MOH OpenPGP Batch Job

Setup guide

Jonathan Keng Hung Aw

2023

Contents

[Introduction 2](#_Toc135920597)

[Key Features 2](#_Toc135920598)

[Options to obtain the Batch job: 2](#_Toc135920599)

[1. Download the Latest Binary Release for Windows 2](#_Toc135920600)

[2. Build the Batch Job from Source Code 3](#_Toc135920601)

[Setup your development machine to build the Batch Job for your target OS platform. 3](#_Toc135920602)

[Cloning the GitHub Project to your Development Machine 3](#_Toc135920603)

[Opening the Solution in Visual Studio 4](#_Toc135920604)

[Building the solution for Windows OS. 4](#_Toc135920605)

[Building the solution for non-Windows OS 5](#_Toc135920606)

[Generating OpenPGP Keypairs (with Passphrase protection) 7](#_Toc135920607)

[Configuring the Batch Job 8](#_Toc135920608)

[The Main Configuration File 8](#_Toc135920609)

[Sample Scenario Configuration Files 8](#_Toc135920610)

[Decide on the method for protecting the confidentiality of the secret passphrase. 9](#_Toc135920611)

[Running the Batch Job 10](#_Toc135920612)

[Running the Batch Job program (for operations). 10](#_Toc135920613)

[Running the Batch Job from Visual Studio (Windows OS). 13](#_Toc135920614)

# Introduction

The MOH OpenPGP Batch Job provides a ready to use option to start implementing OpenPGP in a way that complies with the Secured Coding Rules stated in MOH's OpenPGP Implementation guide without any development efforts.

This project (using gnupgme-sharp) 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 Batch Job that implements OpenPGP in accordance with 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 **scenario configurations** (e.g., Config\_RunAsSender\_for\_SystemA, Config\_RunAsRecipient\_for\_SystemA, etc.). This allows a single instance of this Batch Job to support multiple OpenPGP use cases - e.g., as a 'Sender Role with Partner-System-A' or as a 'Recipient Role with Partner-System-A', etc. Each Execution of the Batch Job will be based on 1 chosen **scenario configuration**, specified in an input argument to run the job.

4. This Batch Job can process all the files in the source folder, inclusive of the files in all the sub-folders therein.

5. This Batch Job can perform auto archival of source files, if an archive folder path is specified in the Runtime Configuration.

6. Logging to Console and to Log Files.

# Options to obtain the Batch job:

1. **Download the Latest Binary Release for Windows:** Download the latest release-built exe of MOH OpenPGP Batch job Program for Windows OS. Choose this if you do not intend to customize the Batch Job.
2. **Build the Batch Job from Source Code:** Choose this if you either intend to customize the Batch Job or need to build the Batch Job for Non-Windows OS.

## Download the Latest Binary Release for Windows

Download the latest release-built exe of **MOH OpenPGP Batch job Program** for Windows OS from <https://github.com/jawkh/gpgme-sharp-implementation-guide/tree/master/OpenPgpBatchJob/Publish/Windows>

You may skip the next section if you are not building the batch job from source code.

## Build the Batch Job from Source Code

### 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.** **This is a key prerequisite to run GnuPG (ie. which provides the OpenPGP capabilities) on your Computer/Server.**

|  |
| --- |
| * On Windows, you will need to install [Gpg4Win](https://www.gpg4win.org/). 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. * 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. |

### 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 in Visual Studio

Open the Visual Studio Solution file (**gpgme-sharp.sln**) in the root folder of the local git repository in Visual Studio. **OpenPgpBatchJob** is the Visual Studio project with source code for the batch job.

A screenshot of a computer

Description automatically generated

### Building the solution for Windows OS.

Simply rebuild the **OpenPgpBatchJob** project or the entire solution in Visual Studio with the default project configurations. [The Gpg4Win OpenPGP program currently only distributes a 32-bit build. Therefore, on Windows you must set your C# app to run in 32-bit mode, i.e., target platform for the batch job must be set to **x86** for windows when building the project.]

A screenshot of a computer

Description automatically generated

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

### Building the solution for non-Windows OS

**IMPORTANT:** The default target platform for this project is set to x86 for Windows. Please select the appropriate platform target for your target Non-Windows OS.

A screenshot of a computer

Description automatically generated

Figure 3: Selecting the appropriate processor for your target Non-Windows OS.

Use **dotnet build** to build the batch job for your target OS.

