

**Kiwinest**

**Developer’s guide**

2024.06.04 last updated – Choi Changho(op11091109)

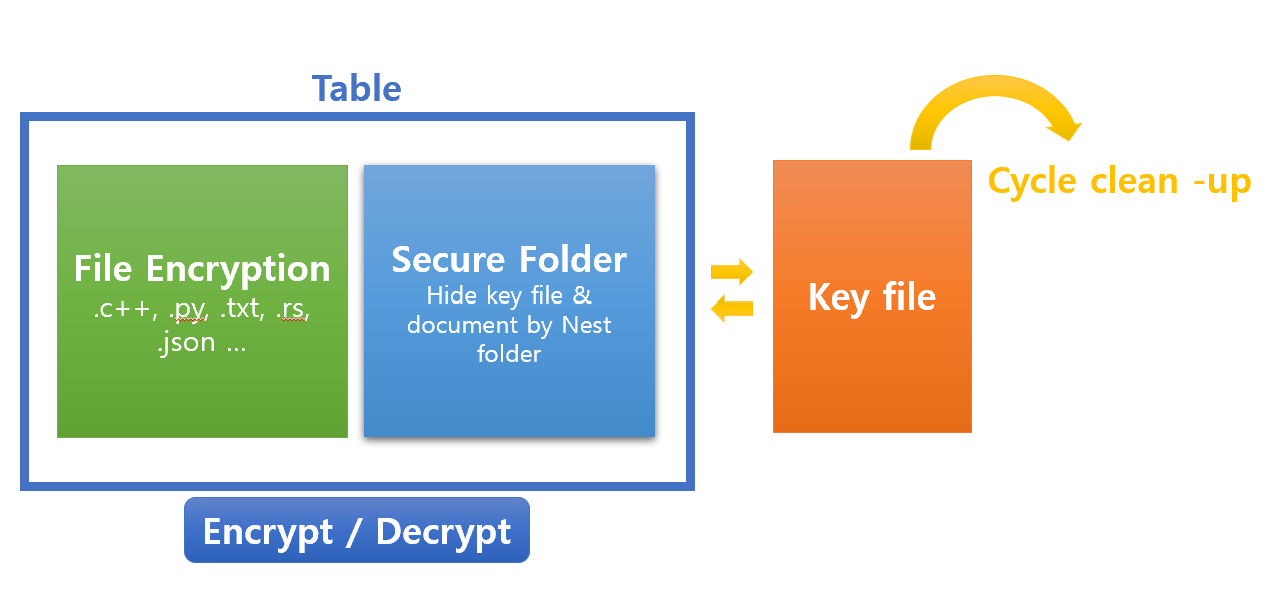


1. **Introduction**

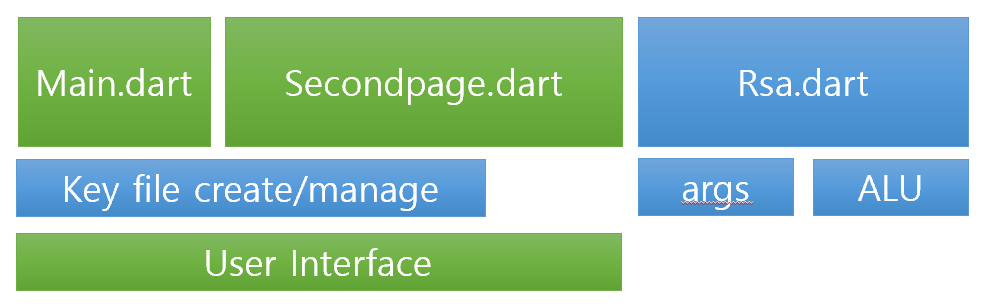
Kiwinest is software that encrypts and stores text files or codes in real time. Real-time encryption means that data is automatically encrypted just before it is stored and decrypted immediately after it is loaded without user intervention. If the correct key file is not used, the data encrypted cannot be read decrypted. Kiwinest is developed through the Flutter framework and can run on Windows, Mac and Linux environments.

■ This is a document for developers who contribute this project.

