**Technical/Architecture Specification**

**Project Title:**  
DTN ADMLOD Export Prototype

**Prepared by:**  
[Your Name/Team]

**Date:**  
[Insert Date]

**Overview**

This document describes the technical architecture and design for a C# command-line prototype that simulates the DTN ADMLOD export process. The prototype generates ADMLOD-compliant export files for Products and Product Groups using mock data, simulates FTP upload, and parses mock DTN response files. All integrations and data sources are mocked.

**Technology Stack**

* **Language:** C#
* **Framework:** .NET 6.0 or later (cross-platform support)
* **Project Type:** Console Application
* **Dependencies:** None required beyond .NET Standard Library
* **File I/O:** Local file system for mock FTP and response files

**High-Level Architecture**

+-------------------+

| Program.cs | <-- Entry point, CLI parsing, main flow

+-------------------+

|

v

+-------------------+ +-------------------+

| AdmLodFileGen |<---->| MockData |

+-------------------+ +-------------------+

|

v

+-------------------+

| FtpClientMock |

+-------------------+

|

v

+-------------------+

| ResponseParser |

+-------------------+

|

v

+-------------------+

| Logger |

+-------------------+

**Module/Class Responsibilities**

| **Module/Class** | **Responsibility** |
| --- | --- |
| **Program.cs** | Entry point, parses CLI arguments, orchestrates the export flow |
| **MockData.cs** | Provides mock Products and Product Groups (hardcoded or loaded from file) |
| **AdmLodFileGen.cs** | Generates ADMLOD export file content from mock data |
| **FtpClientMock.cs** | Simulates FTP upload (writes file locally, logs action, simulates errors) |
| **ResponseParser.cs** | Parses mock DTN response files, outputs results to console/log |
| **Logger.cs** | Handles logging to console and/or file |

**Data Flow**

1. **User triggers export** via CLI.
2. **Program.cs** loads mock data from **MockData.cs**.
3. **AdmLodFileGen.cs** generates an ADMLOD export file (string or file).
4. **FtpClientMock.cs** simulates FTP upload by writing the file to a local directory and logs the action.
5. **Program.cs** selects a mock DTN response file (success or error) based on CLI argument.
6. **ResponseParser.cs** parses the response file and outputs results.
7. **Logger.cs** logs all actions and errors.

**File Structure**

/AdmLodPrototype

Program.cs

MockData.cs

AdmLodFileGen.cs

FtpClientMock.cs

ResponseParser.cs

Logger.cs

/MockFiles

adm\_export.txt

adm\_response\_success.txt

adm\_response\_error.txt

**Mock Data & Files**

* **Products and Product Groups**: Hardcoded in MockData.cs or loaded from /MockFiles/products.json and /MockFiles/productgroups.json.
* **ADMLOD Export File**: Generated as /MockFiles/adm\_export.txt.
* **Mock DTN Response Files**: /MockFiles/adm\_response\_success.txt and /MockFiles/adm\_response\_error.txt.

**Error Simulation**

* **FTP Failure**: If the --error flag is used, FtpClientMock.cs throws an exception and logs an error.
* **Missing Fields**: MockData.cs can include entries with missing required fields to simulate validation errors.
* **Invalid Command Format**: AdmLodFileGen.cs can be configured to generate a malformed line for error testing.
* **DTN Response Errors**: adm\_response\_error.txt includes lines with error messages for parsing.

**Logging**

* All actions and errors are logged to the console.
* If the --logfile argument is provided, logs are also written to the specified file.
* Log entries include timestamps and severity (INFO, ERROR).

**Extensibility & Maintainability**

* **Modular Design**: Each responsibility is encapsulated in a separate class.
* **Mock Data**: Easily extendable for additional scenarios or future ADMLOD commands.
* **Error Handling**: Centralized in Logger.cs and handled gracefully in each module.
* **Configuration**: File paths and mock data sources can be moved to a config file if needed.

**Security**

* No real credentials or sensitive data are used.
* All file operations are local and do not interact with external systems.

**Build & Run Instructions**

* **Build**:
  + dotnet build
* **Run**:
  + dotnet run export --success
  + dotnet run export --error --logfile export.log

**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.