**GMMP Library**

**개발자 가이드**



제.개정 이력서

|  |  |  |
| --- | --- | --- |
| 개정번호 | 개정 페이지 및 내용 | 개정일자 |
| 1.01 | 환경설정 정보 추가 | 2015-01-16 |
|  |  |  |

목 차

[1. GMMP Library 구성 4](#_Toc409176750)

[2. 환경 설정 5](#_Toc409176751)

[2.1. Eclipse 설치 및 C/C++ Compiler 설치 5](#_Toc409176752)

[2.1.1. Java 설치 5](#_Toc409176753)

[2.1.2. Eclipse IDE for C/C++ Developers 설치 6](#_Toc409176754)

[2.1.3. Eclipse Classic + CDT Plug-in 설치 7](#_Toc409176755)

[2.2. 디버그 환경 설정 7](#_Toc409176756)

[2.2.1. gdb설치 7](#_Toc409176757)

[2.2.2. g++설치 7](#_Toc409176758)

[2.3. 프로젝트 연결 7](#_Toc409176759)

[2.4. Make 스크립트 10](#_Toc409176760)

[3. GMMP Library 구조 12](#_Toc409176761)

[3.1. GMMP Library 구조 설명 12](#_Toc409176762)

[3.1.1. Interface Module 12](#_Toc409176763)

[3.1.2. Command Module 12](#_Toc409176764)

[3.1.3. Network Module 13](#_Toc409176765)

[3.2. GMMP Library 파일 구조도 13](#_Toc409176766)

[4. GMMP Library 사용법 14](#_Toc409176767)

[4.1. 새로운 프로젝트 생성 14](#_Toc409176768)

[4.2. GMMP Library 연결 절차 15](#_Toc409176769)

[4.3. 컴파일 옵션 설정 16](#_Toc409176770)

[4.4. GMMP Library 연결 성정 판단 방법 18](#_Toc409176771)

[5. GMMP Library Sample 코드 18](#_Toc409176772)

[5.1. 초기화 18](#_Toc409176773)

[5.2. 등록 21](#_Toc409176774)

[5.3. 주기보고 25](#_Toc409176775)

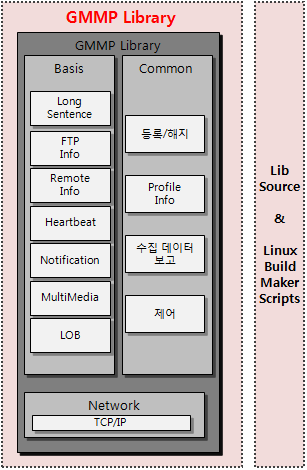
[5.4. TCP Always On 28](#_Toc409176776)

[5.5. 제어 & 제어 결과 보고 31](#_Toc409176777)

# GMMP Library 구성

GMMP Library는 SKT OMP와 M2M Gateway의 연동을 위해 사용되며 TCP 기반 연동 규격이다.

본 문서는 Gateway Library에 대한 가이드만 제공한다.



# 환경 설정

본 문서는 32bit 리눅스 기반에 Eclipse + C/C++ 개발툴에 대한 설명을 한다.

윈도우 환경/리눅스 콘솔환경에서의 구성은 제공 하지 않는다.

## Eclipse 설치 및 C/C++ Compiler 설치

### Java 설치

Eclipse 실행을 위해서는 JRE만 설치 해도 된다. 하지만 이후 Java 프로그래밍을 위해서는 JDK를 설치를 권장한다. 기존에 JRE, JDK가 설치 되어 있다면 설치 하지 않아도 된다.

#### JRE 설치

<http://www.oracle.com/technetwork/java/javase/downloads/index.html?ssSourceSiteId=ocomen> 에 접속하여 OS 환경에 맞게 JRE를 설치한다.



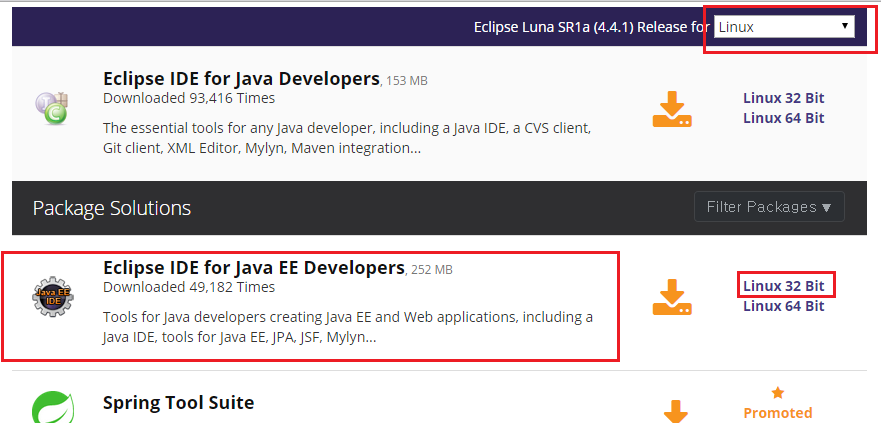
#### JDK 설치

<http://www.oracle.com/technetwork/java/javase/downloads/index.html?ssSourceSiteId=ocomen>에 접속하여 OS 환경에 맞게 JDK 설치한다.



### Eclipse IDE for C/C++ Developers 설치

<http://www.eclipse.org/downloads/> 접속하여 [Eclipse IDE for C/C++ Developers](http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr1)를 설치한다



### Eclipse Classic + CDT Plug-in 설치

* <http://www.eclipse.org/downloads/> 접속하여 [Eclipse Classic](http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr1) 를 설치한다.
* Eclipse를 실행 하고 Help -> Install new Software 선택한다.
* Add를 선택하고 아래에 내용으로 검색한다.
* **Name : CDT**
  + - **Location : http://download.eclipse.org/tools/cdt/releases/galileo**
* 검색된 CDT Main Features와 CDT Optional Features을 설치한다.
* Eclipse를 Restart 한다.

## 디버그 환경 설정

### gdb설치

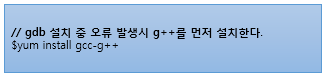
콘솔 기반 디버깅 툴.

gdb가 설치되어 있지 않다면 아래와 같은 순서로 gdb를 설치한다.



