XYZ Company

RWPA Phase 2:
Reader-Writer Process Application Phase 2
Design Specifications Document

Revision 2.1
26.11.2012

By: Bulut Korkmaz

Revision History

Revision	Date	Explanation
1.0	15.10.2012	Initial high level design
1.1	05.11.2012	Improvements
2.0	19.11.2012	Design is changed prior to RSD change.
2.1	26.11.2012	2.2 Class diagram is updated 2.2.3 Restriction about data size is added. 3.2.1 Recordcounter semaphore is added in order to prevent starvation 3.2.2 Recordcounter semaphore is added in order to prevent starvation 3.3.2 Note added, function names and base AES class is changed. 3.3.3 Function names and base AES class is changed.

Contents

1. Introduction

The purpose of the software project is to develop an application in C#, Windows environment, to do mainly the following.

1. Simulate Readers&Writers problem.
2. Encrypt&Decrypt some random data as Readers&Writers' job.
3. Log every action to a file.

The design is based on RWPA Requirements Specification Document, Revision 1.0, in file Korkmaz-P2-RWPA-Phase2-RSD-2012-11-12-Rev-2.0 [1].

The software architecture and overall high-level structure and components in terms of classes of RWPA are given in Section 2 and design details of all application functions and the user interface in terms of are methods of all classes are given in Section 3 of this document.

