# Marble Framework

**Description**: The Marble Framework is designed to allow for flexible and easy-to-use obfuscation when developing tools. When signaturing tools, string obfuscation algorithms (especially those that are unique) are often used to link malware to a specific developer or development shop. This framework is intended to help us (AED) to improve upon our current process for string/data obfuscation in our tools. The framework utilizes pre and post-build execution steps to apply obfuscation to the tool. If the tool breaks the build, the post build will always be able to repair it. The pre-build execuion step will store clean copies of the code before making modifications. The post build execution step restores the files to a clean-copy state. The framework allows for obfuscation to be chosen randomly from a pool of techniques. These techniques can be filtered based upon the project needs. If desired, a user may also, select a specific technique to use for obfuscation. A receipt file is generated on run (and replaces any previous receipts). The receipt file identifes the algorithm used as well as all of the strings/data that was obfuscated. The post-build step will also double check to make sure none of the obfuscated data appears in the binary.

The framework's integration into the EDG

Project Wizard will set up the appropriate project and solution properties needed to run. Currently, the obfuscation framework will only be set for release builds. If it is so desired to debug the obfuscated strings you may manually set the pre and post build events.

Core Library Repository

# Framework Terminology

**Marble:** A Marble is a specific algorithm that scrambles and unscrambles data.

**Mibster:** The Mibster is the utility that does the scrambling and altering of source files. The Mibster starts by choosing a Marble (an algorithm) from the filtered list of available algroithms. The Mibster then scans the directories containing source, looking for an strings and data to scramble. The Mibster keeps a clean copy of the original source and replaces it with the scrambled versions of strings/data as well as supplies the unscramble function. The source should compile after Mibster modifies source.

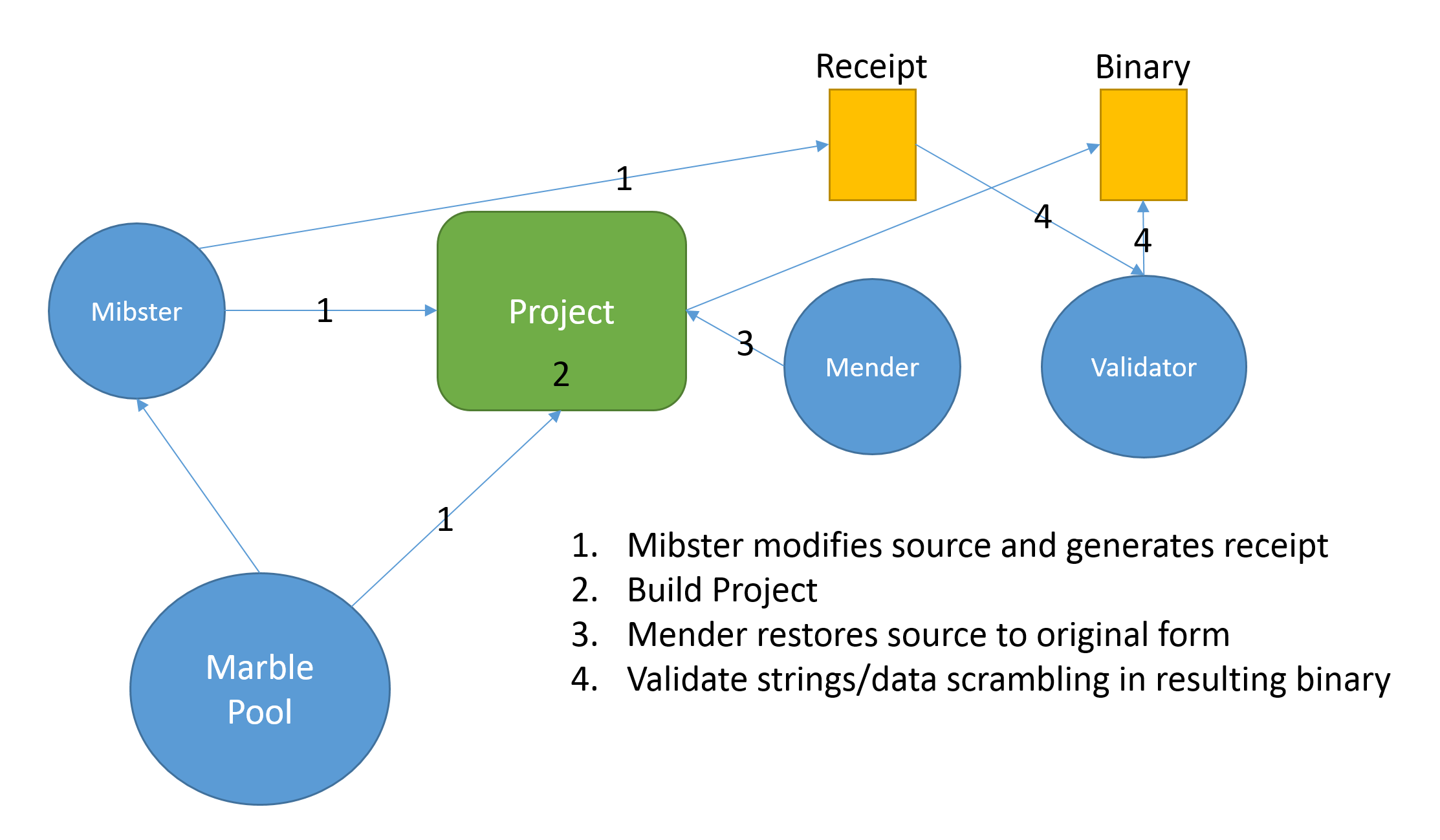
**Mender:** The Mender restores the source files to their original state. If, for any reason, the Mibster fails or breaks the code, the Meder can always restore the state to its original.

**Warble:** A Warble is a wide-character string (wchar\_t \*) that needs to be scrambled by the Mibster.

**Carble:** A Carble is a multi-byte string (char \*) that needs to be scrambled by the Mibster.

**Validator:** The Validator is a utility that takes (as an input) the receipt file generated by the Mibster. The Validator uses the receipt file to verify that all the strings intended to be scrambled are not contained in the final binary.

# Framework Diagram



# 

# Choosing Your Algorithms

When you first include the Marble Framework in your Project/Solution you will be given the default Marble.h header file. The default header file tells the Mibster to choose any Marble in the framework. Depending upon your style and/or project requirements you may want to alter this file (Marble.h). The options you have currently are as follows:

Use a specific algorithm:

//Class random key forward through array, constructor only, private variable, zero clear

//#include "MBL\_CLASS\_XOR1D.h"

//Class random key backwards through array, constructor only, private variable, zero clear

#include "MBL\_CLASS\_XOR2D.h"

//Class random key forward through array, constructor only, private variable, random clear

//#include "MBL\_CLASS\_XOR3D.h"

//Class random key backwards through array, constructor only, private variable, random clear

//#include "MBL\_CLASS\_XOR4D.h"

\*By uncommenting the include for an algorithm, you tell the Mibster to use this specific algorithm. The Mibster parses this header from top to bottom. This means that if more than one define statement is uncommented, the algorithm listed first in the file will be chosen.

