

# **Ubists U9 BT Reader**

**[Android]**

## **Library Reference**

ISSUE DATE : 2011.07.18

---

---

## 목 차

<b>CHAPTER 1. INTRODUCTION .....</b>	<b>4</b>
1.1 개요.....	4
1.2 UBISTS U9 BT FOR ANDROID API 의 구성 .....	4
1.3 UBISTS U9 BT FOR ANDROID API 의 개발 도구.....	4
<b>CHAPTER 2. INSTALLATION .....</b>	<b>5</b>
2.1 JNI 추가를 위한 환경설정 .....	5
2.1 .....	9
2.2 .....	9
2.3 샘플 소스 코드 실행 .....	9
2.4 API 클래스 이용 .....	9
<b>CHAPTER 3. FUNCTION SPECIFICATION.....</b>	<b>10</b>
3.1 CONNECTREADER.....	10
3.2 DISCONNECTREADER.....	10
3.3 GETBLUETOOTHSTATE.....	10
3.4 INITIALIZE.....	10
3.5 WRITERREGISTER .....	11
3.6 READERREGISTER.....	11
3.7 BEGINREAD .....	11
3.8 FINISHREAD.....	11
3.9 GETTAGLIST .....	12
3.10 GETCOUNTLIST .....	12
3.12 GETTAGLISTSIZE.....	12
3.13 GETTAGBYINDEX .....	12
3.14 GETCOUNTBYINDEX .....	12
3.15 GETRSSIBYINDEX .....	13
3.16 GETRSSILEVEL.....	13
3.17 GETREADTIME .....	13
3.18 GETREADTYPEBYINDEX .....	13
3.19 CLEARTAGLIST .....	14
3.20 ISUPDATE.....	14
3.21 SENDSTRING.....	14
3.22 GETREADERVERSION .....	14
3.23 GETLIBVERSION.....	14

---

3.24	READTAGMEMORY .....	14
3.25	WRITETAGMEMORY .....	15
3.26	LOCKTAG .....	15
3.27	KILLTAG .....	16
3.28	SETMEMORYBANK .....	17
3.29	SETGEN2QVALUE.....	17
3.30	SETRfPOWERATTENUATION .....	17
3.31	GETMEMORYBANK .....	18
3.32	GETGEN2QVALUE .....	18
3.33	SETRfPOWERATTENUATION .....	18
3.34	GETREADERLIVE .....	18
3.35	SETREADERLIVETIMEOUT.....	18
3.36	SAVREGISTER .....	18
3.37	DEFAULTREGISTER .....	19

---

## Chapter 1. Introduction

### 1.1 개요

본 문서는 Ubists U9 Bluetooth 리더와 Android 스마트폰을 을 기반으로 하여 개발하고자 하는 사용자를 위한 Library Reference 입니다. 본 문서는 사용자의 PC 가 Window 기반의 OS 가 설치된 PC 이며, Android SDK 가 이미 설치 되어 있고 Android SDK 를 Eclipse 기반 하에 개발한다고 가정하고 설명 합니다.

### 1.2 UBISTS U9 BT for Android API 의 구성

UBISTS U9 BT for Android API 는 다음과 같은 파일들로 구성되어 있습니다.

- libubists.so  
: Ubists 리더기와 Bluetooth 를 통하여 통신하는 데이터를 처리하는 JNI Library 파일
- ConnectSocket.java  
: libubists.so 를 이용할 수 있게 해주는 Wrapper class
- KR900\_LCD\_SAMPLE  
: libubists.so 파일과 ConnectSocket.java 를 이용하여 동작하는 데모 프로그램

### 1.3 UBISTS U9 BT for Android API 의 개발 도구

- Android API 8, revision 1 혹은 동등 이상의 호환성을 가진 google 공식 SDK 배포본
- Eclipse 3.5.2 혹은 호환성을 가진 상위 버전

#### ※ 주의 사항

1. 본 문서에서 설명하는 API는 Android SDK를 Eclipse 환경에서 수행되는 Library입니다. 따라서 해당 Library를 다른 개발 환경에서 수행할 경우 정상적으로 동작되지 않을 수 있습니다.
2. libubists.so는 Android Platform SDK API 8을 이용하여 만들어진 Android용 JNI(Java Native Interface)용 함수를 포함하여 제작되었습니다
3. Sample Source 는 라이브러리 함수 테스트용으로 만든 프로그램이므로 참고 및 기본 동작 테스트용의 기능만을 가지고 있습니다.

