# MyEmailManager

## Specification:

### Existing Problem to be solve:

Currently (Up to MailKit V1.2.8.0) it is not possible to properly reference MimeKit library into a PCL and because MailKit explicitly depends on MimeKit it is not possible to reference it neither.   
Hence, PCL projects can’t directly reference nor work with any part inside of those libraries causing the inability to work with the library at any imagined (by me) level.

MailKit is the most popular, efficient and up-to-date library. They release a new version almost every month it is easy to use.   
In general, it offers a kit of tools to leverage email features but, it does not offer an object oriented perspective to handle them.

### What we want to achieve?

Implement a way to be able to work with the libraries by solving the specific problems for every platform, that is to offer a library that can be referenced from any type of project and leverage all its email features.

By taking advantage of the above approach we can also design a wrapper to the library that encapsulate all the features together creating an Object Oriented Design leveraging an easy-to-use instantiated objet than can be configured, validated, recovered from errors and reused over all the application life cycle.

Because of the power of the MailKit libraries and the importance of leveraging email features, this class would easily excels the borders of any project and be used as technology for support future email-related needs.

### Features:

1. Life Cycle:
   1. The library shall wrap, encapsulate and manage all the functionalities.
   2. Each instance shall represent one email manager and shall leverage all the features: Settings, states, behaviors and features.
   3. Thread-safe is not a priority at all.
   4. The library shall guarantee that every resource will be safely released at the end of it life cycle.
   5. The library shall be testable (seams everywhere):
2. Settings:
   1. The settings shall be easy to set, to change and to check.
   2. Provide as much default values as possible.
   3. Provide a way to report that a setting has been changed.
   4. Provide a way to reset the settings to the default state.
   5. Provide a state which indicate whether the settings are valid or not.
      * The only way to set a valid state is through a successful validation.
      * At the beginning of its life cycle it state shall be invalid.
      * Every time any setting is changed, the manager will gain an invalid state.
      * None feature shall be available for an invalid state. (But for the Validate feature)
   6. Setting that shall be provided:
      * Smtp host: Required
      * Smtp port: Default = 25
      * Imap host: Default = Smtp host
      * Imap port: Default = 143
      * Use SSL: Default = false
      * User email: Required
      * User password: Default = “”
      * User login: Default = User email
      * Recipient email: Default = User email.
      * Message Subject: Default = “”
      * Filter subject: Not Required.
      * Filter sender: Not Required.
      * Filter unread: Default = false
      * After received options: Default = Do Nothing  
        Possibilities:
        + Do Nothing
        + Mark As Read
        + Remove
      * Idle timeout: Default = 60000 seconds = 1 minute
3. Validations:
   1. The Valid state shall be invalid if any setting is invalid.
   2. The Validation feature shall indicate if the validation was a success or not.
   3. For every invalid setting an error message shall be generated.
   4. Validate User Settings:
      * User email shall be a valid email.
      * User password can’t be null.
      * User login can’t be null.
   5. Validate Sending and Receiving Settings:
      * Recipient email shall be a valid email.
      * Message Subject can’t be null.
   6. Validate SMTP settings:
      * Host should have a value.
      * The port should have a correct value.
      * Perform a connection test and detect possible settings errors:
        + Host invalid | unreachable.
        + Server do not answer.
        + Authentication error.
   7. Validate IMAP settings:
      * Host should have a value.
      * The port should have a correct value.
      * Perform a connection test and detect possible settings errors:
        + Host invalid | unreachable.
        + Server do not answer.
        + Authentication error.
   8. Misc.:
      * Idle timeout should be a valid value.
   9. Validate All settings:
      * Perform a whole test: Send a test message and retrieve it.
4. Send:
   1. The feature can only be done if all the settings are valid.
   2. If there are invalid settings, an error shall be generated.
   3. **It shall catch the relevant exceptions to the settings. Set the settings to invalid and save the error messages**
      * System.IOException:
        + Connect, Authenticate, Send: An I/O error occurred.
      * MailKit.ProtocolException:
        + Connect, Authenticate, Send: A protocol error occurred.
      * MailKit.Security.AuthenticationException:
        + Authenticate: Authentication using the supplied credentials has failed.
      * MailKit.SaslException:
        + Authenticate: A SASL authentication error occurred.
      * MailKit.CommandException:
        + Send: The send command failed.
   4. Send a message:
      * Specify the message body.
      * Specify the recipient or take it from the settings
      * Specify the subject or take it from the settings.
   5. Indicate to the user if the Send was successful.
   6. Do not catch relevant exceptions.
5. Receive:
   1. The feature can only be done if all the settings are valid.
   2. If there are invalid settings, an error shall be generated.
   3. **It shall catch the relevant exceptions to the settings. Set the settings to invalid and save the error messages:**
      * System.IOException:
        + Connecting, Authenticate: An I/O error occurred
      * MailKit.ProtocolException:
        + Connecting: A protocol error occurred.
      * MailKit.ImapProtocolEsception:
        + Authenticate: An IMAP protocol error occurred.
      * MailKit.AuthenticationException:
        + Authenticate: Authentication using the supplied credentials has failed.
      * MailKit.SaslException:
        + Authenticate: A SASL authentication error occurred
   4. Retrieve a list of messages:
      * Filter the messages to retrieve by the filter settings.
      * Mark the retrieved messages as specified in the settings.
6. Automation:
   1. Enable automatic detection of new emails:
      * This feature can only be executed if all the settings are valid.
      * This feature can only be in execution while all the settings are valid.
      * Any attempt to start the feature once it is already started shall be ignored.
   2. Disable the automatic detection of new emails:
      * The feature can be disabled at any time.
      * As soon as any setting is invalidated, the feature shall be immediately disabled.
      * Any attempt to disable the feature once it is already disabled shall be ignored.
7. To report and to recover from exceptional states:
   1. Provide an error message list containing all the manager’s recoverable errors.
   2. **Recover from the most of exceptions.**