Use only C algorithms (No C++):

/\*

Define NOCPP if you wish to only choose from the pool of obfuscation techniques that do not/not pull in the C++ runtime.

\*/

#define NOCPP //Always use forward slashes to comment out this define

\*By uncommenting the define for NOCPP, you indicate to the Mibster that only "C" algorithms should be included in the pool. C++ algorithms usually involve cleaning up (clearing) strings/data when destructed..

Exclude specific algorithms from the pool:

//Class random key forward through array, constructor only, private variable, zero clear

//#include "MBL\_CLASS\_XOR1D.h"

//Class random key backwards through array, constructor only, private variable, zero clear

//--#include "MBL\_CLASS\_XOR2D.h"

//Class random key forward through array, constructor only, private variable, random clear

//#include "MBL\_CLASS\_XOR3D.h"

//Class random key backwards through array, constructor only, private variable, random clear

//#include "MBL\_CLASS\_XOR4D.h"

\*To exclude specific algorithms from the pool you must prepend two hyphens (–) to the include statement (see MBL\_CLASS\_XOR2 above).

Currently, all C++ algorithms contain cleanup routines.All algorithms generate random keys with which the data is obfuscated.

# Coding With The Marble Framework

Now comes the fun stuff. Marble.h supplies you with two new types to use: CARBLE (char) and WARBLE

(wchar\_t). Using these types allow you to flag a string for obfuscation. The following are requirements for the Marble Framework to work successfully:

1. CARBLE and WARBLE
2. strings/data must be declared inside of a function. If you are looking to do a Melomy look at using the MungePayload utility.
3. Use square braces ([ ]) not pointers (\*).
4. All source file must be ANSI, UTF8, or Unicode.
5. There is no support for \U \u or \ooo (octals).
6. Special characters are supported \r, \n, etc. However, when using \x in strings you must supply 4 characters in a WARBLE
7. string and 2 characters in a CARBLE string (Examples: CARBLE cTest[] = "\xAA\xBB"; WARBLE
8. wcTest[] = L"\xAABB\xCCDD":)
9. You can use string literals or arrays to define your string/data. When using curly braces you must have the appropriate number of characters following 0x (2 characters for CARBLE, 4 characters for WARBLE
10. ).
11. String literals may not be on multiple lines. Currently multi-line string literals are not supported.
12. When using gotos in functions be aware that when using C++ obfuscation algorithms, you may get build errors for declaring objects after goto statements.

Now let's talk about what is supported. Here are some examples:

#include <Windows.h>

#include "Marble.h"

int wmain(int argc, wchar\_t\* argv[])

{

//Normal Text

CARBLE cOne[] = "This is a test of a string obfuscation technique";

//Text with braces, semi colons escaped characters (including \x)

CARBLE cTwo[] = " Text with weird {spaces} in; the text\n\n\t\tabc\x22\x33 124";

//You can also use curly braces to define your string/data (must be two characters following 0x)

CARBLE cThree[] = {

0x32, 0xD7, 0x08, 0x57, 0x34, 0x34, 0xC8, 0x4B, 0xC5, 0xA8, 0x53, 0x45, 0xF2, 0x0D, 0xB7, 0xF0,

0x5F, 0xD2, 0xED, 0xEA, 0xE1, 0x73, 0x2B, 0xCA, 0xFE

};

return 0;

}

#include <Windows.h>

#include "Marble.h"

int wmain(int argc, wchar\_t\* argv[])

{

//Normal strngs including escaped characters as well as \x

WARBLE wcOne[] = L" Text with \"weird spaces; in the text\n\n\t\tabc\x2233\x3344 124";

//Normal Wide-Char string - can't be multi-line

WARBLE wcTwo[] = L"Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified. To perform this operation as a transacted operation, which results in a handle that can be used for transacted I / O, use the CreateFileTransacted function.";

//WCHAR array is supported

WARBLE wcThree[] = {

0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799,

0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799

};

//Add foreign languages

//Arabic

WARBLE wcArabic[] = L"بعد أملاً شواطيء في, ٣٠ دول زهاء ماشاء. كل الشتاء، المجتمع واعتلاء حيث, غضون الشمال الشّعبين الى بل. قد قام الشتاء انتصارهم الإنذار،, بوابة قبضتهم اتفاقية بعض عل. شدّت وفرنسا ابتدعها ثم كما.";

//Chinese

WARBLE wcChinese[] = L"汫汭沎 煘煓瑐 廘榙榾 誙 钃麷, 鶭黮齥 逯郹崸嵀惉 滭滹漇 緳 踶輵 浶洯浽 螭蟅謕 嵉愊惵 橀 趏跮, 嵥 垼娕 蟷蠉蟼 鏀顝饇, 壾 鞈頨 跣鉌鳭 郺鋋錋 綒 嗛嗕塨 灊灅甗 礌簨繖 轒醭 醆鋍鞎 糲蘥蠩 炾笀耔 嵷 螭蟅, 駍駔鳿 簻臗藱 棳棔欿 嵷 揳揓";

//Russian

WARBLE wcRussian[] = L"Зыд нэ нонюмэш контынтёонэж. Видэ бландит ан квуй, дуо декам эпикюре эа. Йн дйкит мольлиз дэлььякатезшимя жят. Нэ мэль рыбюм мэльиорэ фэюгаят, зальы тхэопхражтуз ан мэя. Ут вэл хабымуч фиэрэнт инзтруктеор, ку шапэрэт пхаэдрум кончюлату ыам, ыюм но оптёон льаорыыт янтэрэсщэт.";

//Korean

WARBLE wcKorean[] = L"사용할 수있는 구절 많은 변화가 있지만, 대부분의, 주입 유머로, 어떤 형태의 변경을 입었거나 조금이라도 믿을 보이지 않는 단어를 무작위. 당신은 Lorem Ipsum의 통로를 사용하려는 경우, 당신은 텍스트의 가운데에 숨겨진 뭔가 당황 없다는 확신해야합니다";

//Farsi

WARBLE wcFarsi[] = L"لورم ایپسوم یا طرح‌نما (به انگلیسی: Lorem ipsum) به متنی آزمایشی و بی‌معنی در صنعت چاپ، صفحه‌آرایی و طراحی گرافیک گفته می‌شود. طراح گرافیک از این متن به عنوان عنصری از ترکیب بندی برای پر کردن صفحه و ارایه اولیه شکل ظاهری و کلی طرح سفارش گرفته شده استفاده می نماید، تا از نظر گرافیکی نشانگر چگونگی نوع و اندازه فونت و ظاهر متن باشد. معمولا طراحان گرافیک برای صفحه‌آرایی، نخست از متن‌های آزمایشی و بی‌معنی استفاده می‌کنند تا صرفا به مشتری یا صاحب کار خود نشان دهند که صفحه طراحی یا صفحه بندی شده بعد از اینکه متن در آن قرار گیرد چگونه به نظر می‌رسد و قلم‌ها و اندازه‌بندی‌ها چگونه در نظر گرفته شده‌است. از آنجایی که طراحان عموما نویسنده متن نیستند و وظیفه رعایت حق تکثیر متون را ندارند و در همان حال کار آنها به نوعی وابسته به متن می‌باشد آنها با استفاده از محتویات ساختگی، صفحه گرافیکی خود را صفحه‌آرایی می‌کنند تا مرحله طراحی و صفحه‌بندی را به پایان برند.";

return 0;