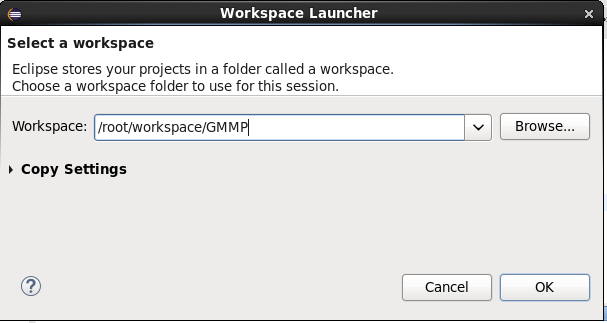
### g++설치

**컴파일도중 g++에러가 발생하는 경우 g++를 설치한다.**

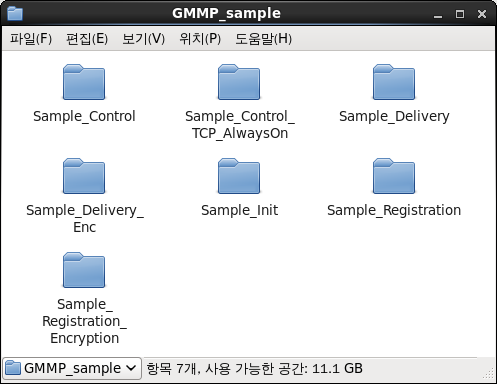


## 프로젝트 연결

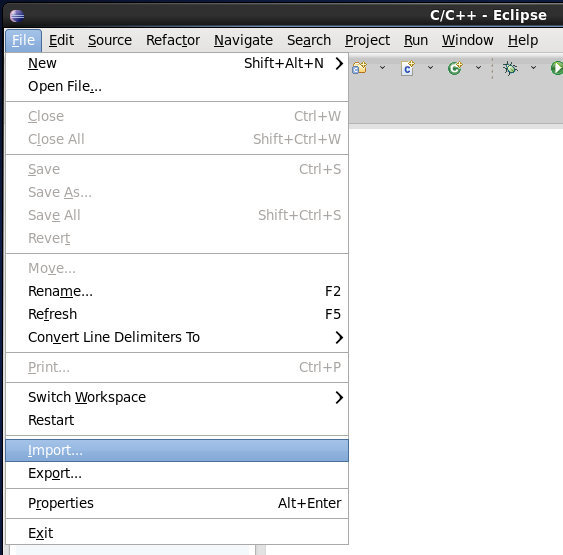
* **Eclipse 를 실행한다.**
* **Workspace 를 설정한다.**



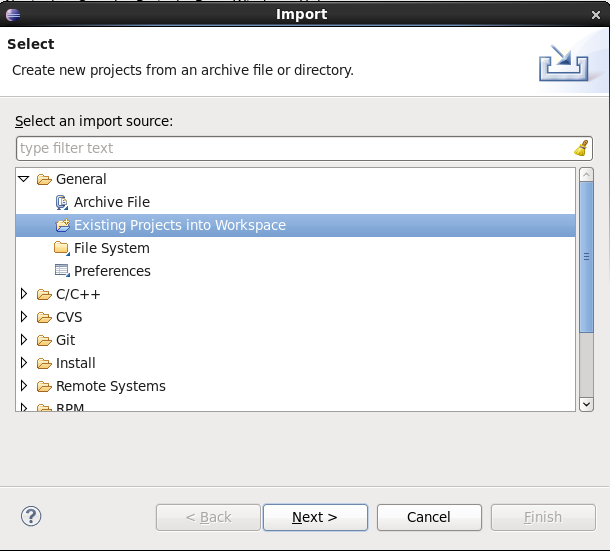
* **설정한 경로(“/root/workspace/GMMP”)에 GMMP\_sample.tar.gz 파일을 복사한다.**
* **복사한 폴더에 GMMP\_sample.tar.gz 파일을 압축 해제 한다.**



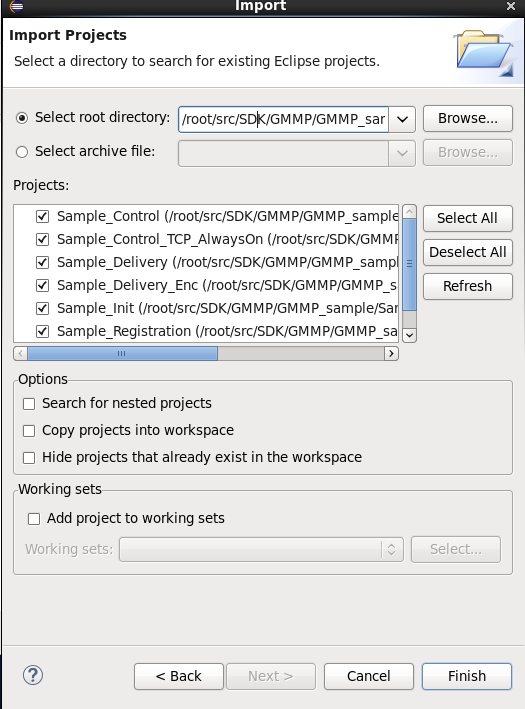
* **eclipse에서 파일을 import한다**

****

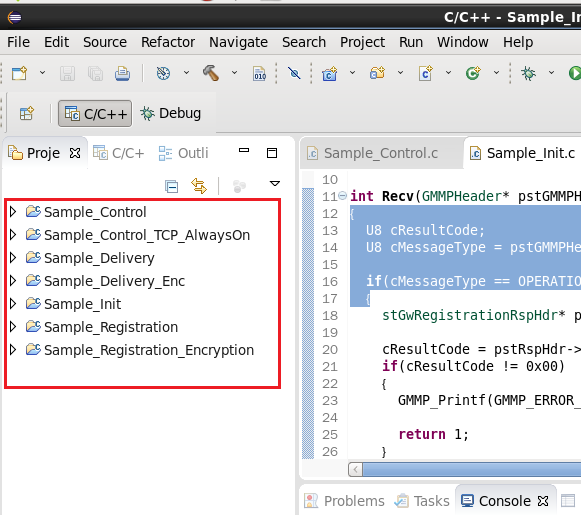
* **General -> Existing Projects into Workspace를 선택한다.**



