MOH OpenPGP Batch Job

Setup guide

Jonathan Keng Hung Aw

2023

Contents

[Introduction 1](#_Toc135834500)

[**1.** The Importance of a Passphrase 1](#_Toc135834501)

[2. Setup Steps for Solution 1: Protect Passphrase using AWS Secrets Manager 3](#_Toc135834502)

[3. Setup Steps for Solution 2: Protect Passphrase using Windows Data Protection API 10](#_Toc135834503)

[4. Setup Steps for Solution 3: Protect Passphrase using ASP.NET Core Data Protection API 14](#_Toc135834504)

# Introduction

The **OpenPgpBatchJob** project implements a fully functional Production-Ready OpenPGP Batch Job (using gnupgme-sharp). This project is written in C# and targets Microsoft .NET Framework 6.0 (LTS). Thanks to .NET 6.0, this Batch Job can be built to run on Windows, Linux and macOS Operating Systems.

## Key Features

1. Ready-to-Use Production-Ready BatchJob that implements OpenPGP in accordance to MOH's OpenPGP Specifications.
2. Thanks to .NET 6.0, this Batch Job can be built to run on Windows, Linux and macOS Operating Systems.
3. This Batch Job supports multiple runtime configurations (eg. Config\_RunAsSender\_for\_SystemA, Config\_RunAsRecipient\_for\_SystemA, etc.). This allows a single instance of this BatchJob to support multiple OpenPGP use cases - eg. as a ***'SenderRole with Partner-System-A'*** or as a ***'RecipientRole with Partner-System-A'***, etc. Each Execution of this Batch Job will be based on 1 chosen Runtime Configuration, specified in an input argument to run the job.
4. This Batch Job is able to process all the files in the source folder, inclusive of files in all the sub-folders therein.
5. This Batch Job is able to perform auto-archival of source files, if an archive folder path is specified in the Runtime Configuration.
6. Logging to Console and to LogFiles.

# Setup your development machine to build the Batch Job for your target OS platform.

1. **Install Visual Studio IDE**

You may install the free **Visual Studio Community Edition** if you do not have a valid Enterprise License for it. <https://visualstudio.microsoft.com/vs/community/>

1. **Install GnuPG on your Development Machine**

|  |
| --- |
| * On Windows, you will need to install [Gpg4Win](https://www.gpg4win.org/). * On Debian and Ubuntu, install the [libgpgme11 package](https://packages.debian.org/stretch/libgpgme11). * On other Linux distros or other operating systems, install libgpgme using your favourite package manager, or compile it from source.   Note that Gpg4Win currently only distributes a 32-bit build, so on Windows you **must** set your C# app to run in 32-bit mode. |

# Cloning the GitHub Project to your Development Machine

Graphical user interface, website

Description automatically generated

Figure 1: Getting the Source Code from Github onto your development machine

# Opening the Solution

Open the Visual Studio Solution file (gpgme-sharp.sln) in the root folder of the cloned project in Visual Studio.

A screenshot of a computer

Description automatically generated

# Generating OpenPGP Keypairs (with Passphrase protection)

Refer to MOH’s OpenPGP Implementation Guide if you are implementing OpenPGP either for MOH or a MOH’s partner. The guide is distributed separately by your MOH IFC Project Manager and not open-sourced in GitHub. Email your MOH IFC Project Manager to obtain a copy of the MOH OpenPGP Implementation Guide if necessary.

# Decide on the method for protecting the confidentiality of the secret passphrase.

This repository provides source code for 3 ready-to-use solutions that protects the confidentiality of the secret passphrases of the OpenPGP private keys.

a. Uses AWS Secrets Manager [Recommended for AWS serverless and containerized based solutions. Also useful for Applications hosted on AWS EC2 Instances.]

b. Uses Windows Data Protection API [Only works for Systems developed for Windows OS. Optimized for Windows-Based Applications!]

c. Uses ASP.NET Core Data Protection API [Works for Windows, Linux and macOS based Applications. Can be used on any .NET core applications, including non-ASP.NET ones. Recommended for all other types of Applications that cannot use Solutions 1 & 2.]

Refer to the respective steps in SETUP.docx to enable each of the option above.

5. Building the BatchJob.

### Building the solution for Windows OS.

Simply Rebuilt the Solution in Visual Studio with the default project configurations. [Target platform for the batch job must be **x86**. Note that Gpg4Win currently only distributes a 32-bit build, so on Windows you must set your C# app to run in 32-bit mode.]

A screenshot of a computer

Description automatically generated

Figure 2: You should be able to rebuild the solution successfully!

a. Option 1: Testing. You may run the sample code in Debug Mode / Without Debug Mode (F5 / Ctrl + F5).

b. Option 2: Production. Execute the console program with an input argument specifying the configuration to use. Eg. OpenPgpBatchJob Config\_RunAsRecipient\_for\_SystemA