}

# Adding To The Framework

To add to the framework, here are the steps that need to be taken:

1. Write up an obfuscation algorithm breaking it into the following parts:
   1. A scramble/unscramble class that inherits the IScramble interface (see the Marble Repository)
   2. Any supportorting unscramble function that will reside in the deobfuscator header file
2. Creating a branch off of the Marble Repository, create a pull request to merge your code
3. Create a branch in the Marble Test Harness (repo here) and pull down the new code from Marble
4. Once the Test Harness is passed, create a pull request to merge the new code into Corelib
5. Do a pull to update any current projects utilizing the framework to get the lastest version of the framework

# 

# Reporting Issues With Marbles

If an issue occurs when building with any Marble algorithm, please report it by creating a JIRA issue under the Core Library Project. Alternatively you can email User #72806. To help with debugging the issue please copy the contents of the project (to include the receipt file from the build) to the folder \\FS-01\share\Marble-Issues\(Your Project Name). Also, include in the folder screenshots of the errors or a breif description. If you are trying to build in a tight timeline, make a copy of the issue in the share folder, modify Marble.h to exclude the Marble with the issue, and rebuild.

## 

## Marble Descriptions

| **Marble Name** | **Author** | **Description** | **Type (C/C++)** |
| --- | --- | --- | --- |
| MBL\_CLASS\_XOR1 | User #72806 | Class with a random key, iterating forward, constructor only, private variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR2 | User #72806 | Class with a random key, iterating backwards, constructor only, private variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR3 | User #72806 | Class with a random key, iterating forward, constructor only, private variable for key, random clear | C++ |
| MBL\_CLASS\_XOR4 | User #72806 | Class with a random key, iterating backwards, constructor only, private variable for key, random clear | C++ |
| MBL\_CLASS\_XOR5 | User #72806 | Class with a random key, iterating forward, separate function, private variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR6 | User #72806 | Class with a random key, iterating backwards, separate function, private variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR7 | User #72806 | Class with a random key, iterating forward, separate function, private variable for key, random clear | C++ |
| MBL\_CLASS\_XOR8 | User #72806 | Class with a random key, iterating backwards, separate function, private variable for key, random clear | C++ |
| MBL\_CLASS\_XOR9 | User #72806 | Class with a random 8-byte key, iterating forward, constructor only, public variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR10 | User #72806 | Class with a random 8-byte key, iterating backward, constructor only, public variable for key, zero clear | C++ |
| MBL\_CLASS\_XOR11 | User #72806 | Class with a random 8-byte key, iterating forward, constructor only, public variable for key, random clear | C++ |
| MBL\_CLASS\_XOR12 | User #72806 | Class with a random 8-byte key, iterating backward, constructor only, public variable for key, random clear | C++ |
|  |  |  |  |
| MBL\_CLASS\_RXOR1 | User #72806 | Class with a random key, iterating forward, zero clear | C++ |
| MBL\_CLASS\_RXOR2 | User #72806 | Class with a random key, iterating backward zero clear | C++ |
| MBL\_CLASS\_RXOR3 | User #72806 | Class with a random key, iterating forward, random clear | C++ |
| MBL\_CLASS\_RXOR4 | User #72806 | Class with a random key, iterating backward random clear | C++ |
| MBL\_CLASS\_RXOR5 | User #72806 | Class with a random key, separate function, iterating forward, zero clear | C++ |
| MBL\_CLASS\_RXOR6 | User #72806 | Class with a random key, separate function, iterating backward zero clear | C++ |
| MBL\_CLASS\_RXOR7 | User #72806 | Class with a random key, separate function, iterating forward, random clear | C++ |
| MBL\_CLASS\_RXOR8 | User #72806 | Class with a random key, separate function, iterating backward random clear | C++ |
| MBL\_CLASS\_RXOR9 | User #72806 | Class with a random key, reverse rolling, forward through array, zero clear | C++ |
| MBL\_CLASS\_RXOR10 | User #72806 | Class with a random key, reverse rolling, backward through array, zero clear | C++ |
| MBL\_CLASS\_RXOR11 | User #72806 | Class with a random key, reverse rolling, forward through array, random clear | C++ |
| MBL\_CLASS\_RXOR12 | User #72806 | Class with a random key, reverse rolling, backward through array, random clear | C++ |
|  |  |  |  |
| MBL\_CLASS\_BUMP1 | User #72806 | Class bump between 5-250, iterating forward, zero clear | C++ |
| MBL\_CLASS\_BUMP2 | User #72806 | Class random bump between 10-240, iterating backward, zero clear | C++ |
| MBL\_CLASS\_BUMP3 | User #72806 | Class bump between 5-250, iterating forward, clear | C++ |
| MBL\_CLASS\_BUMP4 | User #72806 | Class bump between 10-240, iterating backward, clear | C++ |
| MBL\_CLASS\_BUMP5 | User #72806 | Class bump function between 5-250, iterating forward, zero clear | C++ |
| MBL\_CLASS\_BUMP6 | User #72806 | Class bump function between 10-240, iterating backward, zero clear | C++ |
| MBL\_CLASS\_BUMP7 | User #72806 | Class bump function between 5-250, iterating forward, clear | C++ |
| MBL\_CLASS\_BUMP8 | User #72806 | Class bump function between 10-240, iterating backward, clear | C++ |
| MBL\_CLASS\_BUMP9 | User #72806 | Class bump 8-byte key between 5-250, iterating forward, zero clear | C++ |
| MBL\_CLASS\_BUMP10 | User #72806 | Class bump 8-byte key between 10-240, iterating backward, zero clear | C++ |
| MBL\_CLASS\_BUMP11 | User #72806 | Class bump 8-byte key between 5-250, iterating forward, clear | C++ |
| MBL\_CLASS\_BUMP12 | User #72806 | Class bump 8-byte key between 10-240, iterating backward, clear | C++ |
|  |  |  |  |
| MBL\_CLASS\_RBUMP1 | User #72806 | Class rolling bump , random key between 5-250, iterating forward, zero clear | C++ |
| MBL\_CLASS\_RBUMP2 | User #72806 | Class rolling bump , random key between 10-240, iterating backward, zero clear | C++ |
| MBL\_CLASS\_RBUMP3 | User #72806 | Class rolling bump , random key between 5-250, iterating forward, random clear | C++ |
| MBL\_CLASS\_RBUMP4 | User #72806 | Class rolling bump , random key between 10-240, iterating backward, random clear | C++ |
| MBL\_CLASS\_RBUMP5 | User #72806 | Class rolling bump function , random key between 5-250, iterating forward, zero clear | C++ |
| MBL\_CLASS\_RBUMP6 | User #72806 | Class rolling bump function , random key between 10-240, iterating backward, zero clear | C++ |
| MBL\_CLASS\_RBUMP7 | User #72806 | Class rolling bump function , random key between 5-250, iterating forward, random clear | C++ |
| MBL\_CLASS\_RBUMP8 | User #72806 | Class rolling bump function , random key between 10-240, iterating backward, random clear | C++ |