1. **Project Structure**

****

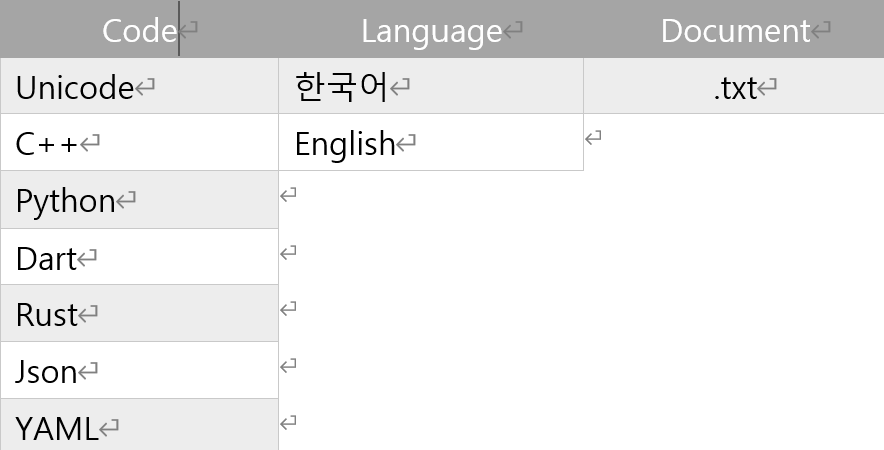
The Abstraction of this project is this. Encryption/decryption is the main function of this project, and both file encryption and security folder functions interact with users through tables. Internally, file encryption/cycle clean-up is operated through key files located in the document/Nest folder.



The abstraction above is the structure of a flutter program created to fit the project structure first described. **Main.dart** provides not only the initialization and build of the overall program, but also the file encryption function and the interaction through the slider. **Secondpage.dart** provides a secure folder function, only Fetch and Eject, but the creation of the Nest folder proceeds during the initialization process when building Main.dart.

**rsa.dart** is a library file that provides RSA operations on the two pages above, and is responsible for key argument generation and operation. Key file generation also takes place during the initialization process at the time of Main.dart build.

1. **Main functions**
   1. **Text / Code Encryption-Decryption**
2. Only the format specified must be encrypted. Only those formats can guarantee 100% integrity when decrypted.
3. Indentation or other symbols may not apply unless they are in the specified format.
4. Generally, 'filename\_encrypted' is generated during encryption, 'filename\_decrypted' is generated during decryption, and 'filename\_encrypted' is deleted, but the code can be modified according to the need for intuition.
   1. **Secure Folder**
5. Because file system control in Mac and Linux environments is not yet fully supported, Nest folders may be visible in certain distributions or versions.
6. In all OSs, Nest creation, Fetch, and Eject all operate mainly on Document files, so there are no special problems.
   1. **Cycle Clean-Up**
7. This feature is activated only when the program is turned on.
8. This is because Flutter does not support background in desktop environments. Therefore, in the current version, the timer runs when Kiwinest is running.
9. **Format**

**  
\* C file is currently not fully compatible**

1. **Notice & FAQ**

* Flutter doesn't support icon and favicon changes on Linux yet. For Macs, you'll need to make changes to all sizes directly customized. Windows is the only one that still displays all the icons completely.

**1. System Structure Architecture 시스템 상세 설계**

Kiwinest는 실시간으로 문서 또는 코드를 암호화하여 보관하는 소프트웨어다. 실시간 암호화란 사용자 개입 없이 데이터가 저장되기 직전에 자동으로 암호화되고 로드된 직후에 해독된다는 의미이다. 올바른 비밀번호/키 파일 또는 올바른 암호화 키를 사용하지 않으면 암호화된 볼륨에 저장된 데이터를 읽을(복호화할) 수 없다. Kiwinest는 Flutter 프레임워크를 통해 개발되며 Windows, Mac 환경에서 구동할 수 있다.

* 1. **RSA Algorithm RSA 알고리즘**

RSA 알고리즘은 Kiwinest의 주축 알고리즘으로 전자 서명, 디지털 뱅킹 등에 널리 쓰인다. 양자 컴퓨팅이 상용되지 않은 현재 가장 복잡한 알고리즘이지만, 알고리즘 코드 자체는 간단하며 그만큼의 유연성이 보장된다.

