**Deployment / Runbook – DTN ADMLOD Export Prototype**

**Prepared by: Jordan Eisenback  
Date: October 5, 2025**

**🎯 Purpose**

**This runbook provides step-by-step instructions for building, configuring, running, and troubleshooting the DTN ADMLOD Export Prototype. It supports both success and error scenarios using dynamic data and response generation.**

**🧰 Prerequisites**

* **.NET 6.0 SDK or later installed**[**Download .NET**](https://dotnet.microsoft.com/download)
* **Access to the project source code (local or version control)**
* **Command-line access (Windows, Linux, or Mac)**
* **JSON files for mock data:** 
  + **/MockFiles/products.json**
  + **/MockFiles/productgroups.json**
* **Configuration file:** 
  + **/appsettings.json**

**📁 Directory Structure**

**/AdmLodPrototype**

**Program.cs**

**/Repositories**

**IProductRepository.cs**

**IProductGroupRepository.cs**

**JsonProductRepository.cs**

**JsonGroupRepository.cs**

**/Responses**

**IResponseGenerator.cs**

**MockResponseGenerator.cs**

**/Services**

**AdmLodFileGenerator.cs**

**FtpClientMock.cs**

**ResponseParser.cs**

**/Utilities**

**Logger.cs**

**/MockFiles**

**products.json**

**productgroups.json**

**adm\_export.txt**

**export.log**

**appsettings.json**

**🛠️ Build Instructions**

**Shell**

**dotnet build**

**Show more lines**

**▶️ Run Instructions**

**✅ Success Scenario**

**Shell**

**dotnet run export --success**

**Show more lines**

**❌ Error Scenario**

**Shell**

**dotnet run export --error**

**Show more lines**

**📝 With Logging to File**

**Shell**

**dotnet run export --success --logfile export.log**

**Show more lines**

**⚙️ Expected Behavior**

* **Generates ADMLOD export file in /MockFiles/adm\_export.txt**
* **Simulates FTP upload by writing the file locally**
* **Dynamically generates DTN-like response based on export command validation**
* **Logs all actions and errors to console and optionally to a log file**
* ### Configuration via appsettings.json
* The prototype supports external configuration using `appsettings.json`. This file defines paths, logging options, and feature flags.
* Sample:
* {
* "Paths": {
* "ExportFile": "MockFiles/adm\_export.txt",
* "ResponseSuccessFile": "MockFiles/adm\_response\_success.txt",
* "ResponseErrorFile": "MockFiles/adm\_response\_error.txt",
* "LogFile": "MockFiles/export.log"
* },
* "Logging": {
* "EnableFileLogging": true,
* "LogLevel": "INFO"
* },
* "FeatureFlags": {
* "SimulateFtpFailure": false,
* "EnableDynamicResponse": true
* }
* }
* Notes:
* - CLI flags override configuration values.
* - If `EnableFileLogging` is true, logs are saved to the path in `Paths.LogFile`.
* - `SimulateFtpFailure` can be toggled to test error handling.

**🧪 Troubleshooting**

| **Issue** | **Resolution** |
| --- | --- |
| **Build Errors** | **Ensure .NET 6.0+ is installed and terminal is in the correct directory** |
| **File Not Found** | **Ensure /MockFiles exists and contains required JSON files** |
| **No Output** | **Check CLI arguments and logs for errors** |
| **Unexpected Errors** | **Validate JSON structure and ensure all required fields are present** |

**🧩 Customization**

**🔧 Mock Data**

* **Edit products.json and productgroups.json to simulate different business scenarios.**
* **Use invalid entries to test error handling.**

**🔁 Response Logic**

* **Modify MockResponseGenerator.cs to adjust validation rules or simulate new error types.**

**⚙️ Configuration**

* **Update appsettings.json to change file paths, logging options, or feature flags.**

**🔄 Maintenance**

* **To add new test scenarios:** 
  + **Extend mock data files**
  + **Update response generator logic**
* **To extend functionality:** 
  + **Follow modular design and coding guidelines**
  + **Inject new services via DI container**