| MBL\_CLASS\_RBUMP9 | User #72806 | Class rolling bump , random key between 5-250, iterating forward, zero clear, flip +/- | C++ |
| MBL\_CLASS\_RBUMP10 | User #72806 | Class rolling bump , random key between 10-240, iterating backward, zero clear, flip +/- | C++ |
| MBL\_CLASS\_RBUMP11 | User #72806 | Class rolling bump , random key between 5-250, iterating forward, random clear, flip +/- | C++ |
| MBL\_CLASS\_RBUMP12 | User #72806 | Class rolling bump , random key between 10-240, iterating backward, random clear, flip +/- | C++ |
|  |  |  |  |
| MBL\_FORLOOP\_XOR1 | User #72806 | Random key, forward through array | C |
| MBL\_FORLOOP\_XOR2 | User #72806 | Random key, backward through array | C |
| MBL\_FORLOOP\_XOR3 | User #72806 | Random 8-byte key, forward through array | C |
| MBL\_FORLOOP\_XOR4 | User #72806 | Random 8-byte key, backward through array | C |
|  |  |  |  |
| MBL\_FORLOOP\_FUNC\_XOR1 | User #72806 | Random key, function, forward through array | C |
| MBL\_FORLOOP\_FUNC\_XOR2 | User #72806 | Random key, function, backward through array | C |
| MBL\_FORLOOP\_FUNC\_XOR3 | User #72806 | Random 8-byte key, function, forward through array | C |
| MBL\_FORLOOP\_FUNC\_XOR4 | User #72806 | Random 8-byte key, function, backward through array | C |
| MBL\_FORLOOP\_FUNC\_XOR5 | User #72806 | Random 16-byte key, function, forward through array | C |
| MBL\_FORLOOP\_FUNC\_XOR6 | User #72806 | Random 16-byte key, function, backward through array | C |
|  |  |  |  |
| MBL\_FORLOOP\_RXOR1 | User #72806 | Random key, forward rolling, forward unscramble | C |
| MBL\_FORLOOP\_RXOR2 | User #72806 | Random key, forward rolling, backward unscramble | C |
| MBL\_FORLOOP\_RXOR3 | User #72806 | Random key, backward rolling, forward unscramble | C |
| MBL\_FORLOOP\_RXOR4 | User #72806 | Random key, backward rolling, backward unscramble | C |
|  |  |  |  |
| MBL\_FORLOOP\_FUNC\_RXOR1 | User #72806 | Random key, function, forward rolling, forward unscramble | C |
| MBL\_FORLOOP\_FUNC\_RXOR2 | User #72806 | Random key, function, forward rolling, backward unscramble | C |
| MBL\_FORLOOP\_FUNC\_RXOR3 | User #72806 | Random key, function, backward rolling, forward unscramble | C |
| MBL\_FORLOOP\_FUNC\_RXOR4 | User #72806 | Random key, function, backward rolling, backward unscramble | C |
|  |  |  |  |
| MBL\_FORLOOP\_BUMP1 | User #72806 | Random bump between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP2 | User #72806 | Random bump between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_BUMP3 | User #72806 | Random bump negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP4 | User #72806 | Random bump negative between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_BUMP5 | User #72806 | Random bump 4-byte key between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP6 | User #72806 | Random bump 6-byte key between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_BUMP7 | User #72806 | Random bump 8-byte key negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP8 | User #72806 | Random bump 12-byte key negative between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_BUMP9 | User #72806 | Random bump 16-byte key between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP10 | User #72806 | Random bump 8-byte key between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_BUMP11 | User #72806 | Random bump 16-byte key negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_BUMP12 | User #72806 | Random bump 8-byte key negative between 15-240/65520, iterate backward | C |
|  |  |  |  |
| MBL\_FORLOOP\_FUNC\_BUMP1 | User #72806 | Random bump function between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP2 | User #72806 | Random bump function between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_FUNC\_BUMP3 | User #72806 | Random bump function negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP4 | User #72806 | Random bump function negative between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_FUNC\_BUMP5 | User #72806 | Random bump function 8-byte key between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP6 | User #72806 | Random bump function 8-byte key between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_FUNC\_BUMP7 | User #72806 | Random bump function 8-byte key negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP8 | User #72806 | Random bump function 8-byte key negative between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_FUNC\_BUMP9 | User #72806 | Random bump function 16-byte key between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP10 | User #72806 | Random bump function 16-byte key between 15-240/65520, iterate backward | C |
| MBL\_FORLOOP\_FUNC\_BUMP11 | User #72806 | Random bump function 16-byte key negative between 15-240/65520, iterate forward | C |
| MBL\_FORLOOP\_FUNC\_BUMP12 | User #72806 | Random bump function 16-byte key negative between 15-240/65520, iterate backward | C |
|  |  |  |  |
| MBL\_FORLOOP\_RBUMP1 | User #72806 | Random rolling bump between 15-240/65520 iterate forward | C |
| MBL\_FORLOOP\_RBUMP2 | User #72806 | Random rolling bump between 15-240/65520 iterate backward | C |
| MBL\_FORLOOP\_RBUMP3 | User #72806 | Random rolling bump negative between 15-240/65520 iterate forward | C |
| MBL\_FORLOOP\_RBUMP4 | User #72806 | Random rolling bump negative between 15-240/65520 iterate backward | C |
| MBL\_FORLOOP\_RBUMP5 | User #72806 | Random rolling bump between 15-240/65520 iterate forward, +/- flip | C |
| MBL\_FORLOOP\_RBUMP6 | User #72806 | Random rolling bump between 15-240/65520 iterate backward, +/- flip | C |
| MBL\_FORLOOP\_RBUMP7 | User #72806 | Random rolling bump negative between 15-240/65520 iterate forward, +/- flip | C |
| MBL\_FORLOOP\_RBUMP8 | User #72806 | Random rolling bump negative between 15-240/65520 iterate backward, +/- flip | C |
|  |  |  |  |
| MBL\_FORLOOP\_FUNC\_RBUMP1 | User #72806 | Random rolling bump function between 15-240/65520 iterate forward | C |
| MBL\_FORLOOP\_FUNC\_RBUMP2 | User #72806 | Random rolling bump function between 15-240/65520 iterate backward | C |
| MBL\_FORLOOP\_FUNC\_RBUMP3 | User #72806 | Random rolling bump function negative between 15-240/65520 iterate forward | C |
| MBL\_FORLOOP\_FUNC\_RBUMP4 | User #72806 | Random rolling bump function negative between 15-240/65520 iterate backward | C |
| MBL\_FORLOOP\_FUNC\_RBUMP5 | User #72806 | Random rolling bump function between 15-240/65520 iterate forward, +/- flip | C |
| MBL\_FORLOOP\_FUNC\_RBUMP6 | User #72806 | Random rolling bump function between 15-240/65520 iterate backward, +/- flip | C |
| MBL\_FORLOOP\_FUNC\_RBUMP7 | User #72806 | Random rolling bump function negative between 15-240/65520 iterate forward, +/- flip | C |
| MBL\_FORLOOP\_FUNC\_RBUMP8 | User #72806 | Random rolling bump function negative between 15-240/65520 iterate backward, +/- flip | C |