## Chapter 2. Installation

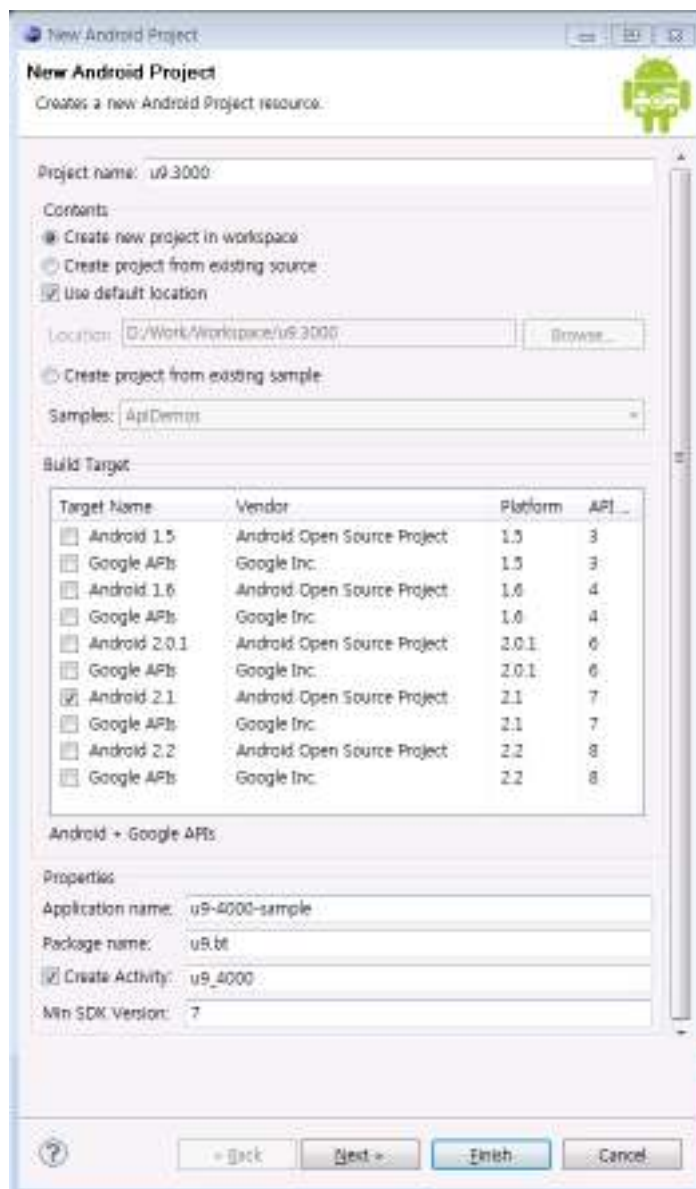
UBISTS U9 Bluetooth Android API 를 이용하는 방법은 Eclipse 에서 새로운 Android Project 또는 기존의 Android Project 에 제공되는 jni 파일과 class 파일을 추가하여 개발을 할 수 있습니다.

이번 장에서는 UBISTS U9 Bluetooth Android API 를 사용하기 위한 환경 설정에 대한 내용을 설명합니다.