2. System Design

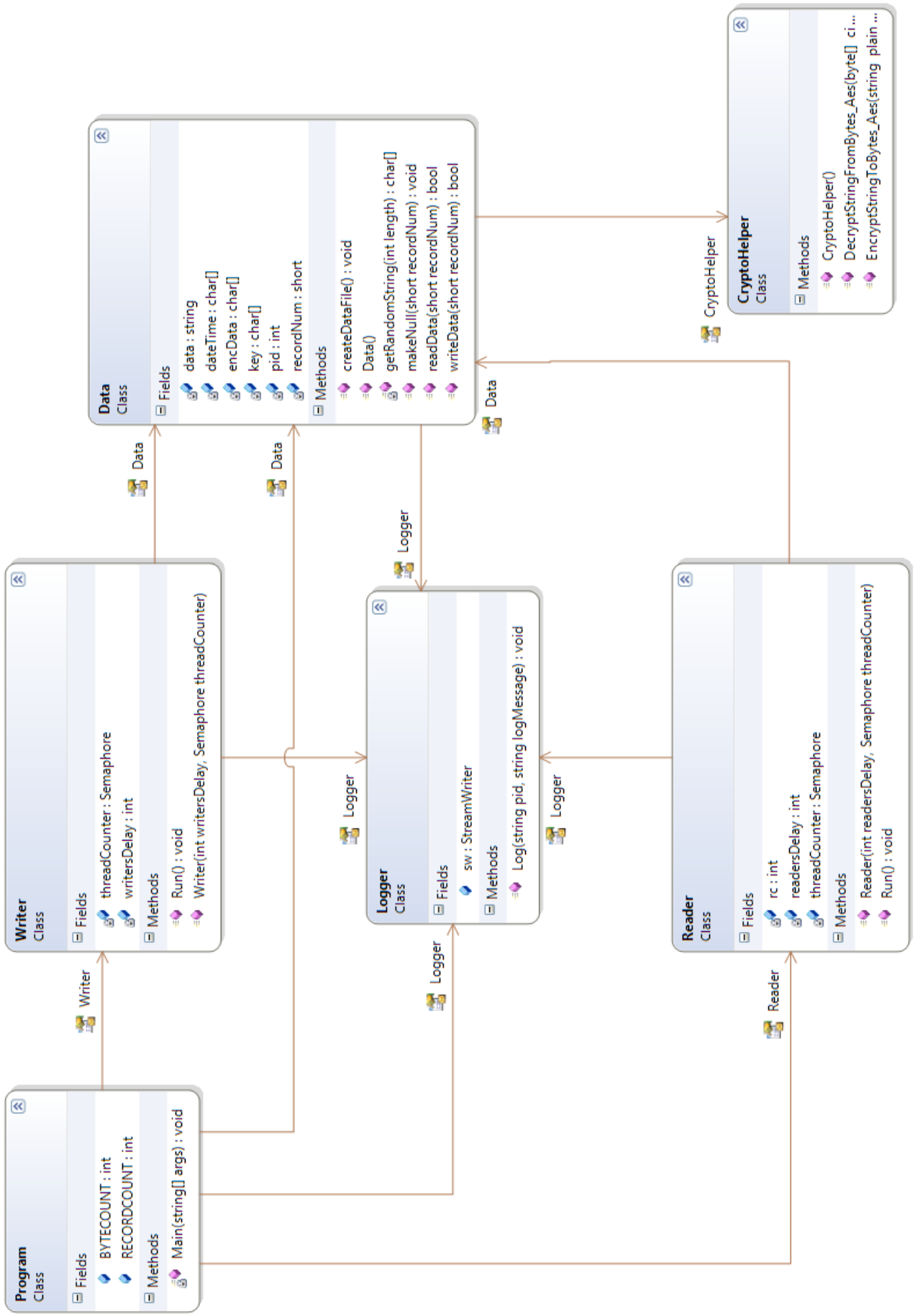
2.1. Software Architecture

Software architecture is multi-threaded classical objects.

2.2. Software System Components

The application is a console application and reads needed information from a config file and log actions to a log file. Further information about filenames is on below.

Class Diagram



2.2.2 Config file structure

Config.rwpa structure
mDelay:int (delay for writers) nDelay:int (delay for readers) readersNumber:int (# of readers) writersNumber:int (# of writers) recordNumber:int (# of records) byteCount:int >200(byte size of each record)

The application must read the number of readers and writer processes from a config file named : config.rwpa.

Number of readers and writer can be set differently.

Process delays must be set in config file also.

Application must control if the input in the config file is a valid positive decimal number.

Config file must be ready to read with C# AppConfig class.

Structure example:

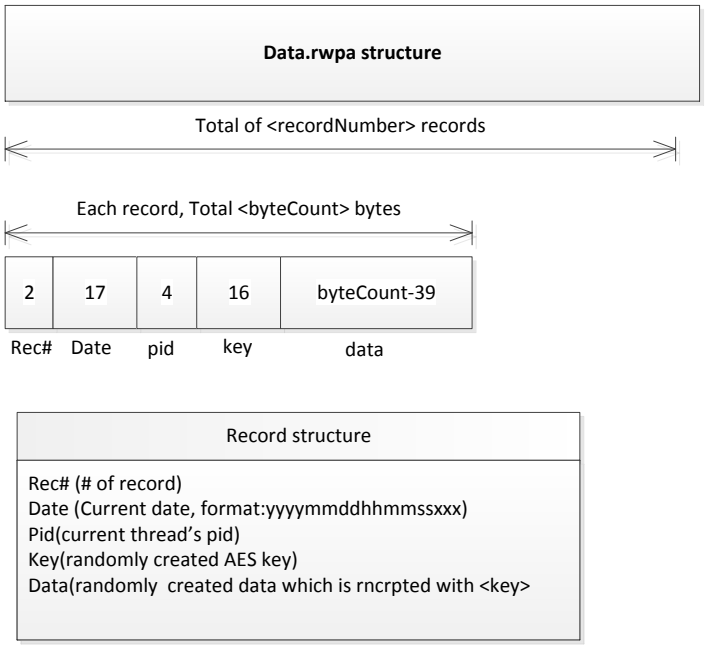
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="readersNumber" value="5"/>
    <add key="writersNumber" value="5"/>
    <add key="delayN" value="50"/>
    <add key="delayM" value="50"/>
    <add key="demoLength" value="5"/>
    <add key="byteCount" value="200"/>
    <add key="recordCount" value="5"/>
  </appSettings>
</configuration>
```

2.2.3 Log file structure

Log format: date&time in msec, pid of process, log message
(time format: yyyymmdd hh/mm/ss/sss)
Example:
20120810 09/45/12/332, pid=1234, main program starts

Each significant event/action of main program or reader or writer processes is to be logged to a shared log file.