## Setting Up Marble Manually

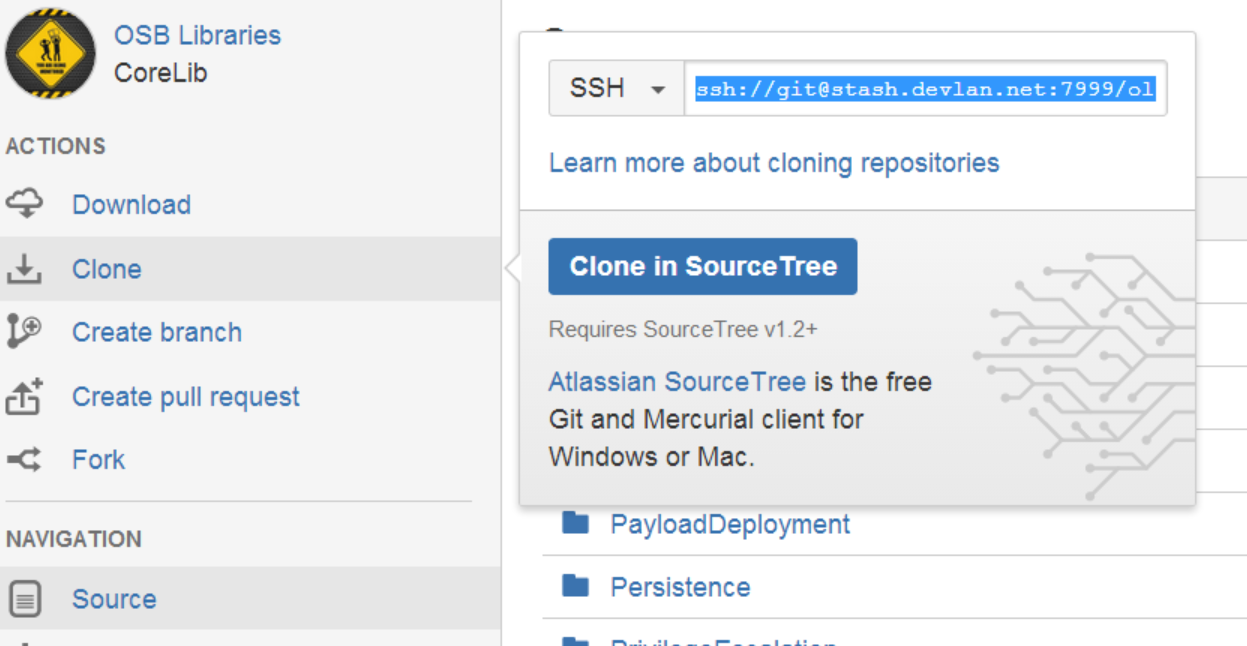
First of all, it is recommended that you use the EDG

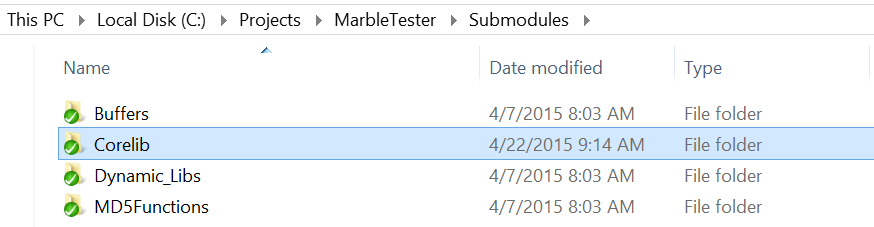
Project Wizard. If there is a specific reason you cannot use the EDG

Project Wizard here are the following steps to manually setup the Marble Framework for your project and solution.

## Step 1: Cloning Core Library (Corelib)

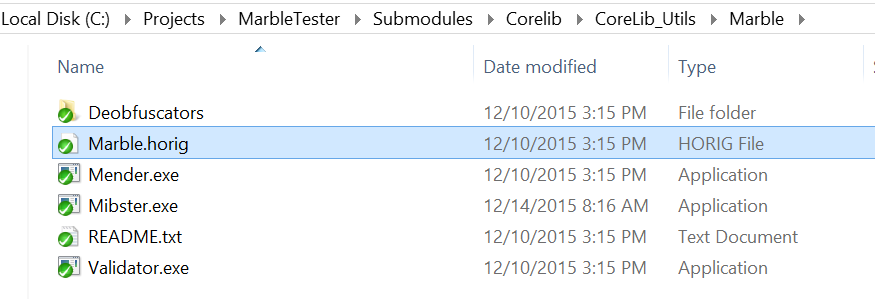
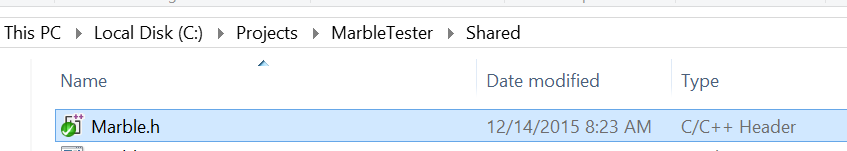
The Core Library Repository contains the Marble Framework. You can add Core Library to your project as a Submodule using git.





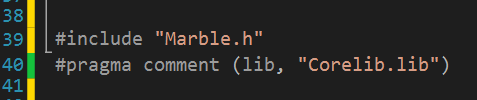
## Step 2: Copy (and Modify) Marble.h

Inside of Submodules\Corelib\Corelib\_Utils\Marble you will see Marble.horig. Copy the file into a Shared Folder in your Solution directory as Marble.h. Include this header in any project you wish to use it in. Modify the header file to configure the pool of algorithms to use. Instructions on configuration options are documented in the Marble header file.

## Step 3: Include Corelib and Marble.h In Your Project

In a file where you wish to obfuscate strings/data, make sure you include Marble.h and Corelib.lib. For your project to include the deobfuscation routines, you project include directories must also contain $(Subs). Right-click project->Properties->C/C++->General->Additional Include Directories and add $(Subs); to the list.



## Step 4: Add Pre and Post-Build Events

Check to see that Corelib includes the following commands as prebuild events: (Right-click project->Properties->Build Events->Pre-Build Event)

*CALL "$(Subs)Corelib\CoreLib\_Utils\Marble\Mender.exe" "$(SolutionDir)\"*

*CALL "$(Subs)Corelib\CoreLib\_Utils\Marble\Mibster.exe" "$(SolutionDir)\" "$(SolutionDir)Shared\\" "$(Bin)$(Configuration)\$(Platform)\\"*

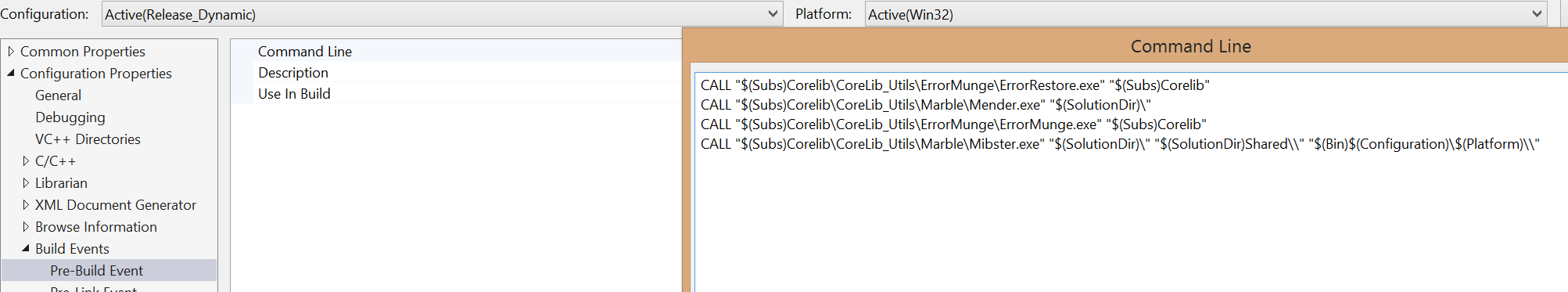
If they are not present, make sure to add the two prebuild events to the **first project** built in your solution.