* **GMMP\_sample.tar.gz 압축 해제한 경로를 지정한다. (“/home/workspace/GMMP/GMMP\_sample”)**



* **프로젝트 로딩 화면**

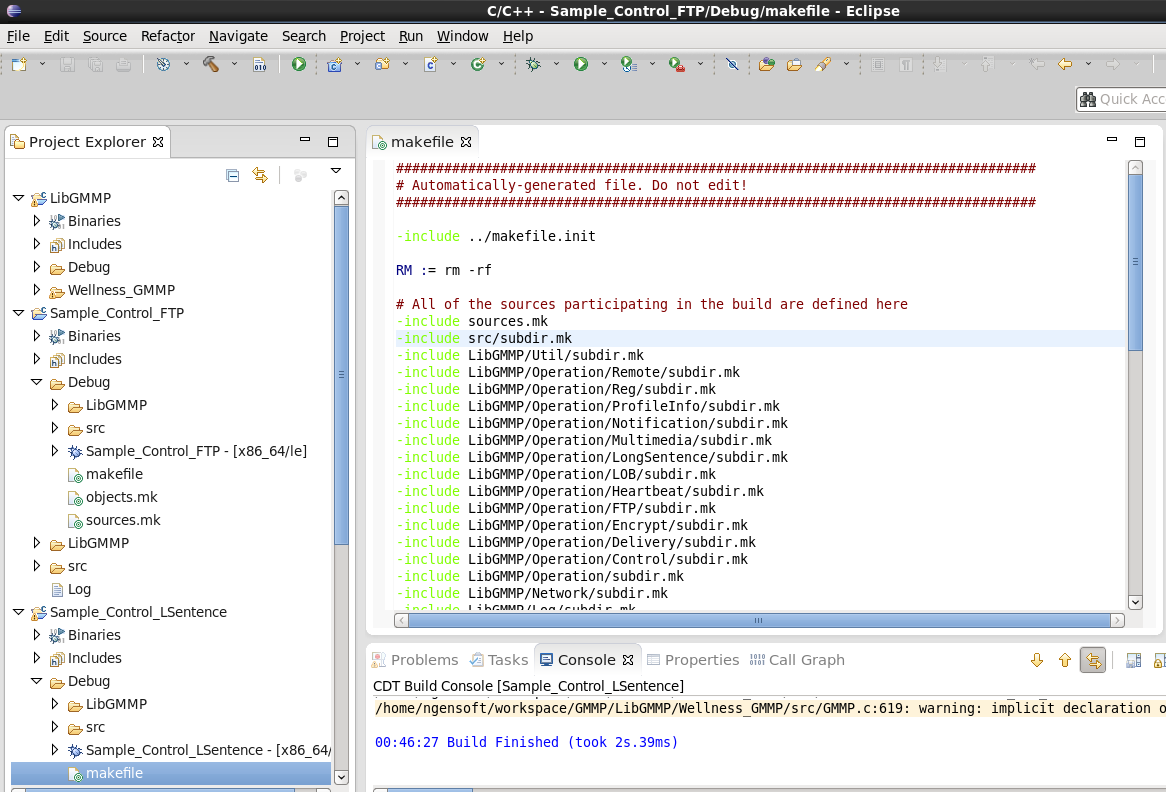


## Make 스크립트

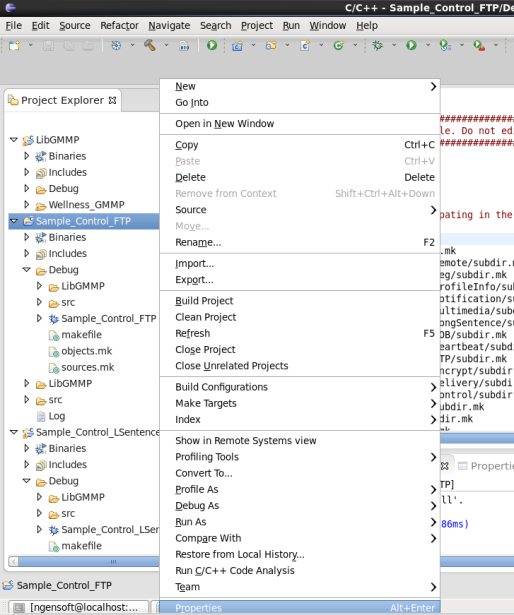
프로젝트의 Debug 폴더에 makefile은 자동 생성이 되어 사용자가 수정 할 수 없다.

makefile 수정을 해야 할 경우에는 프로젝트 옵션을 통해 수정 할 수 있다.

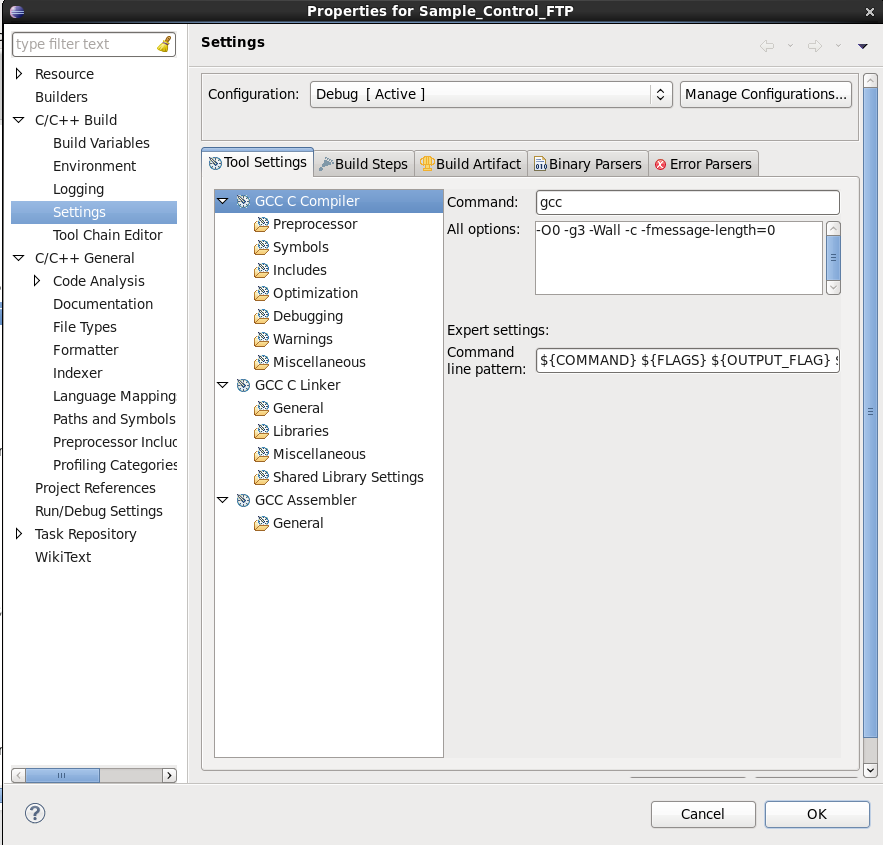
* **makefile의 위치**



* **컴파일 옵션 변경하는 방법은 프로젝트를 선택하고 오른쪽 마우스 클릭 -> Properties를 선택한다.**



* **C/C++ Build와 C/C++ General 을 통해 옵션을 변경할 수 있다.**



# GMMP Library 구조

## GMMP Library 구조 설명

GMMP Library는 크게 3가지로 구분이 된다.

Library를 사용하기 위한 Interface Module, Embedded 시스템에 Porting을 위한 Command Module과 OMP와 통신을 위한 Network모듈로 구분이 된다.

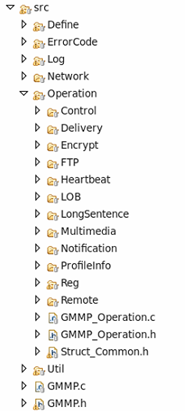
### Interface Module

Library 사용자를 위해 제공되는 Module로 GMMP의 각 Operation에 해당하는 기능들을 쉽게 사용하기 위해 구성되어져 있으며, Library 소스에서 GMMP.h, GMMP.c 파일에 해당 한다.

### Command Module

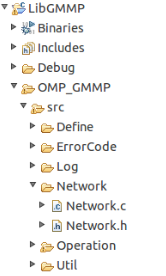
기능을 수행하기 위한 Structure 에 직접 데이터를 담아 TCP 패킷을 구성하는 모듈이다.

각 Operation별 디렉토리 구조에 기능이 구현되어 있으므로 Embedded 시스템에 Porting 시 해당 구조에데이터 입력 부분만 개발하여 연결하여 사용이 가능하다. 구조체에 데이터 입력 부분은 GMMP\_Operation.c GMMP\_Operation.h를 기준으로 작성하면 된다.



### Network Module

Command Module에 의해 저장된 Structure 데이터를 OMP 서버에 통신으로 직접 전송하는 모듈이다.