In order to prevent scrambling of letters of different processes, it is better to do logging when in critical section of program, just after program delay.

2.2.3 Data file structure

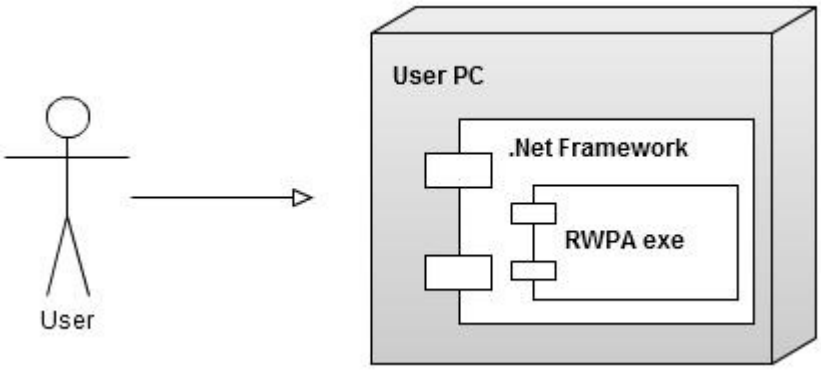


Record structure
Rec# (# of record) Date (Current date, format:yyyymmddhhmmssxxx) Pid(current thread's pid) Key(randomly created AES key) Data(randomly created data which is rncrpted with <key>

Note that, data size must be in multiples of 16 bytes because of AES algorithm.

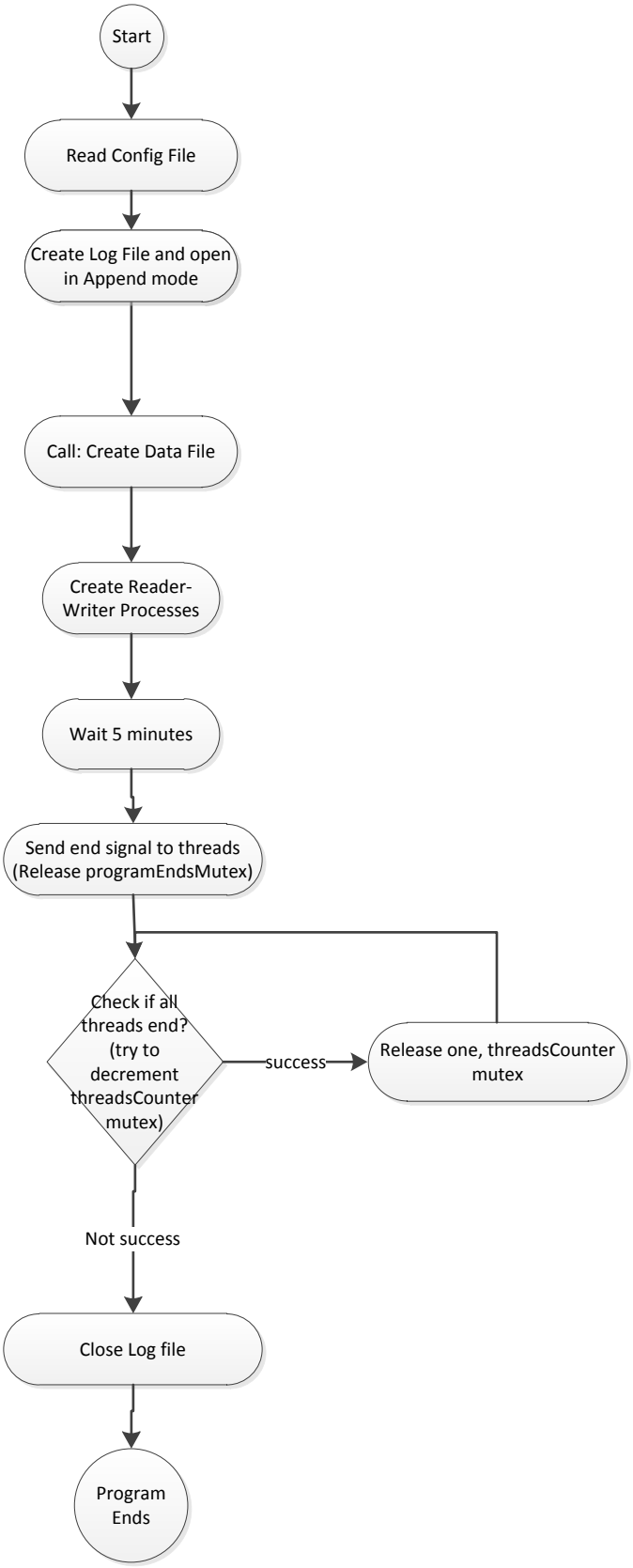
2.3. Environment

- This software must be ready to execute windows 7 32bit or 64bit computer.
- Programming Language is C#, and framework is .NET 3.5.