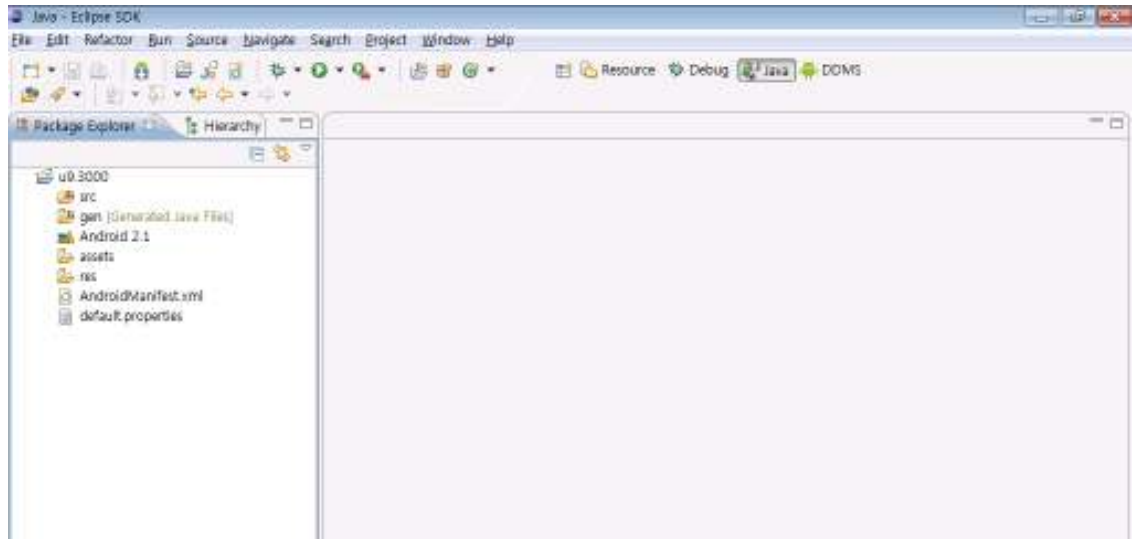
### 2.1 JNI 추가를 위한 환경설정

새로운 안드로이드 프로젝트를 추가하는 것을 기준으로 설명합니다.

- ① 이클립스의 메뉴에서 [File]->[New]->[Android Project]를 선택합니다.



- ② 위의 그림과 같이 적절한 프로젝트 설정을 하고 **Finish** 를 클릭하면 아래 같이 새로운 프로젝트가 생성이 됩니다.



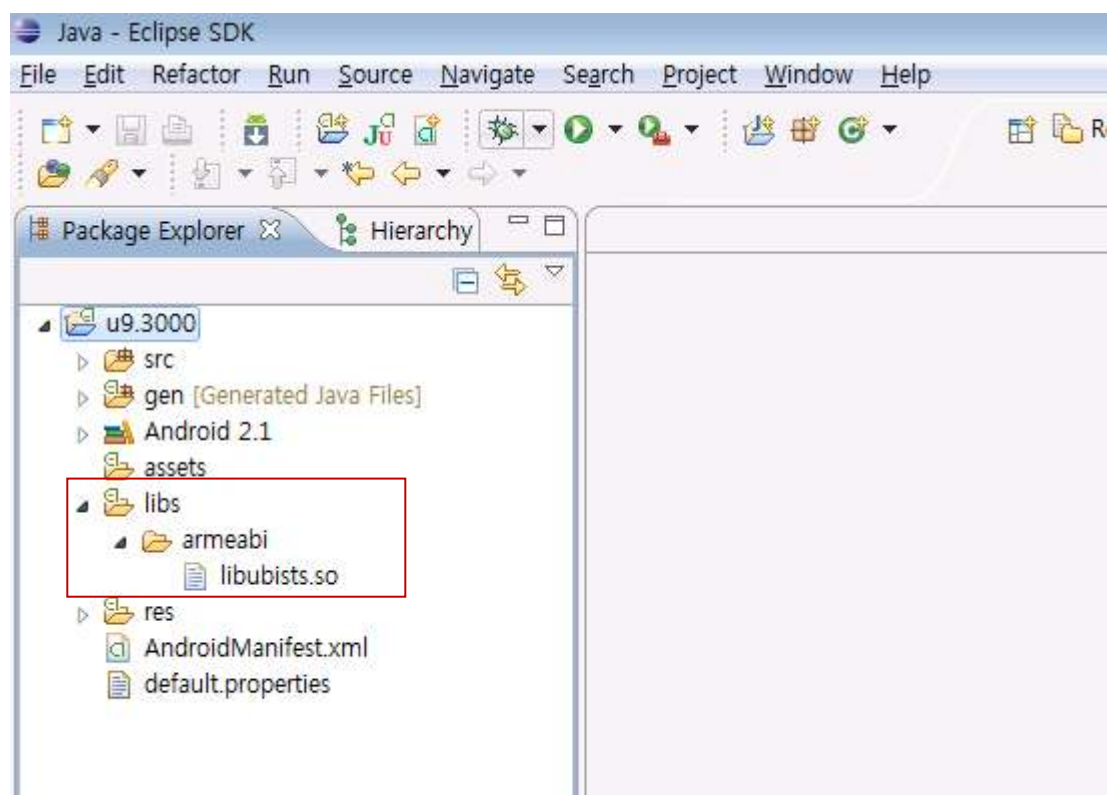
- ③ 윈도우 탐색기를 열어 프로젝트가 생성된 폴더에 제공되는 **libs** 의 압축 파일을 복사 합니다.