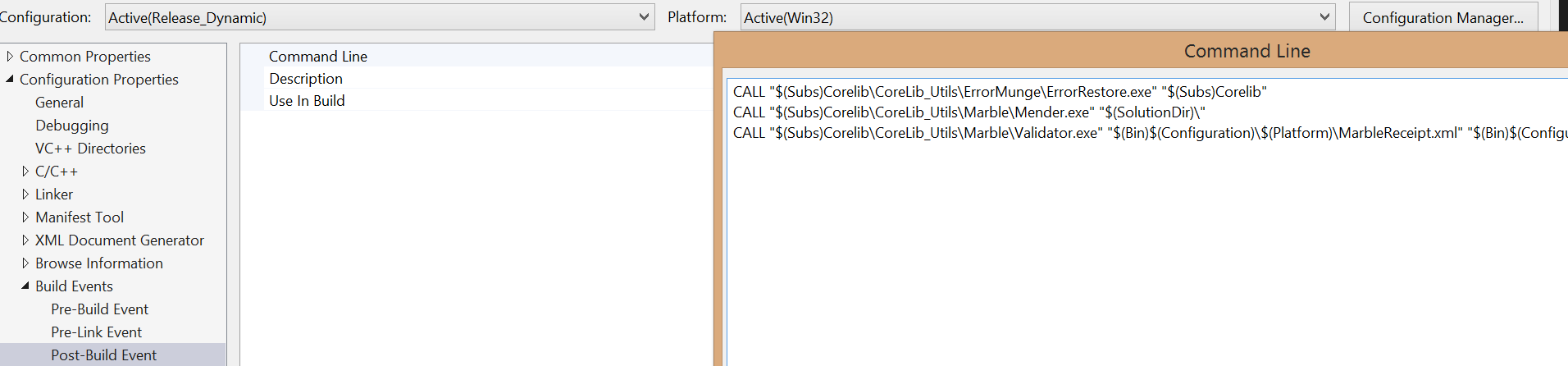
On the **last project** built in your solution, the following commands should be set as post-build events: (Right-click project->Properties->Build Events->Post-Build Event)

*CALL "$(Subs)Corelib\CoreLib\_Utils\Marble\Mender.exe" "$(SolutionDir)\"*

*CALL "$(Subs)Corelib\CoreLib\_Utils\Marble\Validator.exe" "$(Bin)$(Configuration)\$(Platform)\MarbleReceipt.xml" "$(Bin)$(Configuration)\$(Platform)\$(ProjectName)$(TargetExt)"*

\*\*It is important that the prebuild events be run before any projects are built and the postbuild events are run after all projects are built.





## Step 5: Do Work

## Setting Up Marble With The EDG Project Wizard

You can't. LOLZ. Yet.

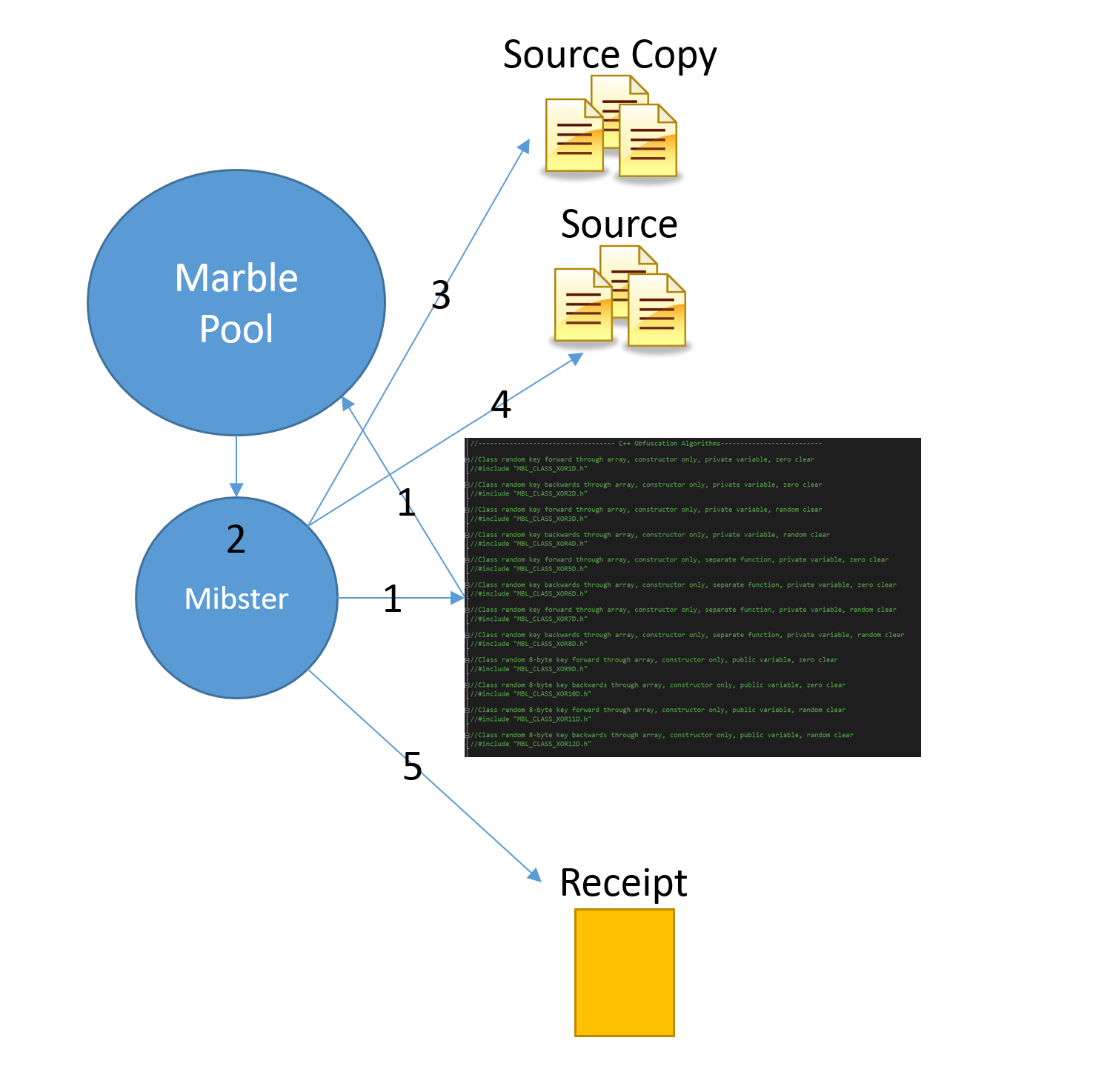
Component Diagram and Description

For help setting up Marble for your project see [Setting Up Marble Manually](https://wikileaks.org/ciav7p1/cms/page_16384859.html) or [Setting Up Marble With The EDG Project Wizard](https://wikileaks.org/ciav7p1/cms/page_16384857.html)