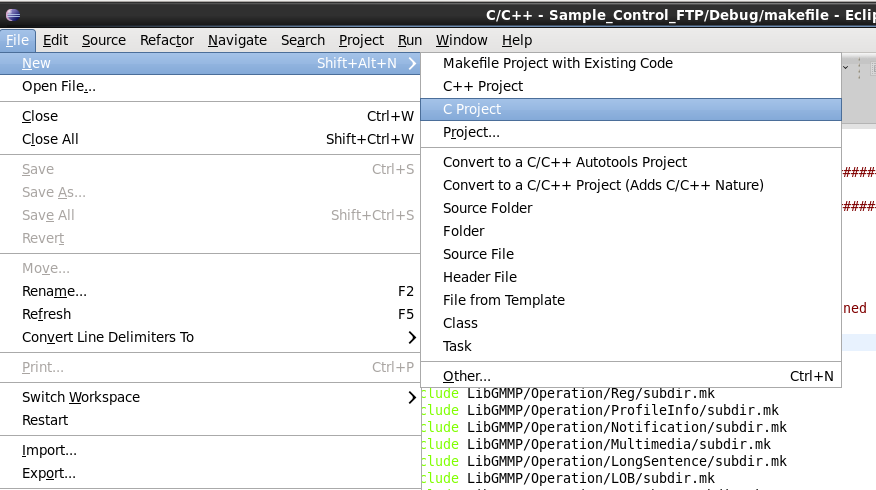


## GMMP Library 파일 구조도

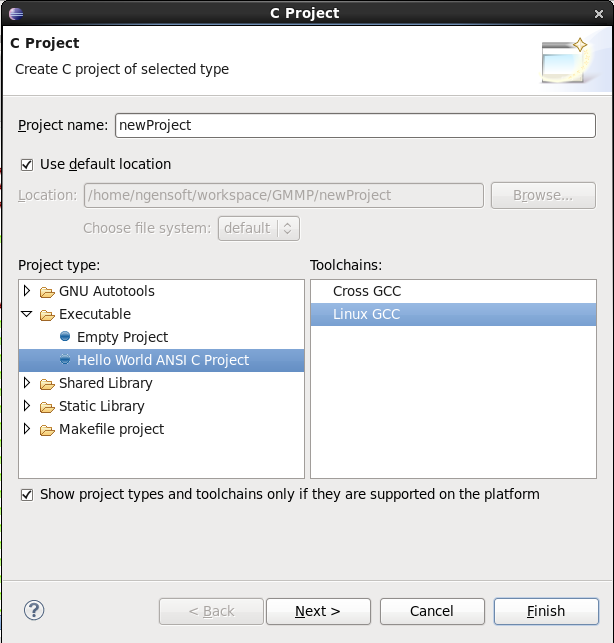
# GMMP Library 사용법

## 새로운 프로젝트 생성

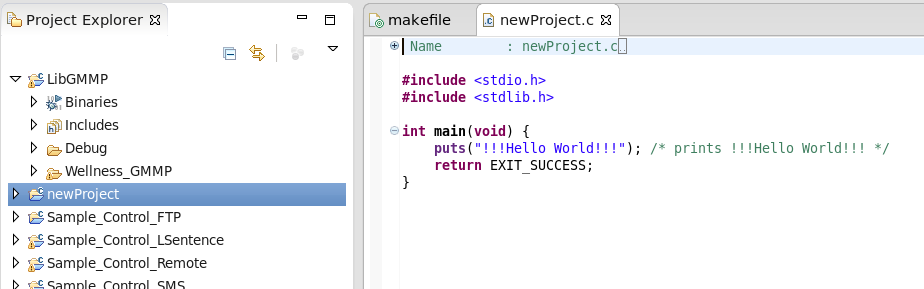
* **신규 프로젝트 생성**



* **프로젝트 생성 옵션**

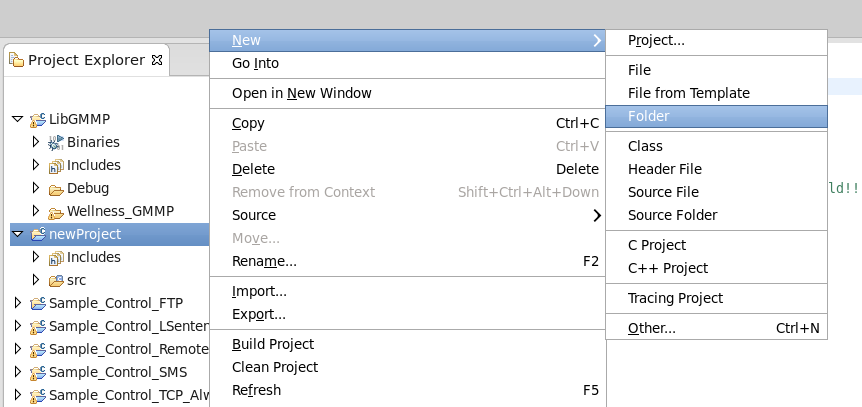


* **신규 프로젝트 생성 화면**

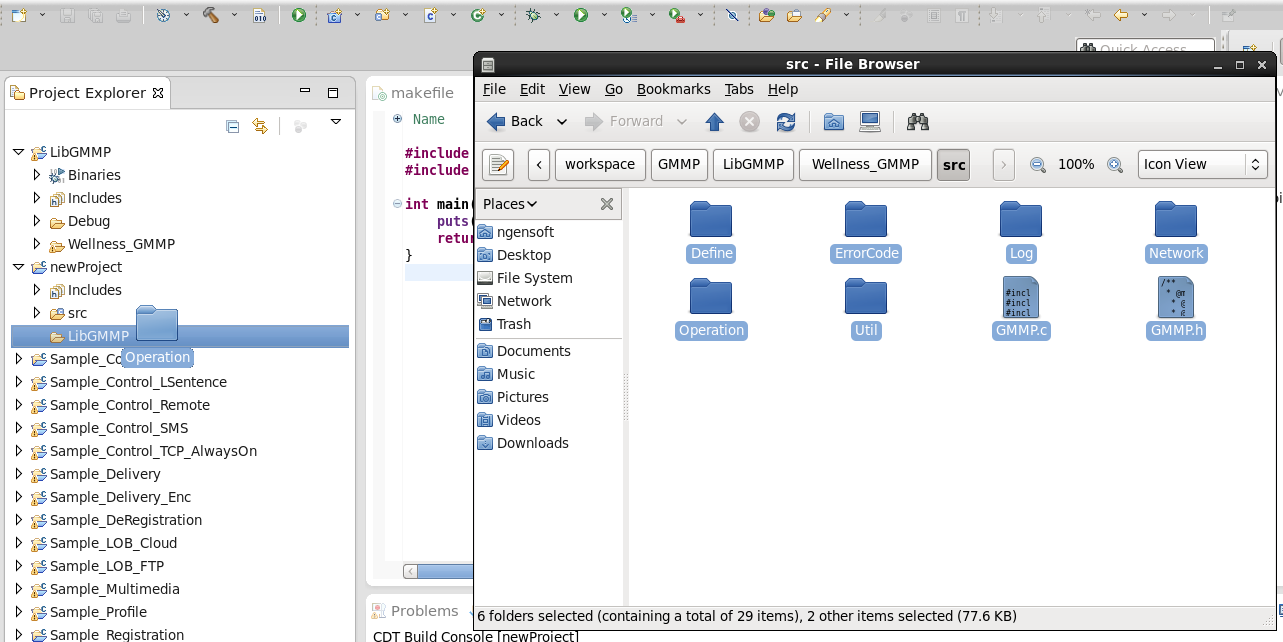


## GMMP Library 연결 절차

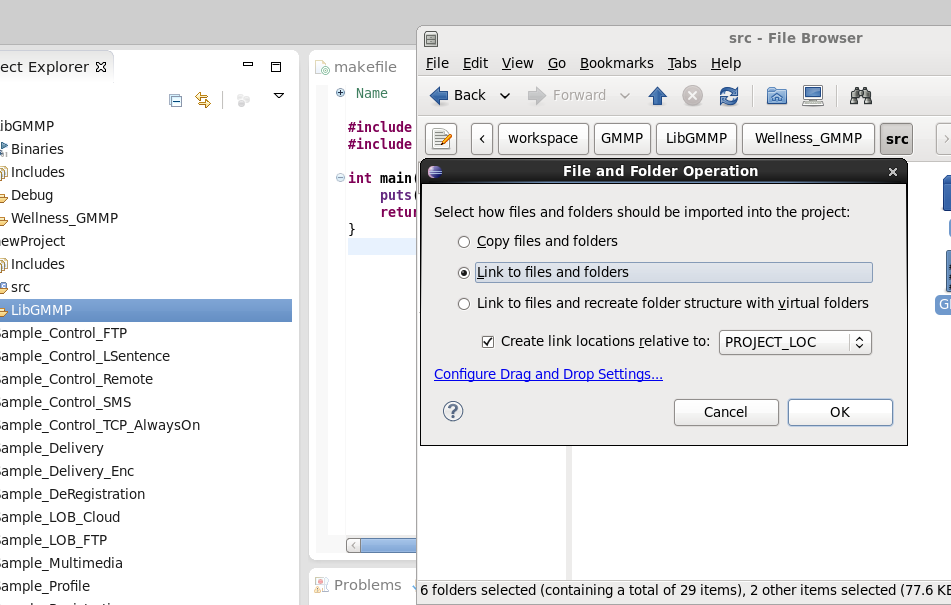