<https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-build>

Pay attention to the following **dotnet build** arguments:

--os <OS>

Specifies the target operating system (OS). This is a shorthand syntax for setting the [Runtime Identifier (RID)](https://learn.microsoft.com/en-us/dotnet/core/rid-catalog), where the provided value is combined with the default RID. For example, on a win-x64 machine, specifying --os linux sets the RID to linux-x64. If you use this option, don't use the -r|--runtime option. Available since .NET 6.

-r|--runtime <RUNTIME\_IDENTIFIER>

Specifies the target runtime. For a list of Runtime Identifiers (RIDs), see the [RID catalog](https://learn.microsoft.com/en-us/dotnet/core/rid-catalog). If you use this option with .NET 6 SDK, use --self-contained or --no-self-contained also. If not specified, the default is to build for the current OS and architecture.

--self-contained [true|false]

Publishes the .NET runtime with the application so the runtime doesn't need to be installed on the target machine. The default is true if a runtime identifier is specified. Available since .NET 6.

**An Example:**

|  |
| --- |
| Build a project and its dependencies for a specific runtime (in this example, Ubuntu 18.04):  dotnet build --runtime ubuntu.18.04-x64 --self-contained  A screenshot of a computer  Description automatically generated with medium confidence  Figure 4: Building the .Net program for your target OS using dotnet build.    Figure 5: Built is successful.    Figure 6: Self-Contained Built Package inside a folder. Copy this folder to your Server and run the batch job as a standard console program therein the Server. |

# Generating OpenPGP Keypairs (with Passphrase protection)

By now, you would have a copy of Batch Job executable for your target OS. Before we start configuring the Batch Job, you will need some OpenPGP Keypairs (with passphrase protection enabled for them).

Please refer to **MOH’s OpenPGP Implementation Guide** for detailed instructions on generating the OpenPGP Keypairs based on **MOH’s OpenPGP Specifications** andhardening the **GnuPG client** on your server if you are implementing OpenPGP either for MOH or a MOH’s partner. The implementation 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.

# Configuring the Batch Job

Now you are ready to configure the Batch Job to run on your machine!

The Batch Job utilizes 2 types of Configuration Files, namely one **Main Configuration File** and one or more **Scenario Configuration Files**.

## The Main Configuration File

The Main Configuration is **OpenPgpBatchJob.dll.config** *(in your built folder)* and **app.config** *(in the source code)*.

A screenshot of a computer screen

Description automatically generated with medium confidence

Figure 7: OpenPgpBatchJob.dll.config

## Sample Scenario Configuration Files

I provided 2 sample configurations files (asSenderForSystemA.app.config & asRecipientForSystemA.app.config) for reference. Follow the instructions in the comments to configure your Scenario Configuration Files.

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 8: A Sample Scenario Configuration File

To configure the Scenario Configuration File, you need to first decide on the method to protect the confidentiality of secret passphrases of your OpenPGP private keys.

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

The batch job supports 3 options for protecting the confidentiality of the secret passphrases of the OpenPGP private keys.

1. Uses **AWS Secrets Manager** [Recommended for AWS serverless and containerized based solutions. Also useful for Applications hosted on AWS EC2 Instances.]
2. Uses **Windows Data Protection API** [Only works for Systems developed for Windows OS. Optimized for Windows-Based Applications!]
3. 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 - Protect PgP SecretPassphrase.docx](SETUP%20-%20Protect%20PgP%20SecretPassphrase.docx) for detailed step-by-step instructions on how to setup each of the 3 options.

# Running the Batch Job

You are now ready to run the Batch Job.

## Running the Batch Job program (for operations).

Execute the Batch Job as a console program on your target Operating System.

Supply an input argument specifying the Scenario Configuration to use.

E.g., **OpenPgpBatchJob** Config\_RunAsRecipient\_for\_SystemA

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 9: Running the Batch job as a Console program on Windows.

A screenshot of a computer

Description automatically generated

Figure 10: Logs captured in the log file. Path of the Logs is configured in the main Configuration.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11: Example of a Source Folder. All the files in this folder and all its sub folders will be processed by the Batch Job.

A screenshot of a computer

Description automatically generated

Figure 12: Example of an Archive Folder. The Batch Job will automatically move all the successfully processed source files into the specified archive folder. Batch job will create sub folders in the archive folder to match that of the source folder.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 13: Example of a Destination Folder. Batch job will create sub folders in the destination folder to match that of the source folder.

## Running the Batch Job from Visual Studio (Windows OS).

You may run the sample code in Visual Studio in Debug Mode / Without Debug Mode.

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 14: Select 1 of the 2 preconfigured debug profile. Each one is linked to a sample scenario configuration to run the batch job)

A screenshot of a computer

Description automatically generated

Figure 15: Batch Job running in Visual Studio Debug Console.