    - Validate parameters and check authorization
2. **Preparation**:
    - Retrieve ERP database configuration
    - Obtain database credentials from Vault
    - Determine extraction mode (full or incremental)
    - Initialize data lineage tracking
3. **Query Construction**:
    - Build appropriate SQL query for the data type
    - Apply filters for incremental extraction if needed
    - Set appropriate query timeout and batch size
4. **Execution**:
    - Establish database connection with proper isolation
    - Execute query with bulkhead protection
    - Stream results to minimize memory usage
    - Map database results to standardized format
5. **Transformation**:
    - (Same as API workflow)
6. **Storage**:
    - (Same as API workflow)
7. **Completion**:
    - (Same as API workflow)

3 IDesign Layer Components

3.1 Clients Layer

The clients layer serves as the entry point for the extraction process, receiving commands and initiating the workflow.

3.1.1 Components

1. **Program Class**:
    - Parses command-line arguments
    - Configures dependency injection
    - Initializes and starts the host
2. **ExtractionHostedService**:
    - Serves as the main entry point for the extraction process
    - Delegates to appropriate managers
    - Handles top-level error reporting and exit codes

3.2 Managers Layer

The managers layer orchestrates the workflows, coordinating between different components without containing business logic.

3.2.1 Components

1. **ExtractionManager**:
    - Coordinates the overall extraction process
    - Delegates to other managers and engines
    - Tracks progress and ensures completion
    - Handles exceptions and ensures proper cleanup
2. **SecurityManager**:
    - Manages credential retrieval and security concerns
    - Ensures least privilege access to resources
    - Handles rotation and expiration of credentials
3. **ConfigurationManager**:
    - Retrieves and manages ERP configuration
    - Determines correct extraction mode and settings
    - Configures feature flags and system behavior

3.3 Engines Layer

The engines layer contains the core business logic, implementing the rules and processes that define the system's behavior.

3.3.1 Components

1. **ExtractorEngine**:
   - Implements data extraction logic for different sources
   - Builds appropriate requests or queries
   - Processes extracted data into standardized format
   - Applies ERP-specific extraction rules
2. **TransformerEngine**:
   - Standardizes column names per data dictionary
   - Applies data type conversions
   - Implements transformation rules
   - Generates Parquet output
3. **ValidationEngine**:
   - Validates data against schemas and rules
   - Identifies quality issues
   - Reports validation results
   - Enforces data governance policies
4. **LineageEngine**:
   - Tracks data provenance and transformations
   - Records extraction parameters and results
   - Maintains audit trail for compliance
   - Links extracted data to downstream processes

3.4 Accessors Layer

The accessors layer handles all interactions with external systems and resources.

3.4.1 Components

1. **VaultAccessor**:
   - Interacts with HashiCorp Vault
   - Retrieves and manages secrets
   - Applies caching and resilience patterns
   - Ensures secure credential handling
2. **FerretDBAccessor**:
   - Connects to FerretDB (MongoDB protocol)

- Retrieves configuration documents
- Manages database connections efficiently
- Applies retry and circuit breaker patterns

3. **ERPApiAccessor**:

   - Handles HTTP communication with ERP APIs
   - Applies authentication and security measures
   - Manages request/response lifecycle
   - Implements resilience patterns for API calls

4. **ERPDatabaseAccessor**:

   - Connects to ERP databases
   - Executes queries and processes results
   - Manages database resources efficiently
   - Applies connection pooling and isolation

5. **S3Accessor**:

   - Uploads data to S3 storage
   - Manages bucket permissions and organization
   - Ensures data integrity during transfer
   - Handles S3 API interactions

3.5 Resources Layer

The resources layer represents external systems and dependencies that our system interacts with.

3.5.1 Components

1. **HashiCorp Vault**:

   - Secure storage for credentials and secrets
   - Provides short-lived, least-privilege access tokens

2. **FerretDB**:

   - Configuration storage using MongoDB protocol
   - Stores ERP-specific settings and parameters

3. **ERP APIs**:

   - External API endpoints for various ERP systems
   - Provides data access via HTTP/HTTPS

4. **ERP Databases**:
    - Direct database connections to ERP systems
    - Provides data access via SQL queries
5. **S3 Storage**:
    - Pre-dropzone bucket for extracted data
    - Destination for transformed Parquet files