* **GMMP Library 연결을 위한 디렉토리 생성**



* **생성된 디렉토리에 Library 소스 파일 연결**



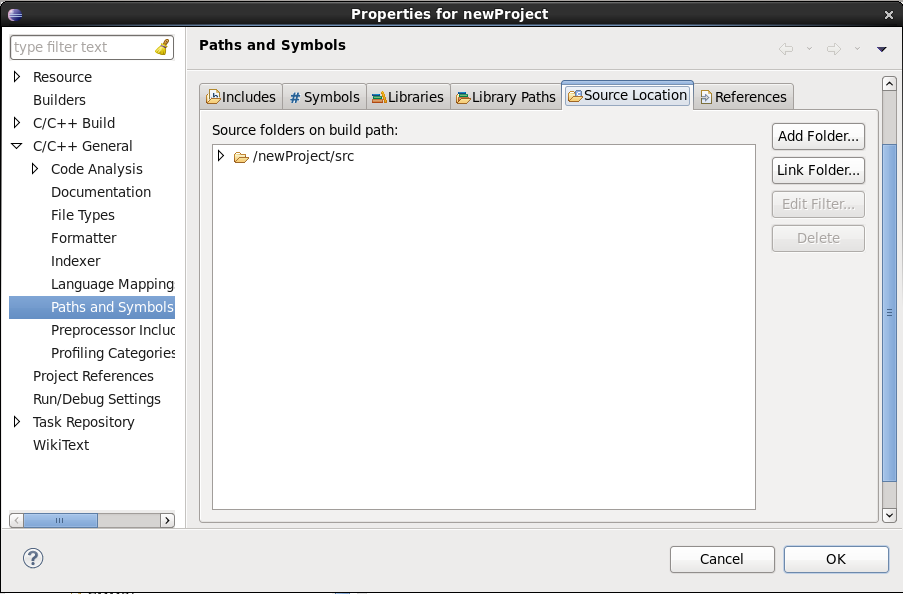
* **Library 연결 링크 설정 ( 소스파일을 복사 하지 않고 링크 설정을 통해 makefile에 Library 소스 위치를 등록하는 절차이다. 소스파일일 복사하여 사용해도 무관하다.)**



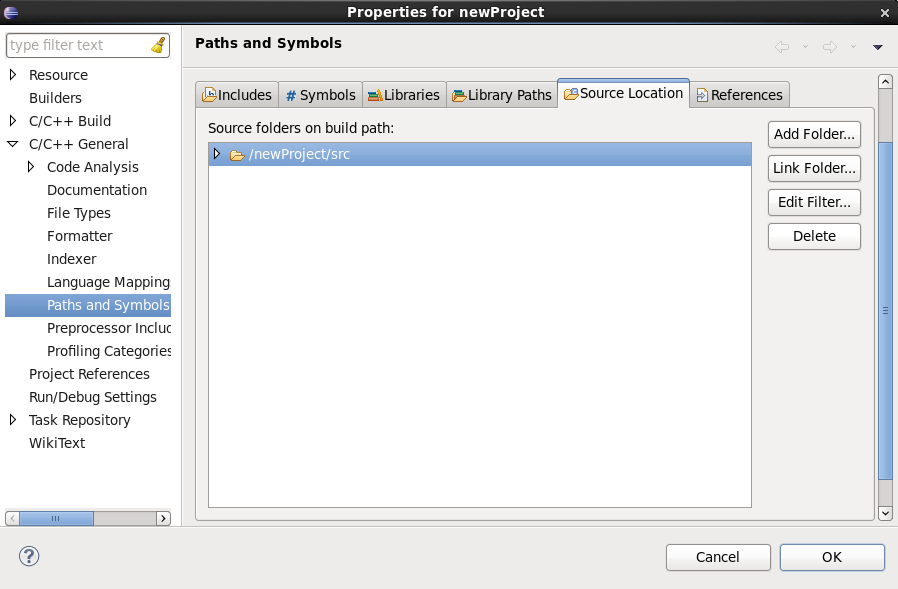
## 컴파일 옵션 설정

디폴트로 src 경로까지 범위로 잡혀있는걸 GMMP\_lib경로를 포함하기위해 프로젝트경로 전체로 변경하는 절차이다.

