Extraction Manager Design

Nathan Lunceford

2025-03-06

Table of contents

Overview

The ExtractionManager will serve as the central orchestrator for the ERP data extraction process in our system. It will coordinate the complete extraction workflow from initiation to completion, delegating specific tasks to appropriate engines and accessors.

Core Responsibilities

The ExtractionManager will:

1. Coordinate the extraction workflow for both API and database sources
2. Delegate to specialized engines for business logic processing
3. Interact with accessors for all external resource interactions
4. Handle error scenarios and provide recovery mechanisms
5. Track lineage of all extracted data

Interface Operations

The ExtractionManager will provide these primary operations:

1. **ExtractViaApi**: Extract data from ERP systems through their API interfaces
2. **ExtractViaDatabase**: Extract data through direct database connections
3. **GetExtractionStatus**: Provide status information about a specific extraction
4. **RetryExtraction**: Attempt to re-run a previously failed extraction
5. **GetRecentExtractions**: List recent extraction operations with their status

Workflow Orchestration

API Extraction Workflow

When extracting data via API, the ExtractionManager will:

1. Create a unique extraction ID for tracking and begin lineage recording
2. Request the appropriate configuration from the Configuration Engine
3. Obtain secure credentials from the Vault Accessor
4. Delegate the actual extraction to the Extractor Engine, which interacts with the ERP API
5. Send the extracted data to the Transformer Engine for standardization
6. Pass the transformed data to the Validator Engine for quality checking
7. Instruct the Storage Accessor to store the data in Parquet format
8. Complete the lineage record with processing details and results
9. Return extraction results including status and metadata

Database Extraction Workflow

The database extraction will follow a similar pattern:

1. Create a unique extraction ID and begin lineage recording
2. Obtain database-specific configuration from the Configuration Engine
3. Retrieve secure database credentials from the Vault Accessor
4. Delegate extraction to the Extractor Engine, which will run database queries
5. Process the extracted data through the Transformer Engine
6. Validate the transformed data using the Validator Engine
7. Store the final data using the Storage Accessor
8. Complete the lineage record with all relevant details
9. Return the extraction results

Error Handling Approach

The ExtractionManager will implement consistent error handling:

1. Wrap each core operation in appropriate exception handling
2. Categorize errors by type (configuration, credentials, extraction, etc.)
3. Record detailed error information in the logs and lineage records
4. Return structured error responses with clear error codes and messages
5. Provide context-specific error information to aid troubleshooting

Dependencies

The ExtractionManager will depend on:

Engines

- **Configuration Engine**: For retrieving and processing configuration data
- **Extractor Engine**: For implementing extraction logic (API and database)
- **Transformer Engine**: For standardizing data formats
- **Validator Engine**: For ensuring data quality
- **Lineage Engine**: For tracking data provenance

Accessors

- **Configuration Accessor**: For accessing configuration storage
- **Vault Accessor**: For secure credential retrieval
- **ERP API Accessor**: For communication with ERP APIs
- **ERP Database Accessor**: For database connections
- **Storage Accessor**: For storing extracted data
- **Lineage Accessor**: For persisting lineage information

Utilities

- **Monitoring Utility**: For metrics and observability
- **Security Utility**: For security-related operations

Configuration Options

The ExtractionManager will be configurable with:

- Default storage locations for extracted data
- Performance tuning parameters (batch sizes, timeouts)
- Retry settings and limits
- Logging detail levels

Response Models

The ExtractionManager will use consistent response patterns:

1. **ExtractionResult**: Contains information about a completed extraction
    - Extraction ID, client ID, ERP type, data type
    - Record count and destination path
    - Completion timestamp and warning indicators
2. **ExtractionStatus**: Provides the current status of an extraction
    - Status (in progress, completed, failed)
    - Timing information and record counts
    - Failure details if applicable
3. **Error Information**: Structured error details
    - Error code indicating the error category
    - Human-readable error message
    - Additional context where appropriate

Security Considerations

The ExtractionManager will implement security best practices:

1. Never directly handle or store credentials
2. Use the security context for authorization checking
3. Delegate all credential operations to the Vault Accessor
4. Ensure proper sanitization of logging and error messages
5. Implement auditing of security-relevant operations

Example Usage Scenarios

Basic Data Extraction

The client will call ExtractViaApi or ExtractViaDatabase with: - ERP system identifier - Client ID - Data type to extract - Optional cancellation token

The manager will orchestrate the complete extraction and return the results.

Status Checking

The client will provide an extraction ID, and the manager will: - Retrieve lineage information from the Lineage Accessor - Translate technical details into a user-friendly status - Return comprehensive information about the extraction

Failure Recovery

When an extraction fails, the client can: - Request detailed status information to understand the failure - Call the retry operation with the failed extraction's ID - Receive updated status as the operation is retried

Conclusion

This ExtractionManager design follows the IDesign pattern, focusing solely on orchestrating the extraction workflow. It maintains clear boundaries by delegating business logic to engines and resource access to accessors. This approach provides a maintainable, extensible foundation for handling different ERP systems and extraction methods while ensuring consistent behavior, error handling, and security practices.