OpenDespatch Architecture Documentation

OpenDespatch is a open soure application written in asp.net C#. The source code is available from https://opendespatch.codeplex.com. This document outlines the architectural structure of the project to help developers understand the project should they need to modify or extend the code.

Project Architecture

After you have downloaded OpenDespatch and loaded it into the IDE you will see there are multiple projects loaded. The main project for the OpenDespatch app is the project named “OpenDespatch.WindowsApp” it controls the logic for the graphical user interface of the application. The project named “OpenDespatch.Entities” contains the object definitions for the despatch data that is used in the application. It also contains some factory classes that are responsible loading the plugin projects.

The other projects in the IDE are plugins that are loaded at runtime with late binding / reflection. This allows the flexibility for extension. For example one of these plugins is for Royal Mail (“OpenDespatch.RoyalMail”) and it contains the logic for data that is sent to that specific carrier. To extend OpenDespatch for use with another carrier the idea is to copy the RoyalMail project and adapt it for the new carrier. This new carrier will then also be loaded by the main project.

Anatomy of a carrier plugin for OpenDespatch

1. The carrier plugin will require a reference to the “OpenDespatch.Entities” project.
2. The carrier will extend using inheritance the base despatch class (located in the entities project) so it can implement customisations of this class for the carrier. In the Royal Mail project this is the class that does this is the “DespatchExt.cs” class.
3. “RMCountry.csv” in the Royal Mail plugin is a flat data store for country mappings you can add to this any country names and give it the RM code you want to be fed through without having to re-compile the project.
4. “RMTemplate.txt” a template file for how the data is going to be fed through to Royal Mail, again this can be adjusted without the need for recompilation. The names that are enclosed in curly braces eg "{CustomerName}" are placeholders for the actual data. At runtime the data is inserted into these placeholders as the logic looks for that name as a property on the extended despatch class.
5. “RoyalMailPlugin.cs” contains the logic for the actual sending of the csv data to the correct input folder. It also has logic to wait for a result file and extract the trackingnumber. This tracking number is then set as a property on the base plugin class and is then accessible to the main windows application. The main windows application can then send this tracking number back to PVX via the PVX API.