3. Detailed Design:

3.1. Main Module/Class

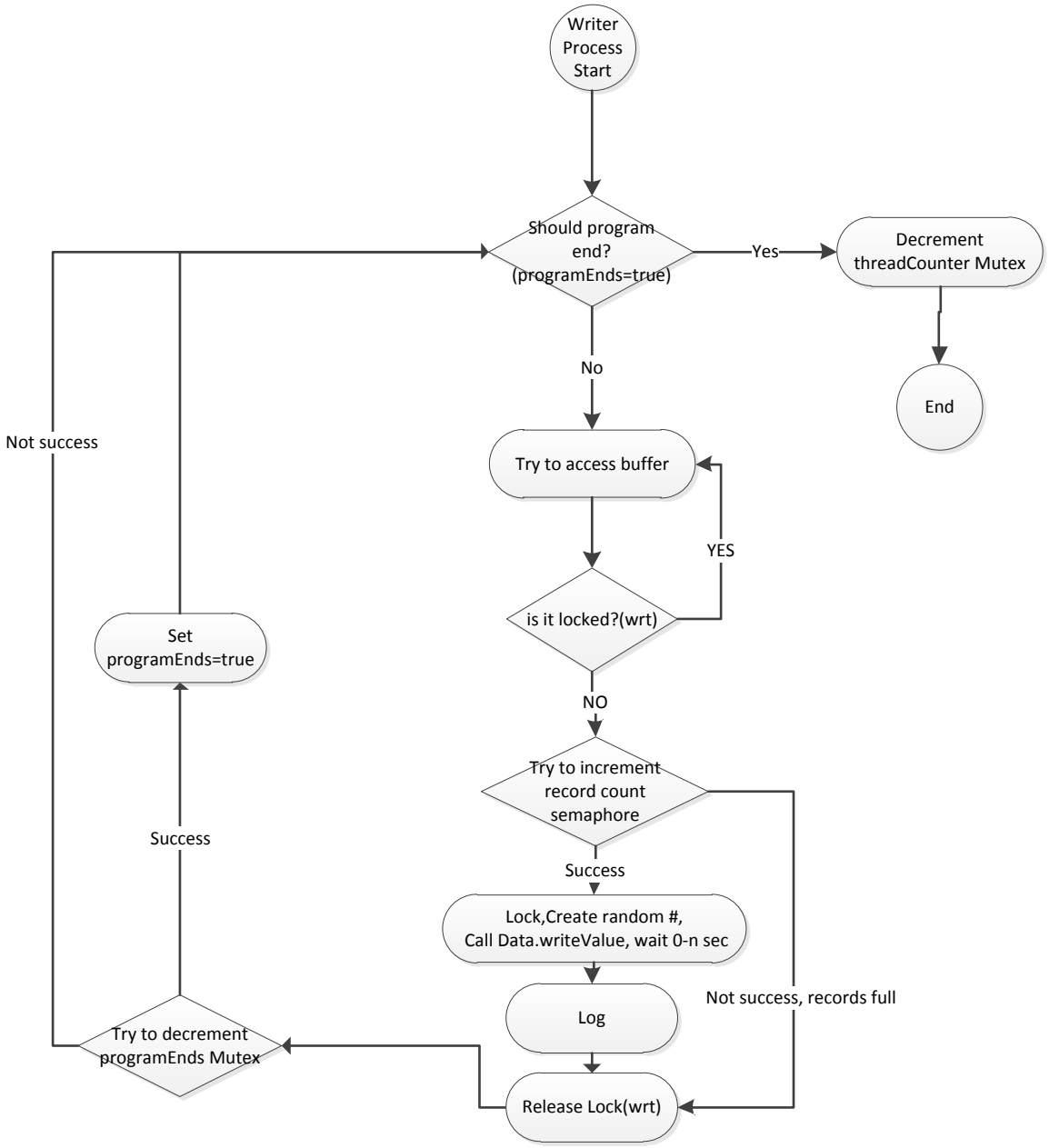


3.2. Other Modules/Classes

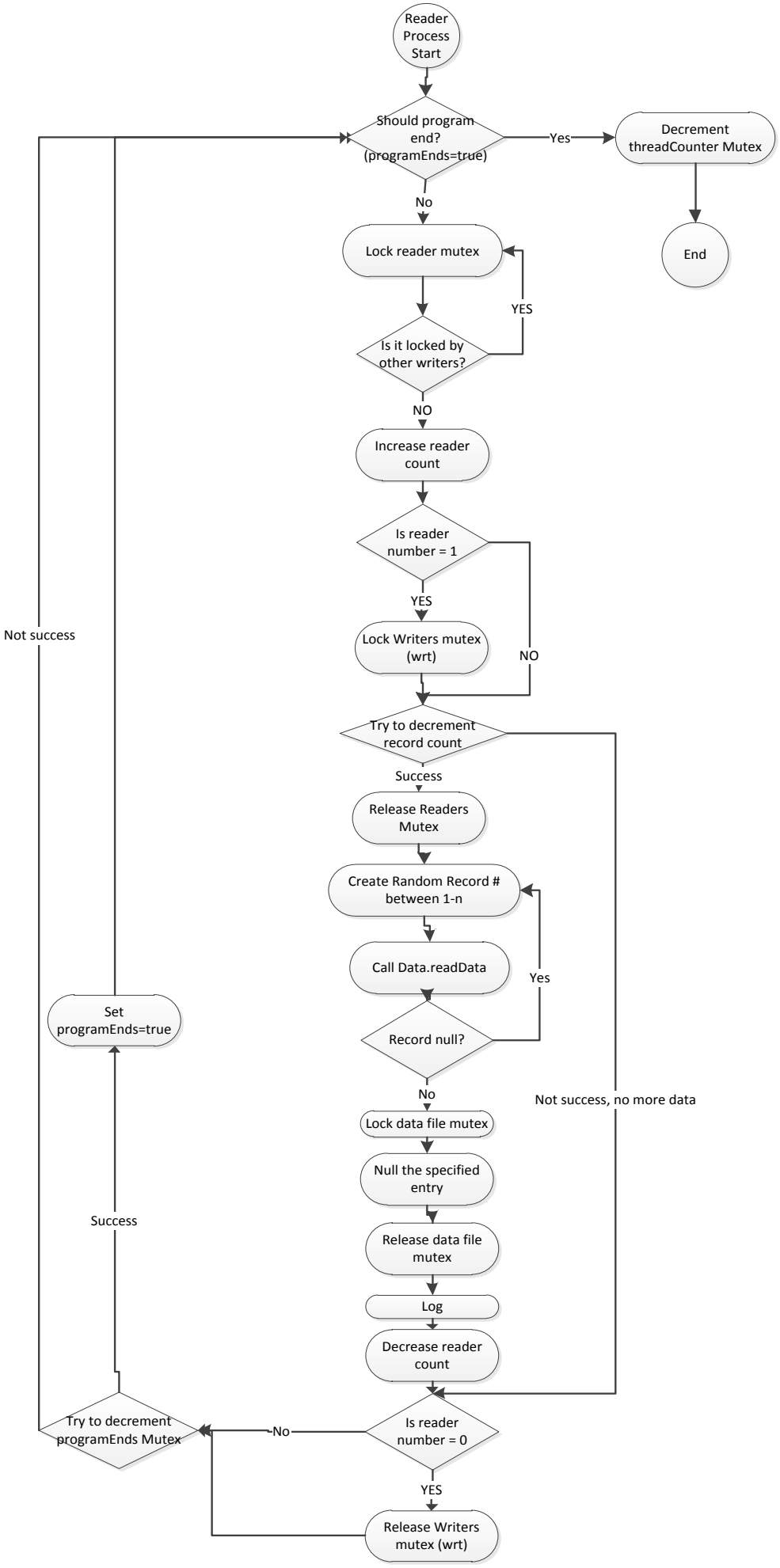
Processes will try to access a shared buffer.
Concurrent write access must be prevented using mutex.
When readers is accessing, write access must be prevented using mutex.

Actions must be simulated by random delay, 0-n msecs for readers, 0-m msecs for writers, uniformly distributed, n & m to be specified in config.rwpa.
Class diagram is stated above.

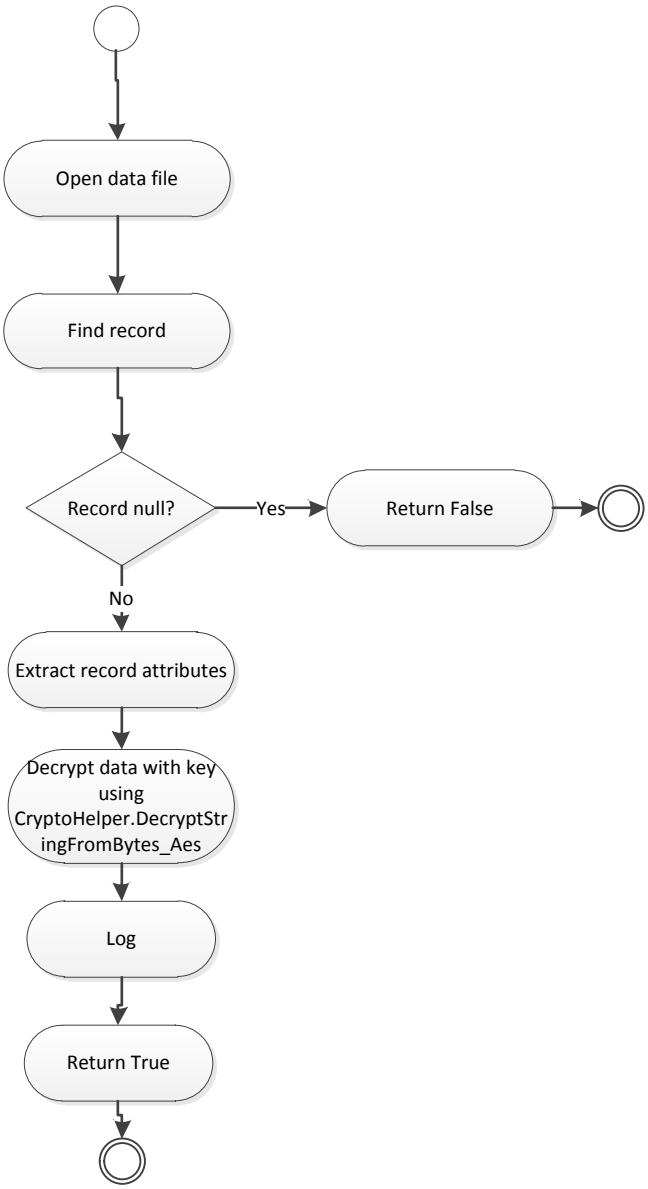
3.2.1. Writer/RUN()



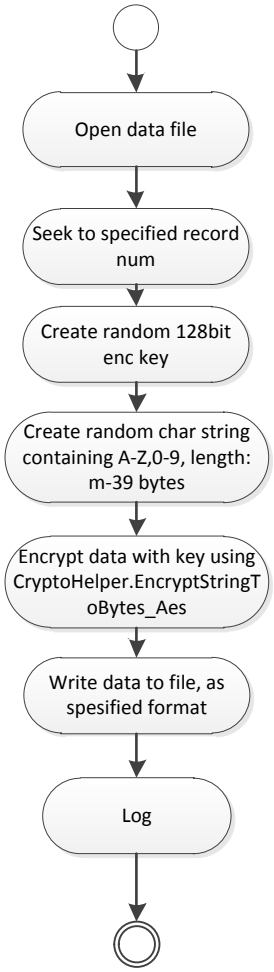
3.2.2. Reader/RUN()


Reader Process Start