4 Implementation Plan

4.1 Phase 1: Core Extraction Capability

1. **Setup Project Structure**:
    - Create solution with projects for each layer
    - Set up dependency injection framework
    - Implement basic interfaces and models
2. **Implement Initial Vertical Slice**:
    - Focus on one ERP type (Epicor API)
    - Implement minimal components in each layer
    - Create end-to-end workflow for basic extraction
3. **Add Basic Validation and Transformation**:
    - Implement column name standardization
    - Add simple data quality checks
    - Convert to Parquet format

4.2 Phase 2: Expand ERP Support

1. **Add Database Extraction**:
    - Implement database accessor
    - Add query building functionality
    - Support direct database extraction
2. **Add Additional ERP Types**:
    - Implement Sage connector
    - Support for other common ERP systems
    - Create factory functions for dynamic resolution

3. **Enhance Transformation Capabilities**:
    - Implement more advanced data validation
    - Add field-level masking for sensitive data
    - Support incremental extraction with CDC

4.3 Phase 3: Resilience and Monitoring

1. **Implement Resilience Patterns**:
    - Add circuit breakers for external dependencies
    - Implement retry with exponential backoff
    - Add bulkhead isolation for critical operations
2. **Add Monitoring and Metrics**:
    - Implement OpenTelemetry integration
    - Add performance metrics collection
    - Create health check endpoints
3. **Enhance Error Handling**:
    - Implement comprehensive exception handling
    - Add detailed logging for troubleshooting
    - Create structured error reporting

4.4 Phase 4: Security and Compliance

1. **Enhance Security Features**:
    - Implement mutual TLS for applicable ERPs
    - Add field-level encryption
    - Enhance credential management
2. **Complete Data Lineage**:
    - Implement comprehensive lineage tracking
    - Add audit logging for compliance
    - Create lineage visualization tools
3. **Documentation and Training**:
    - Finalize system documentation
    - Create operator guides and runbooks
    - Conduct training sessions for support teams

5 Detailed Interface Specifications

5.1 IExtractionManager

```
/// <summary>
/// Manages the extraction workflow for ERP data
/// </summary>
public interface IExtractionManager
{
    /// <summary>
    /// Extracts data from the specified ERP system
    /// </summary>
    /// <param name="erpType">The type of ERP system</param>
    /// <param name="clientId">The client identifier</param>
    /// <param name="dataType">The type of data to extract</param>
    /// <param name="cancellationToken">Cancellation token</param>
    /// <returns>A task representing the extraction operation</returns>
    Task ExtractDataAsync(
        string erpType,
        string clientId,
        string dataType,
        CancellationToken cancellationToken);
}
```

5.2 ISecurityManager

```
/// <summary>
/// Manages security concerns including credential retrieval
/// </summary>
public interface ISecurityManager
{
    /// <summary>
    /// Gets credentials for the specified ERP system
    /// </summary>
    /// <param name="erpType">The type of ERP system</param>
    /// <param name="clientId">The client identifier</param>
    /// <param name="dataType">The type of data to extract</param>
    /// <returns>Credentials for the specified ERP system</returns>
    Task<Credentials> GetCredentialsAsync(
```

```
        string erpType,
        string clientId,
        string dataType);
}
```

5.3 IExtractorEngine

```
/// <summary>
/// Implements core extraction logic for different ERP systems
/// </summary>
public interface IExtractorEngine
{
    /// <summary>
    /// Extracts data from the specified ERP system
    /// </summary>
    /// <param name="erpType">The type of ERP system</param>
    /// <param name="config">Extraction configuration</param>
    /// <param name="credentials">ERP credentials</param>
    /// <param name="cancellationToken">Cancellation token</param>
    /// <returns>The extracted data</returns>
    Task<ExtractedData> ExtractAsync(
        string erpType,
        ExtractorConfig config,
        Credentials credentials,
        CancellationToken cancellationToken);
}
```

6 Conclusion

This IDesign architecture provides a clear, maintainable structure for our Distributor Data Extraction System. By following the principles of separation of concerns and single responsibility, we've created a design that can accommodate various ERP systems while remaining flexible for future enhancements.

The layered approach ensures that:

1. Each component has a clear, focused responsibility
2. Dependencies flow in a single direction
3. Components can be tested and replaced independently
4. The system can be extended without major refactoring

Implementation should proceed in phases, starting with a minimal vertical slice and gradually expanding to cover all required functionality. This approach will allow us to deliver value incrementally while maintaining a consistent architectural vision.