RSA는 두 개의 키를 사용한다. 여기서 키는 상수이다. 일반적으로 공개키(public key)는 모두에게 알려져 있으며 메시지를 암호화하는데 쓰이며, 암호화된 메시지는 개인키(private key)를 가진 자만이 복호화(decrypt)하여 열어볼 수 있다. 하지만 Kiwinest는 사용자가 PC 도메인에 귀속된 한 명 뿐이므로 모두 키 파일에 안전하게 보관된다.

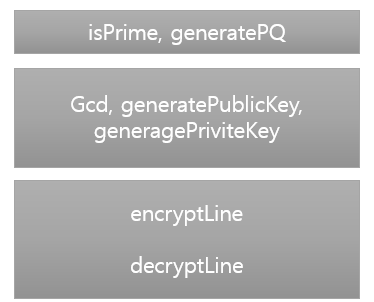
공개키 알고리즘은 누구나 어떤 메시지를 암호화할 수 있지만, 그것을 해독하여 열람할 수 있는 사람은 개인키를 지닌 단 한 사람만이 존재한다는 점에서 대칭키 알고리즘과 차이를 가진다. 이는 AES도 일반적인 암호화 솔루션에서 괜찮은 알고리즘이지만 RSA 또한 적합한 알고리즘임을 증명한다.

A와 B가 보안이 보장되어 있지 않은 환경에서 서로 비밀 메시지를 주고 받을 때, B가 A에게 메시지를 전달하기 위해서는 A의 공개키가 필요하다. A는 아래와 같은 방법을 통해 그 만의 공개키와 개인키를 제작한다.

p 와 q 라고 하는 두 개의 서로 다른 ( p ≠ q ) [소수](https://ko.wikipedia.org/wiki/%EC%86%8C%EC%88%98_(%EC%88%98%EB%A1%A0))를 고른다.

1. 두 수를 곱하여 를 찾는다.
2. φ ( N ) = ( p − 1 ) ( q − 1 ) 를 구한다.
3. φ ( N ) 보단 작고, φ ( N ) 와 서로소인 정수 *e*를 찾는다.
4. 유클리드 호제법을 이용하여 를 로 나누었을 때 나머지가 1인 정수 *d*를 구한다.
   * 여기서 공개키는 <N,e>, 개인키는 <N,d>로, 암호화 시에는 평문 M에 의 과정을 거친 c를 산출하고, 복호화 시에는 을 시행하여 다시 평문 M을 도출한다.

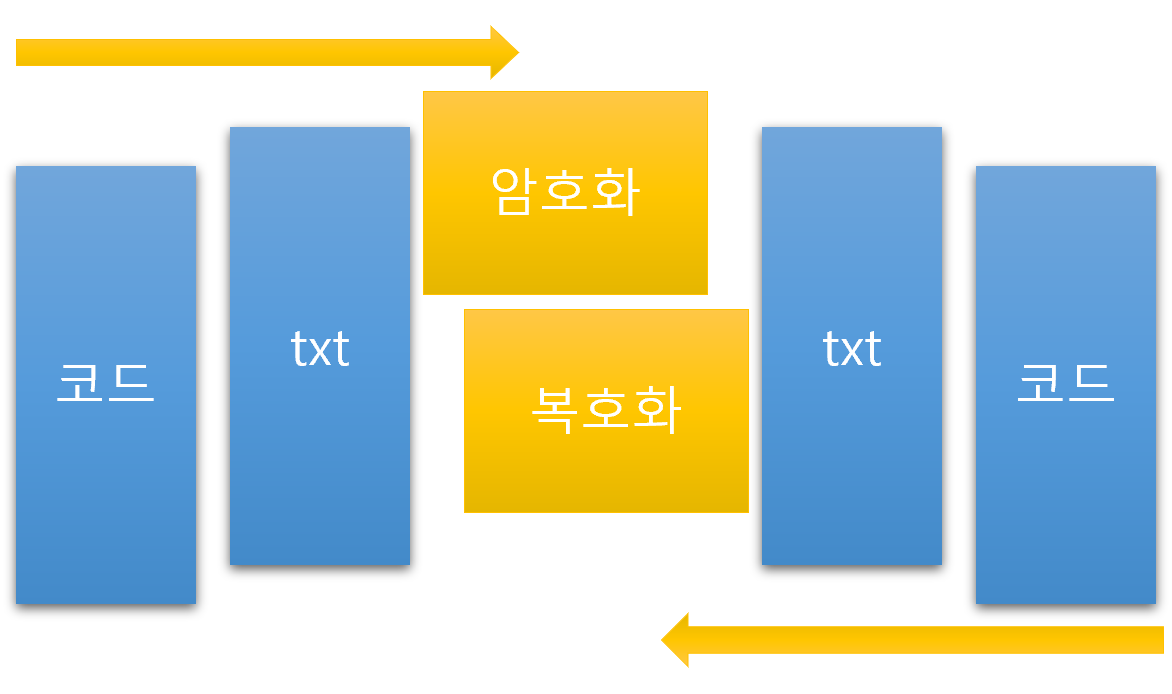
해당 과정들을 Kiwinest는 rsa.dart를 통해 제공한다.

