**Functional Specification (FS)**

**Project Title:**  
DTN ADMLOD Export Prototype

**Prepared by:**  
Jordan Eisenback

**Date:**  
9/30/25

**Overview**

This specification describes the functional requirements for a C# command-line prototype that simulates the DTN ADMLOD export process. The prototype will generate ADMLOD-compliant export files for Products and Product Groups using mock data, simulate FTP upload, and parse mock DTN response files. All integrations and data sources are mocked.

**Functional Requirements**

**Command-Line Interface**

* The prototype must be executable from the command line
* The primary command is export, which triggers the export process
* Optional arguments:
  + --success or --error to select which mock DTN response file to use
  + --logfile <filename> to specify a log file (optional)

**Example Usage:**

dotnet run export --success

dotnet run export --error --logfile export.log

**Mock Data**

* Mock data must be loaded from external configuration files (e.g., JSON or YAML)
* The system must validate mock data structure and required fields before export
* CLI must support configuration file input for mock data paths and logging option
* Each Product includes:
  + Product Code (PIDXCode)
  + Product Name
  + Product Family
* Each Product Group includes:
  + Product Group Name
  + List of assigned Product Codes

**ADMLOD Export File Generation**

* The prototype generates an ADMLOD export file with the following structure:
  + Header: VERSION;2.4;
  + Export command: EXPORT;PGRP;;Y;
  + One MERGE;PGRP;ProductGroupName; line per product group
  + One MERGE;PRODUCT;PIDXCode;ProductName;ProductFamily; line per product
* The file is saved to a local directory (e.g., MockFiles/adm\_export.txt)

**FTP Upload Simulation**

* The prototype simulates FTP upload by writing the export file to a local directory
* The action is logged as a successful upload
* If the --error flag is used, the prototype simulates an FTP failure by logging an error and aborting the process

**DTN Response Simulation**

* The prototype reads a static mock DTN response file (e.g., adm\_response\_success.txt or adm\_response\_error.txt)
* The response file contains lines indicating success or error for each command
* The prototype parses the response and displays results in the console, distinguishing between successes and errors

**Logging**

* All actions (file generation, upload, response parsing) are logged to the console
* If the --logfile argument is provided, logs are also written to the specified file
* Logging must follow structured standards including severity levels (INFO, WARN, ERROR), timestamps, and correlation IDs for traceability
* All log entries must include a correlation ID for traceability across CLI operations

**Error Handling**

* The prototype must simulate and log the following error scenarios:
  + Missing required fields in mock data (e.g., Product Code is empty)
  + Invalid command format in the export file
  + Simulated FTP failure (if --error flag is used)
  + Errors reported in the mock DTN response file
  + structured error categorization (e.g., validation, FTP, parsing)
  + retry logic for recoverable failures
  + enhanced traceability through correlation IDs
* Retry logic must be implemented for recoverable operations such as FTP upload and response parsing.

**Non-Functional Requirements**

* The prototype must be implemented in C# (.NET 6 or later recommended)
* The code must be organized into logical modules/classes (see Architecture section)
* The prototype must run on Windows and, if possible, cross-platform
* No real FTP or DTN integration is permitted; all integrations are mocked

**Architecture & Module Responsibilities**

| **Module/Class** | **Responsibility** |
| --- | --- |
| Program.cs | Entry point, CLI parsing, main flow |
| MockData.cs | Provides mock Products and Product Groups |
| AdmLodFileGenerator.cs | Generates ADMLOD export file content |
| FtpClientMock.cs | Simulates FTP upload (local file write, error sim) |
| ResponseParser.cs | Parses mock DTN response files |
| Logger.cs | Handles logging to console and file |

**User Stories / Use Cases**

| **As a...** | **I want to...** | **So that...** |
| --- | --- | --- |
| CLI User | Trigger an export operation | I can generate and test ADMLOD files |
| Developer | Simulate FTP upload and DTN response | I can demonstrate the integration flow |
| QA/Tester | Simulate and observe error scenarios | I can verify error handling and logging |
| Stakeholder | Review logs and outputs | I can validate the prototype’s behavior |

**Acceptance Criteria**

* The prototype generates an ADMLOD export file with correct structure and mock data
* The prototype simulates FTP upload and logs the action
* The prototype parses and displays results from a mock DTN response file
* The prototype simulates and logs error scenarios as described
* All actions are visible in the console and, if specified, in a log file
* The prototype is easy to run and review by stakeholders

**Glossary**

* **ADMLOD**: Automated Data Management Load, DTN’s batch file protocol
* **FTP**: File Transfer Protocol, simulated in this prototype
* **Mock Data**: Simulated data used in place of real system data
* **Mock Response**: Simulated DTN response file