Should program end? (programEnds=true)
Yes
Decrement threadCounter Mutex
End
No
Lock reader mutex
YES
Is it locked by other writers?
NO
Increase reader count
Is reader number = 1
YES
Lock Writers mutex (wrt)
NO
Try to decrement record count
Success
Release Readers Mutex
Create Random Record # between 1-n
Call Data.readData
Yes
Record null?
No
Lock data file mutex
Null the specified entry
Release data file mutex
Log
Decrease reader count
Is reader number = 0
YES
Release Writers mutex (wrt)
No
Try to decrement programEnds Mutex
Success
Set programEnds=true
Not success
Not success, no more data

3.2.3. Data/readData()

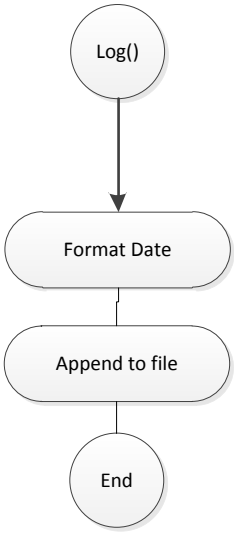


3.2.3. Data/writeData()

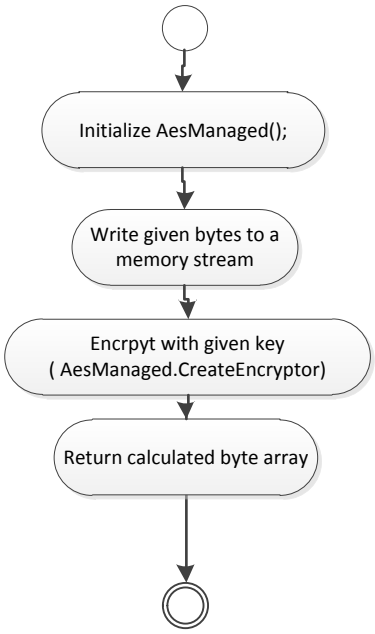


3.3. Common Infrastructure Modules/Classes

3.3.1. Logger/LOG()

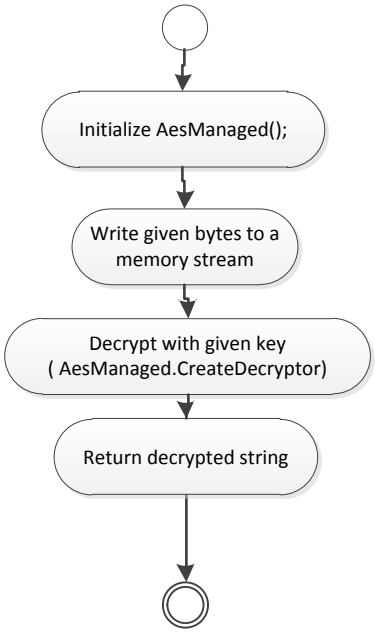


3.3.2. CryptoHelper/Encrypt()



AesManaged class requires initilization vector. We alse use key for IV. Also Padding must be zero. Default block size = 126bit.

3.3.3. CryptoHelper/Decrypt()



3. Testing Design

- Demo and test will be end in 5 minutes. So, after program starts, main threat should wait 5 minutes and send signal to child threads to end properly.
- Special testing for indefinite postponement may needed.

4. References

1. Korkmaz-P2-RWPA-Phase2-RSD-2012-11-12-Rev-2.1
2. Korkmaz-P2-RWPA-Phase2-DSD-2012-11-19-Rev-2.0