 첫 번째 단에서는 p, q를 생성하고 생성과정에서 필요한 소수 판별 함수를 제공한다.

두 번째 단의 Gcd는 이후의 사용할 유클리드 호제법을 정의하고, generatePublickey는 말그대로 공개키를 생성하는 동시에(전술한 과정 2~3) 파이 값을 도출한다. generatePrivatekey는 gcd를 이용하여 d를 구해 개인키를 도출한다.

세 번째 단에서는 암호화/복호화를 진행하는 함수로 텍스트형 문서를 한 줄 한 줄 읽어들여 저장하는 역할을 한다. 해당 함수들은 Main.dart에서 import되어 사용된다.

* 1. **Main Functions 주요 기능**
     1. **File encryption / decryption 문서 암/복호화**

****

‘암호화’ 버튼을 눌러 파일 포맷(지원 되는 파일 포맷은 ‘2.4 지원 포맷’에서 후술)을 선택 하면 테이블에 암호화 되어있는 파일의 이름과 상태가 출력된다. 이후 테이블에서 암호화된 파일을 선택하면 이전에 비활성화 되어있던 ‘복호화‘ 버튼이 활성화 되고, 복호화 버튼을 누르면 파일은 원래대로 돌아가며 테이블에서 해당 파일의 이름이 삭제된다. 해당 과정은 Dart를 통해 개발된다(상세 구현은 ‘1.3 개발 도구 및 구현 참조’).

(위 시각자료 참고)암/복호화의 경우 코드들은 모두 txt ↔ 코드의 변환과정을 거친다. 코드를 암/복호화하면 txt로 변환된 다음 다시 코드로 변환되어 저장된다. 기존 지원하던 문서(docx, odt)의 경우 txt 변환과정이 불안정할 뿐더러 복호화 된 파일에 대한 무결성을 보장할 수 없어지기에 전술한 두 포맷은 부득이하게 삭제되었다.