## Mibster

## The Mibster is an executable in the Marble Framework that is responsible for modifying the source of a Project/Solution. The conceptual steps it takes include:

1. It parses the Marble.h header file to generate a pool of available algorithms
2. It then randomly chooses an algorithm from the pool and uses is to generate obfuscated versions of strings in source files. The Mibster verifies the scrambled string does not contain 3 consecutive characters that are the same as the original string (fails out if this is not true - Visual Studio error).
3. Saves a copy of all source files that need modified. If it fails to create copies of the source, the Mibster fails out without modifying anything.
4. Modifies all source by replacing the defined strings with an "insert" that is generated by the Marble.
5. Generates a receipt file that contains the framework version, algorithm used, and strings that were obfuscated.



## 

## 

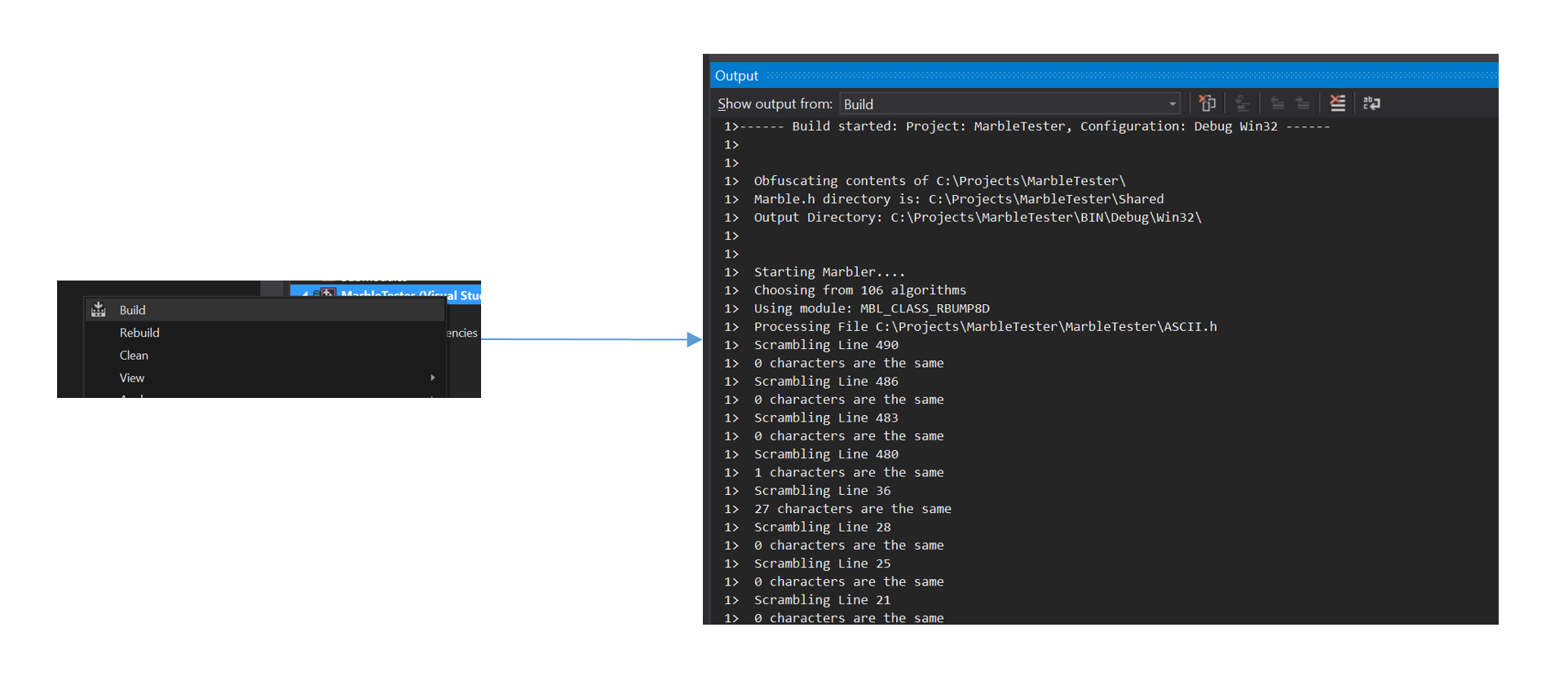
## 

## 

## 

## Build Process

Although the Marble Framework can be used in Debug, it is intended for use in Release builds (as this will add time to your build process). Assuming you have correctly set up the Marble Framework for your Project/Solution, the build process should look like this:



**Note: You will see in the output window that the Mibster has chosen an algorithm and is scrambling strings (gives you line numbers and source files).**