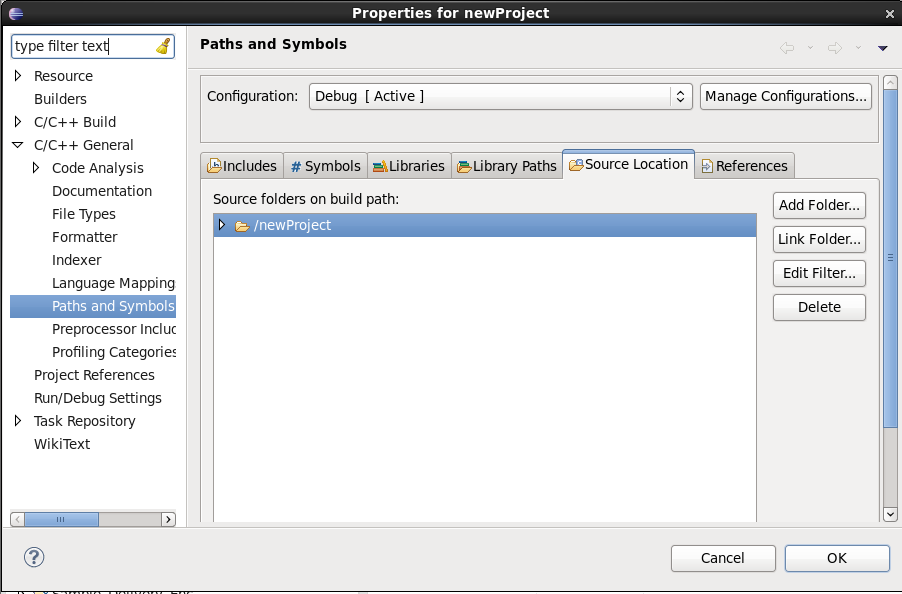
* **컴파일 옵션 선택 -> C/C++ General -> Path and Symbols -> Source Location**



* **컴파일 범위 경로 변경 ( /newProject/src -> /newProject 변경)**



* **컴파일 범위 경로 변경 화면**



## GMMP Library 연결 성정 판단 방법

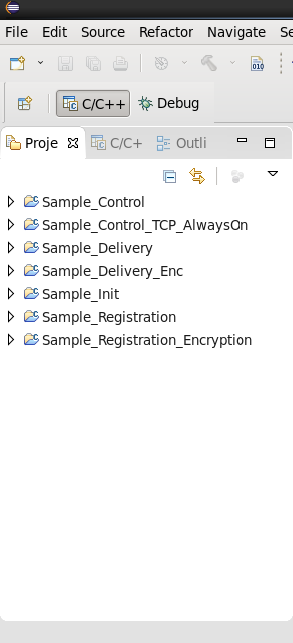
샘플 코드 중 Sample\_Registration의 main.c 코드를 새로 생성한 프로젝트의 main.c에 입력하여 컴파일을 해본다. 문제 없이 빌드가 된다면 정상적으로 Library를 사용할 수 있다.

# GMMP Library Sample 코드

GMMP Library는 Linux CentOS 환경에서 사용이 가능하다.

※ Sample 코드에 적용된 domain, serviceName 등의 값들은 테스트코드에 사용된 값으로 기능 동작에 대한 책임은 지지 않는다. 해당 값들은 SKT OMP Portal을 통해 등록 절차가 진행이 되어야만 한다.

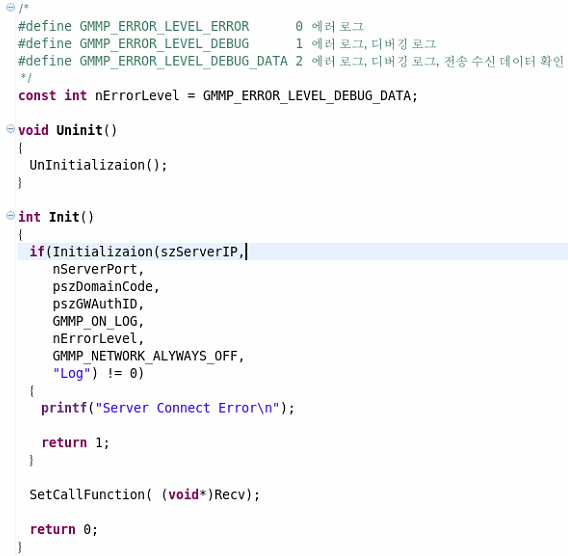
GMMP Library 소스를 링크하여 샘플 소스가 제작되어 샘플안에 GMMP Library 소스가 포함된 파일로 압축되어 있고 기능별로 프로젝트를 생성하였다.



## 초기화

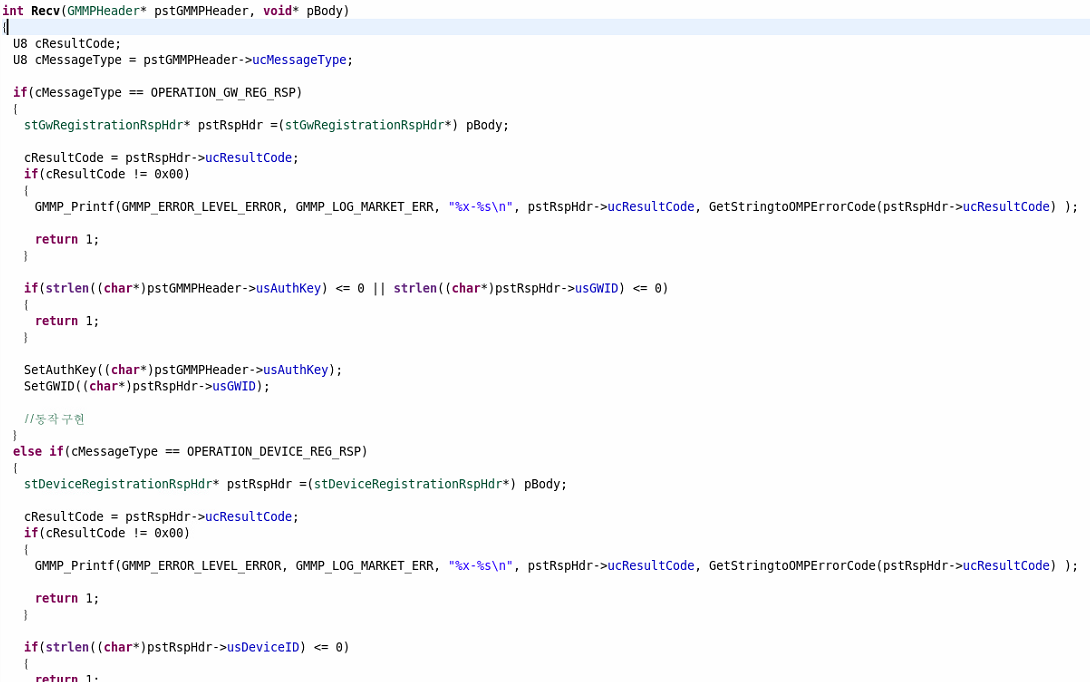
Server IP, Server Port, Domain Code, Auth ID, 로그 레벨 등을 입력절차와 OMP로부터 응답받을 콜백 함수를 등록 절차로 이루어 진다. ( SetCalFunction 함수 참조.)

입력된 값은 Library 내에서 관리하며 이후 샘플 설명에서는 생략하도록 한다. 자세한 내용은 Sample\_Init 프로젝트를 참고한다.



콜백 함수는 GMMP Header 포인터와 Body포인터를 제공하며 Body포인터를 GMMP Header의 MessageType에 맞게 캐스팅하여 사용을 해야 하며 캐스팅된 Body의 ResultCode의 값이 0x00 또는 STATUS\_OK 이 아니면 OMP 서버로부터 에러 코드를 제공 받은것이다.