* + 1. **Secure Directory 보안 디렉토리**

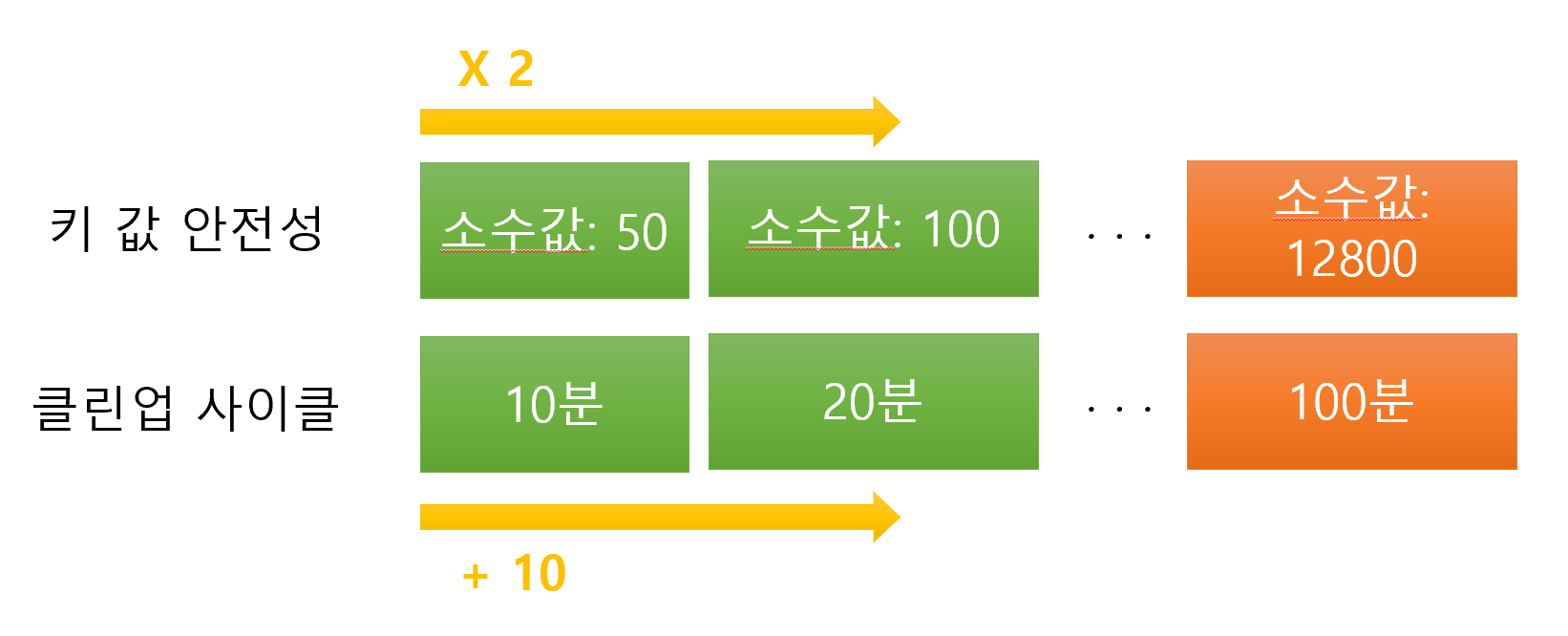
해당 기능은 문서 포맷의 부재로 전반적으로 빈약해진 프로그램의 효용성을 보완하기 위해 기존의 폴더 암호화에서 보안적인 측면을 부각하게 되었다. 먼저 처음 프로그램을 다운받거나 컴파일하면 각 OS의 ‘문서’ 폴더에 ‘Nest’ 폴더가 생성된다.

Nest 폴더는 Kiwinest에서 자체적으로 관리하는 폴더로 각 OS의 파일시스템에 맞추어 폴더에 암호를 걸고 그 암호를 RSA로 다시 암호화하여 관리한다. 그 이유는 OS별로 암호 알고리즘 및 시퀀스가 다르기 때문이다. 때문에 Nest 폴더는 외부에서는 열람이 불가능하고 오직 Kiwinest를 통해서만 관리할 수 있다. 또한 기존의 암호화, 복호화는 Fetch, Eject로 변경되며 이는 단순히 보안성을 높이고 싶은 객체를 Nest에 넣고 빼는 것이다. Nest에 들어간 파일들은 파일명, 파일 확장자, 날짜 순대로 데이터 셀에 저장된다. 또한 문서 암/복호화와 마찬가지로 Fetch된 객체는 테이블에 표기되고 Eject한 객체는 테이블에서 방출된다.

**1.2.3 Cycle clean-up 사이클 클린-업**

**** 우선 해당 기능을 설명하기 이전에 ‘키 파일’을 정의할 필요가 있다. 키 파일은 다음과 같은 추상화로 설계되며, 사용자의 설정 수준에 따른 n값까지의 많은 공개키와 개인키 쌍을 저장하는 파일이다. 이는 프로그램 다운로드 시 처음으로 생성되며, Rsa.dart와 사이클 클린-업 슬라이더를 통해 전적으로 관리된다. 키 파일은 Nest 폴더에 안전하게 저장된다.

사이클 클린-업은 키 파일의 공개키, 개인키들의 리스트의 인덱스를 일정 주기로 옮기면서 파일에 대한 강한 보안성을 보장하는 기능이다. 예를 들어 주기가 50분인 상태라면, 50분 안에 키 파일의 N1, e1, d1을 통해 파일 3개와 폴더 1개를 암호화하였다면, 다음 50분이 지난 상태에서는 N1, e1, d1을 통해 복호화 된 후 N2, e2, d2를 통해 다시 암호화된다. 해당 기능은 백그라운드에서 동작하도록 개발될 예정이며, 주기 시간은 컴퓨터의 전원이 켜져 있고 로그인한 상태에서 측정된다.

