# TwilTool: Developer ReadMe

## Overview

The purpose of the document is help anyone who plans to perform development on TwilTool a quick understanding of how the application has been constructed.

## Technologies

* .NET Framework 4.7.1
* Visual Studio 2017
* C#

## Patterns and practices

**MVVM** – The MVVM pattern is being used for all the UI components; windows and controls.

**Dependency Injection** – DI is being used for nearly all dependent logic to encourage good separation as well as easy Unit Testing. As a result, most classes that do work are represented by an interface, which everyone who needs it, simply accepts it in the constructor.

**Inversion of Control** – IoC is being used (via Ninject) to automatically inject the dependent services/objects with constructor injection.

**Unit Tests** – some basic unit tests have been included

## NuGet Packages

**NinjaMvvm** – used to implement MVVM pattern by providing some ViewModel base logic like INotifyPropertyChanged Implementation and RelayCommand.

**Ninject** – used to provide IoC

**MahApps.Metro** – Provides the basic look -n- feel/theming

**Twilio** – package for interfacing the Twilio Api

**NLog** – used to provide debug and error logging

## TwilTool.Services.Sms

All the logic for actually sending and receiving messages is contained in the TwilTool.Services.Sms project. All information sent to and received from Twilio is abstracted, so local classes exist to translate the responses received from the Twilio client library.

The purpose of this extra abstraction is that if you decide to swap out twilio for another service, you should be able to simply create a new implementation of ISmsService and everything will simply work without any changes to the View Models.

## TwilioConfiguration

The Twilio Sms implementation requires the consumer to provide an implementation of the ITwilioConfigurationService. This is done in the TwilTool Wpf library, and it pulls its settings from the App.Config.

## TwilTool.Services.PhoneNumberParsing

The logic for splitting the bulk inputted phone numbers is handled in the TwilTool.Services.PhoneNumberParsing Project. Originally, this project would also format the phone numbers and ensure they were valid. However, the validation logic was written to support only U.S. numbers, and since this project requires international numbers, the validation logic was commented out. Note: the Parser is still called, and all code to react to invalid numbers is in place, so one could simple alter the ParseFromText routine to support international numbers.

## UI Theme

The basic theme is provided by MahApps.Metro. You’ll find those resources being included in the Themes\Standardard\StandardTheme.xaml.

## UI Navigation

Opening windows and binding UI controls to View Models is done using a ViewModel driven navigation pattern. This is accomplished with 2 pieces.

1. Navigator.cs – this class provided the INavigation implementation and is responsible for showing dialogs. If the App ever grows, simply provide an implementation of in the NavigateTo method to choose how to get to a new view.
2. DataTemplates – Controls are mapped to View Models using Data Templates found in App.Xaml. This allows you to bind ContentPresenters directly to View Models.

## Log files

NLog is used to write errors and debug information. You configure where those logs go by editing the NLog.config file. At the time of this writing, Logs are configured to be written to %ProgramData%\TwilTool