OMP 서버로부터 에러 코드는 GetStringtoOMPErrorCode(pstRspHdr->ucResultCode) 함수를 사용하여 에러 코드를 문자열로 변환하여 제공하며, Library의 에러 코드는 GetStringtoErrorCode(nErrorCode)를 사용하여 문자열로 변환 할 수 있다.

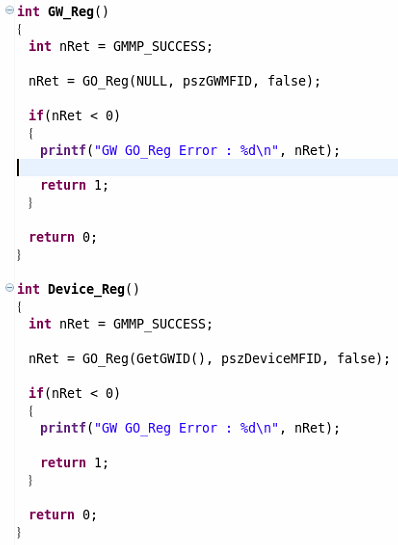


## 등록

GW를 등록할 때는 M2M OMP 포탈의 [단말 관리] 메뉴에서 등록 신청을 하고 승인을 받은 상태여야만 가능하다.

GW/Device 등록은 GW\_Reg()함수를 통해 구동이 되며 등록된 Callback 함수를 통해 결과를 제공한다.

자세한 내용은 Sample\_Registration, Sample\_Registration\_Encryption(암호화) 샘플 프로젝트와 GMMP API 문서를 참고한다.

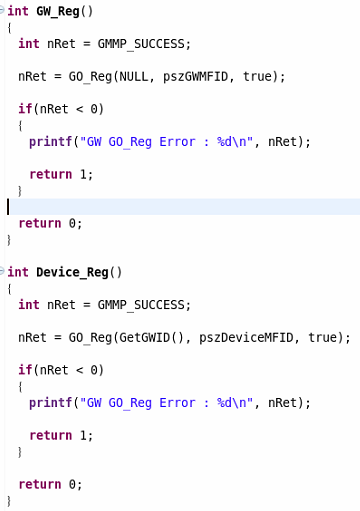


OMP 서버로부터 제공받은 GW ID와 AuthKey를 Library에 등록하여 사용 해야한다.



주기보고의 데이터가 암호화 되어 전달되기 원하는 서비스에 대해서 OMP 서버에 암호화 설정이 되어 있다면 GO\_Reg의 함수의 마지막 파라메타의 암호화 사용유무에 대한 판단 기능 Flag를 true로 하여 함수를 호출 하여야 한다.

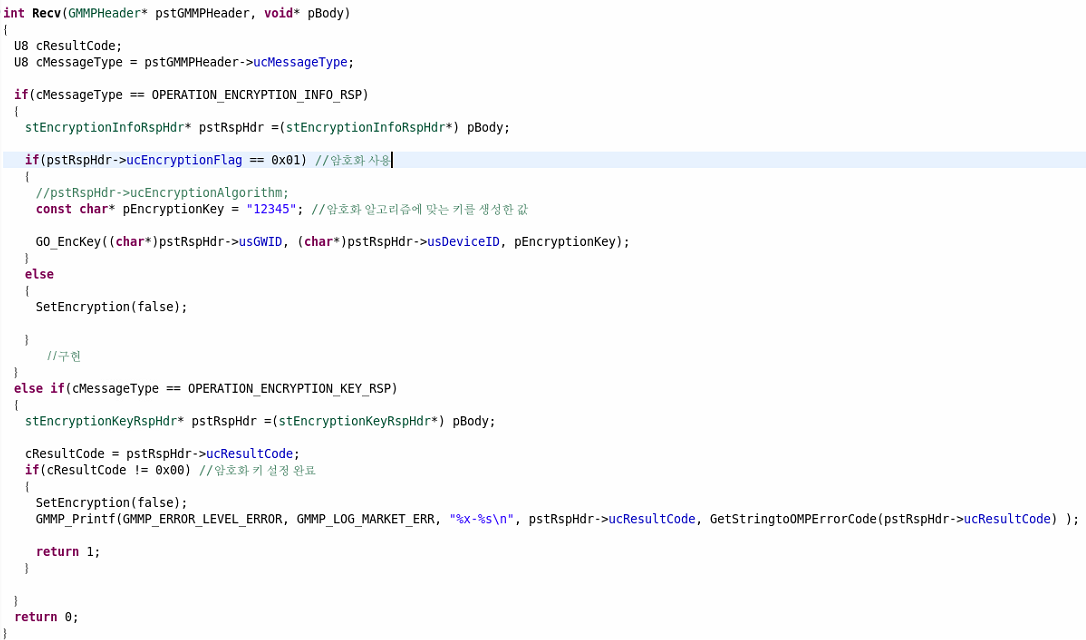
암호화 사용 Flag가 true이면 GO\_EncInfo 함수를 통해 Library에서 암호화 사용유무를 OMP 서버로부터 요청을 한다. 암호화 사용 조회는 개발자가 직접 사용하여 처리 할 수 있다.



OMP 서버로부터 제공받은 EncryptionFlag 가 0x01이면 암호화 기능을 사용하는 것으로 EncryptionAlgorith 변수에 OMP서버에 등록된 암호화 알고리즘 정보를 제공받는다.

암호화 알고리즘은 AES(128,192,256), SEED(128,256)을 지원하며 암호화 수행 시 사용하는 키는 GO\_EncKey 함수를 이용하여 OMP 에 설정할 수 있다.

암호화 사용을 안 할 경우 SetEncryption(0)을 수행해야 한다.



## 주기보고

GW/Device 등록 절차를 통해 등록된 GW/Device의 수집된 정보를 OMP서버에 전송할 때 사용된다.

수집 데이터 보고는 4종류로 각각 ‘수집 데이터 / 이벤트 데이터 / 장애 데이터 / 장애 해제 데이터’이다.