사이클과 키 값 안전성은 각 화면의 슬라이더를 통해 조정되는데, 둘 다 인수는 10가지이고 문서 암/복호화와 보안 디렉토리에서는 서로 독립적이다. 변경 값은 다음과 같다. 

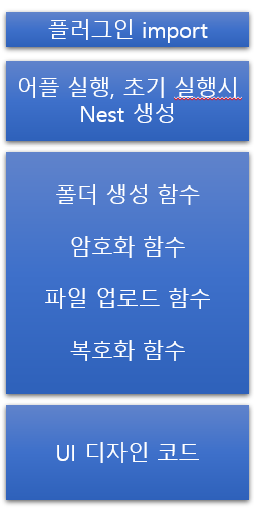
**2. Implement 구현**

Kiwinest는 Flutter를 통해 전 과정이 개발되며, 사용되는 외부 플러그인은 file\_picker이며, 나머지 플러그인으로는 flutter/material.dart, dart:io, dart:math 등이 사용된다. 본래에는 C 코드와 FFI를 통해 통신으로 구현할 예정이었으나, 통신을 할 때 많은 양의 텍스트가 주어지면 오버헤드가 발생할 확률이 높고, 클린 사이클을 적게 설정할수록 그 확률은 배가 되기에 안전하게 Dart를 통해 RSA 알고리즘을 구현하였다.

Main.dart에서는 초기의 Nest 폴더 생성과 파일의 암호화가 이루어지고 Secondpage.dart는 보안 디렉토리 기능이다. Rsa.dart는 크게 키 파일 생성/관리가 이루어지고, 미시적으로는 키 인수를 관리하며 Main.dart와 Secondpage.dart에서 암호화하는데 사용되는 연산 클래스를 정의한다.

보고서 작성일 기준 RSA 연산 구현과 암/복호화 시의 테이블 업로드, 초기 컴파일 시의 Nest 파일 생성이 완료되었다. 앞으로 남은 것은 키 파일 생성 및 상호작용, Nest 암호화, 코드 txt 변환과정과 사이클 클린업이다.

* 1. **Main.dart**

 본 목차에서는 보안 디렉토리 기능이 일부 구현된 Secondpage.dart는 Main.dart를 오버라이드 했기에 Main.dart 위주로 기술하였다.

Main.dart의 플러그인 import는 사용자와의 상호작용을 위해 filer\_picker 등의 편리성 플러그인 등을 차용한다. 또한 rsa.dart 등의 내부 구현 기능도 임포트한다.

두 번째 단에서는 Flutter에서 어플을 실행하는 main()이 있으며 초기 실행 시(컴파일 시) Nest를 생성하기 때문에 비동기 옵션이 추가된다.

세 번째 단에서는 폴더 생성 함수, 파일 업로드 함수와 함께 rsa.dart에서 불러온 기능들을 쓸 수 있도록 암호화, 복호화 함수가 있다. 파일 업로드 함수는 파일 암호화 시 테이블에 올릴 수 있도록 데이터 셀에 파일명을 슬라이스 해서 맞추는 역할 또한 시행한다.

네 번째 단은 UI 디자인 코드로 Flutter는 기능 구현과 디자인 모두 dart로 하는 특징 때문이다. 이 곳에서 세 번째 단의 함수들을 불러와 사용한다.

* 1. **Schedule Amendment 스케줄 변경사항**

Flutter로 개발하였을 때 docx 암호화 시의 문제, Mac에서의 파일 권한 충돌 등의 문제와 OS별 파일시스템의 문제로 인해 부득이하게 개발기간부터 연장되었으며(5/12 -> 5/26), 이에 맞춰 각종 문서들을 업데이트하는 데에도 시간이 걸릴 것이다. 또한 추후에 발생할 기여자를 고려하였을 때 한국어 문서보다 영어 문서가 나을 것이라는 확신을 하였고 릴리즈 단계에서 여태껏 써왔고 Github에 올라갈 전 문서를 영어로 수정할 예정이기 때문에 릴리즈 단계에서도 필요하다면 일정이 더 추가될 수 있다.

* 1. **Format 지원 포맷**

변경된 지원 포맷 목록

|  |  |  |
| --- | --- | --- |
| 코드 | 언어 | 문서 |
| C | 한국어 | .txt |
| C++ | English |
| Python |
| Dart |
| Rust |
| Json |
| YAML |