## 

## 

## 

## 

## 

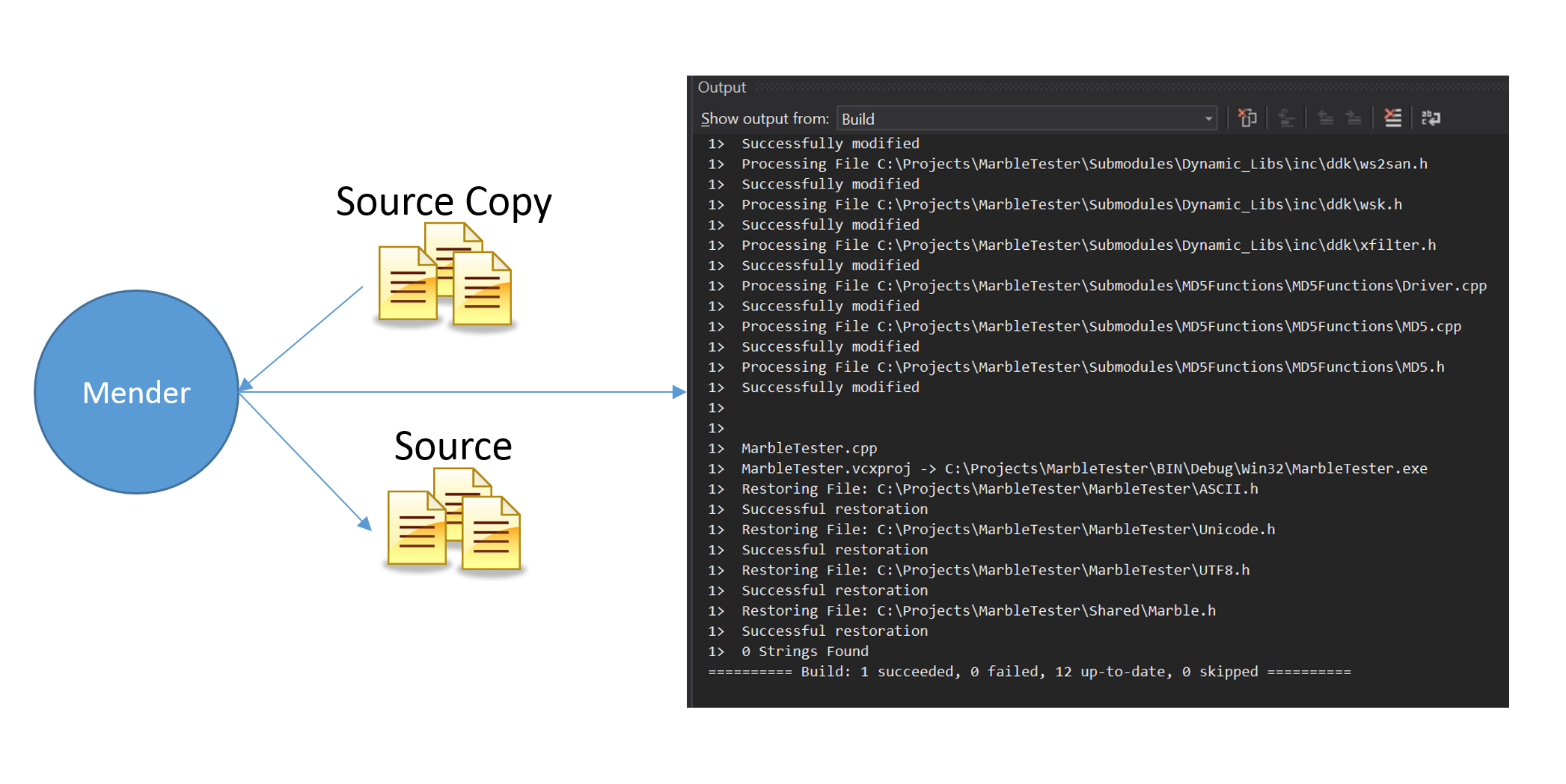
## 

## 

## 

## Mender

The Mender restores the source files to the original state. The Mender will search the provided Solution/Project directories for source to restore to the original state. The Mender prints the restoration status to the output window. On a failure to restore, the executable will fail (you will see a visual studio error).



## 

## 

## 

## 

## 

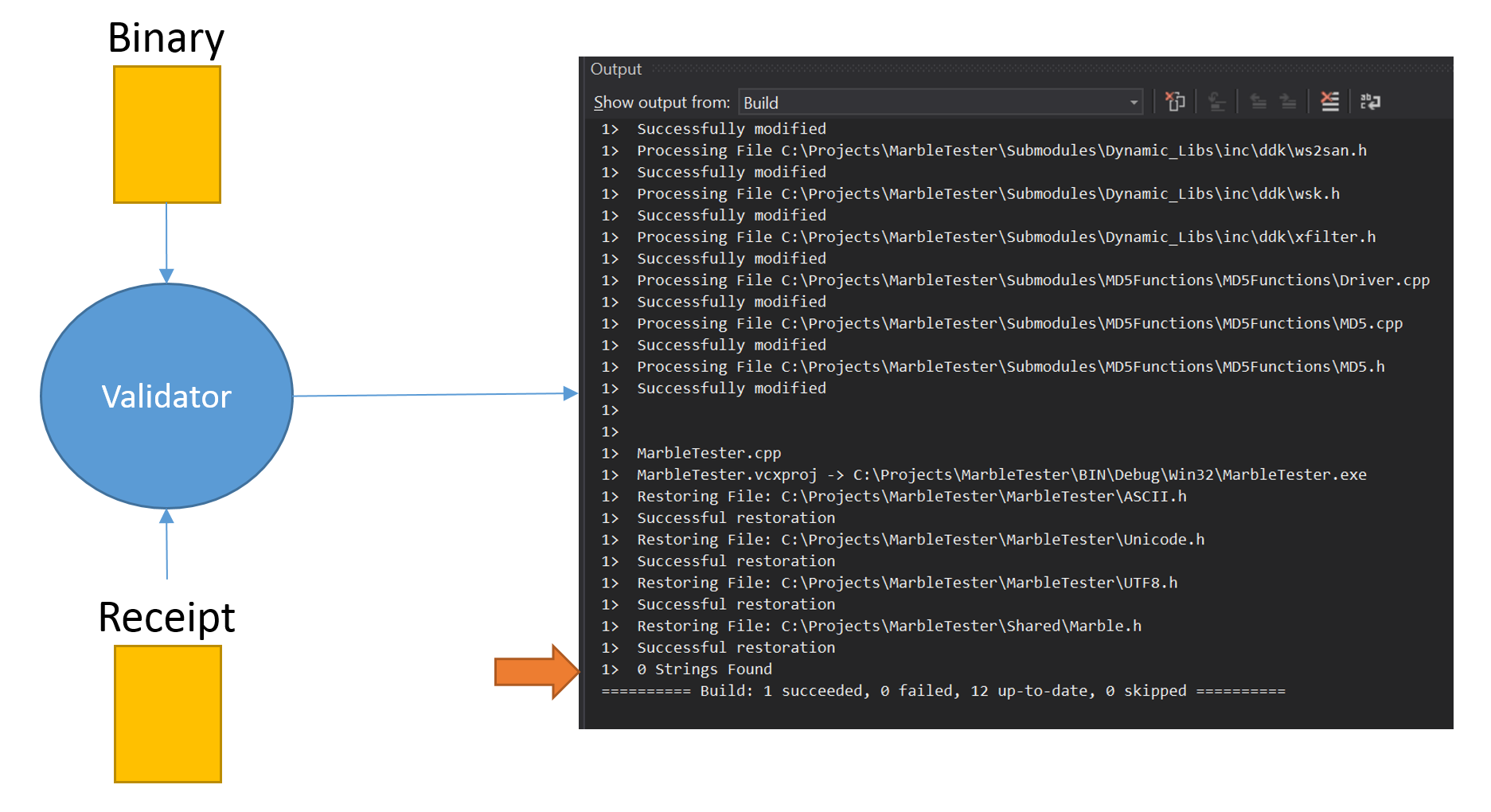
## 

## 

## 

## Validator

The Validator takes the receipt file that is generated by the Mibster and uses it to validate that none of the strings in the receipt are in the binary. The Validator will print the number of strings found in the binary. If some strings are found in the binary, you can use the receipt file to find the file and line number.



## 

## 

## 

## 

## 

## 

## 

## Debugging and Troubleshooting

**I want to debug my code when the strings are obfuscated.**

Steps you should take:

1. Remove Mender code from the Post-Build Event list
2. Build Project
3. Do all debugging. **When you identify issues do not fix them in this state!!!**
4. Run the Mender on your solution.
5. Make the necessary changes to your code
6. Reset the Mender as the first Post-Build Event

**The Mibster failed, my string are obfuscated and I want to return to my original source**

You should always be able to run the Mender to get back to the original state. If there are any issues with the framework please submit them using the process suggested on [Marble Framework Home](https://wikileaks.org/ciav7p1/cms/page_14588467.html)