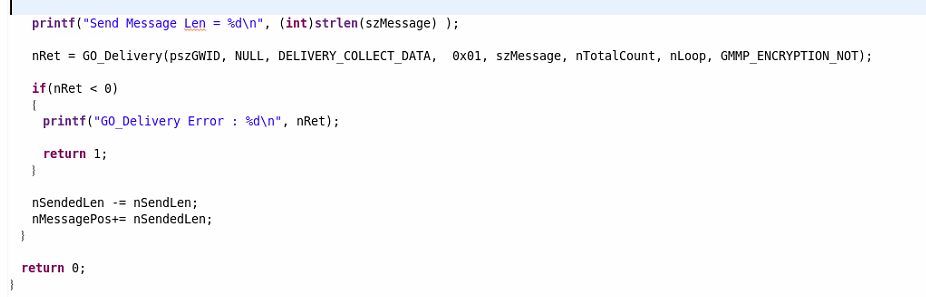
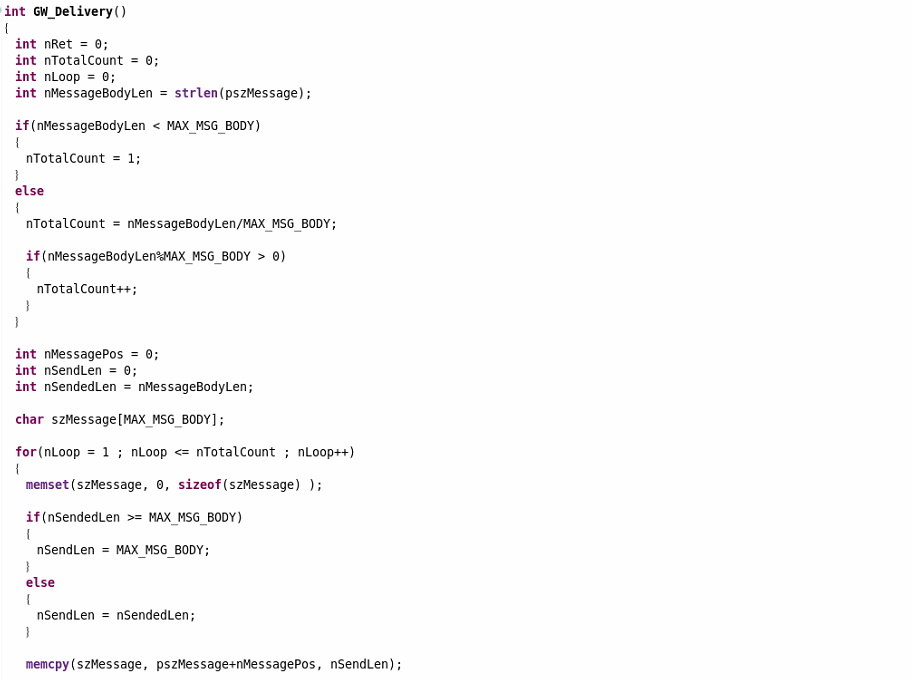
MessageBody 전송 데이터의 최대 길이는 2048byte보다 작아야 한다.

2048byte보다 큰 데이터는 아래 샘플 소스처럼 직접 2048byte 크기만큼 나누어 전송하여야 한다.

주기보고 데이터를 암호화 하여 전송할 경우 MessageBody의 처음 2byte는 암호화 이전의 원본 데이터 길이의 값을 설정(short형, Big Endian)하고 3byte부터 주기보고 데이터를 사용한다.

Library는 GMMP Header의 GMMP Message Length 필드 값은 데이터가 암호화 되면 길이가 변하므로 변경된 길이 값으로 재설정을 한다.

자세한 내용은 Sample\_Delivery, Sample\_Delivery\_Enc(암호화) 프로젝트와 GMMP API 문서를 참고한다.



기본 샘플





암호화 샘플

## TCP Always On

TCP Always On은 TCP 통신 방식의 하나로 하나의 socket 세션을 이용하여 OMP 서버와 세션을 유지하여 통신하는 방식이다.

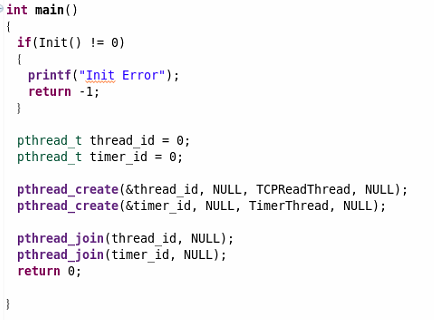
세션을 유지 하기위해서 Heartbeat 기능을 사용을 하며 OMP 서버에 등록된 Heartbeat 주기 시간이 넘어가면 OMP 서버에서 자동으로 세션이 종료 시킨다.

Profile 기능을 통해 OMP 서버로부터 제공 받은 Heartbeat 정보를 설정하여 Heartbeat 주기를 설정 할 수 있다.

TCP Always On 방식은 Thread를 생성하여 Thread에서 socket 수신을 처리해야 한다.

Thread 생성은 CentOS 기준으로 처리 되어 있다.

자세한 내용은 Sample\_Control\_TCP\_AlwaysOn 프로젝트를 참고한다.

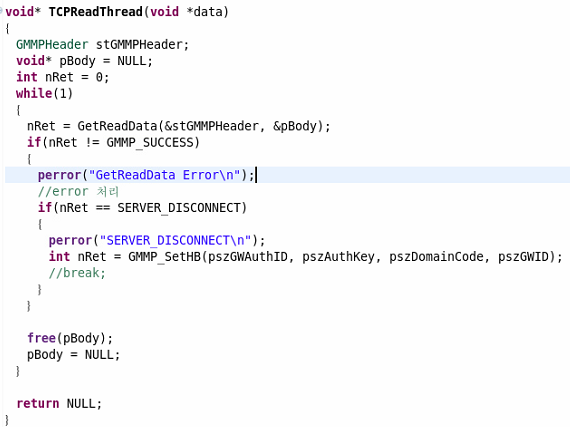


OMP 서버에 등록된 Heartbeat 주기 시간에 맞춰 Heartbeat를 OMP 서버에 전송하여야 한다.

샘플 코드에는 Heartbeat 주기를 Thread로 처리하였고 Sleep 함수를 통해 주기 시간을 설정하였다.



Socket 수신 Thread에서 OMP 서버와 Disconnect이 되더라도 Library에서는 각 기능(등록/해지/Profile 등)에서 초기화에 등록된 서버IP, Port를 통해 세션을 연결하여 기능을 수행한다.



## 제어 & 제어 결과 보고

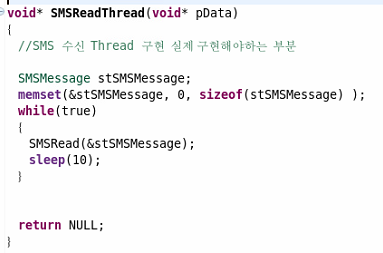
제어 수신은 TCP Always On 방식과 SMS 방식을 통해 가능하다.

TCP Always On 방식은 7.6절에 설명과 동일 하게 TCP Always On 방식은 Thread를 생성하여 Thread에서 socket 수신을 처리해야 한다. ( Recv socket Thread 생성 생략)

SMS 방식은 SMS 수신 Thread를 생성하여 처리하고 수신 데이터 구조는 SMS데이터를 직접 파싱하여 SMSMessage 구조체에 맞춰 데이터를 입력하여 처리 하여야 한다.

TCP Always On 방식과 SMS 방식은 수신 받은 방법만 다르고 수신에 대한 응답과 제어 결과에 대한 보고는 동일하다.

자세한 내용은 Sample\_Control 프로젝트와 GMMP API 문서를 참고한다.



제어 명령을 수신을 받게 되면 제어 명령에 대한 응답을 전송하고(GO\_Control 함수 참조) 제어 명령 기능을 수행 후 제어 결과 완료 보고(GO\_Notification)를 하면 된다.

제어 수신시 OMP로부터 제공받은 TID를 사용하여 GO\_Control, GO\_Notification을 수행하면 된다.



controlProcess에서 제어명령에 대한 세부 로직를 처리 한다..