Workspace ▶ u9.3000 ▶		
새 폴더		
이름	수정한 날짜	유형
assets	2011-07-18 오후...	파일 폴더
bin	2011-07-18 오후...	파일 폴더
gen	2011-07-18 오후...	파일 폴더
res	2011-07-18 오후...	파일 폴더
src	2011-07-18 오후...	파일 폴더
.classpath	2011-07-18 오후...	CLASSPATH I
.project	2011-07-18 오후...	PROJECT 파일
AndroidManifest.xml	2011-07-18 오후...	XML 문서
default.properties	2011-07-18 오후...	PROPERTIES I
libs.zip	2011-07-18 오후...	ALZip ZIP File

- ④ 복사된 **libs** 압축파일을 아래의 화면과 같이 경로명을 유지한데 압축해제 하고 압축파일은 삭제 합니다.

Workspace ▶ u9.3000 ▶		
새 폴더		
이름	수정한 날짜	유형
assets	2011-07-18 오후...	파일 폴더
bin	2011-07-18 오후...	파일 폴더
gen	2011-07-18 오후...	파일 폴더
libs	2011-07-18 오후...	파일 폴더
res	2011-07-18 오후...	파일 폴더
src	2011-07-18 오후...	파일 폴더
.classpath	2011-07-18 오후...	CLASSPATH I
.project	2011-07-18 오후...	PROJECT 파일
AndroidManifest.xml	2011-07-18 오후...	XML 문서
default.properties	2011-07-18 오후...	PROPERTIES II

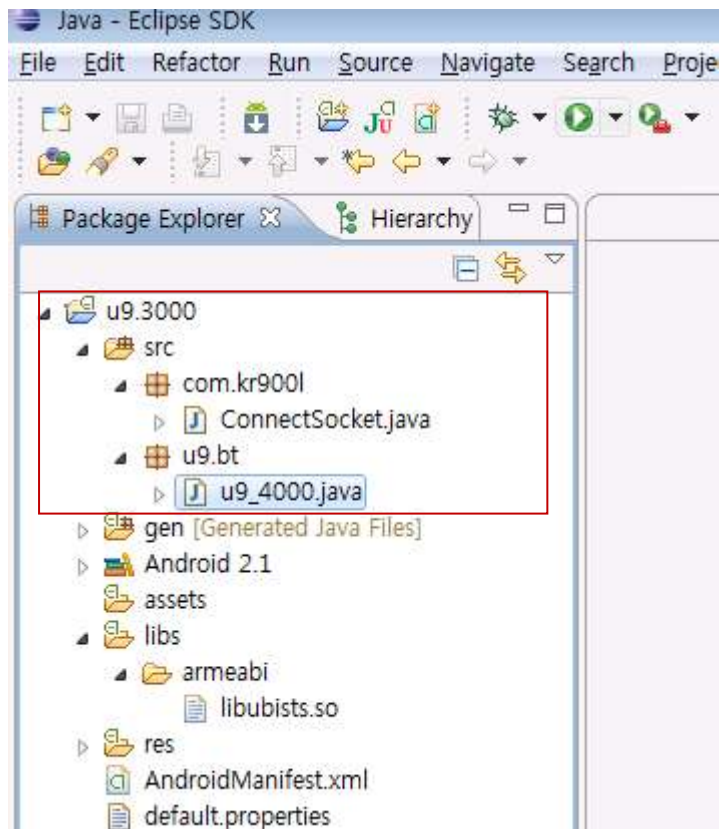
- ⑤ 이클립스에서 프로젝트를 선택후 메뉴에서 [File]->[Refresh]를 선택하여 방금 복사한 jni library 가 로드가 되는지 확인 합니다.



- ⑥ 제공되는 class 파일을 추가하기 위해서 아래와 같이 프로젝트의 src 폴더 아래에 src 압축파일을 복사하고 경로를 유지한 채 압축해제 합니다. 압축파일은 삭제 합니다.

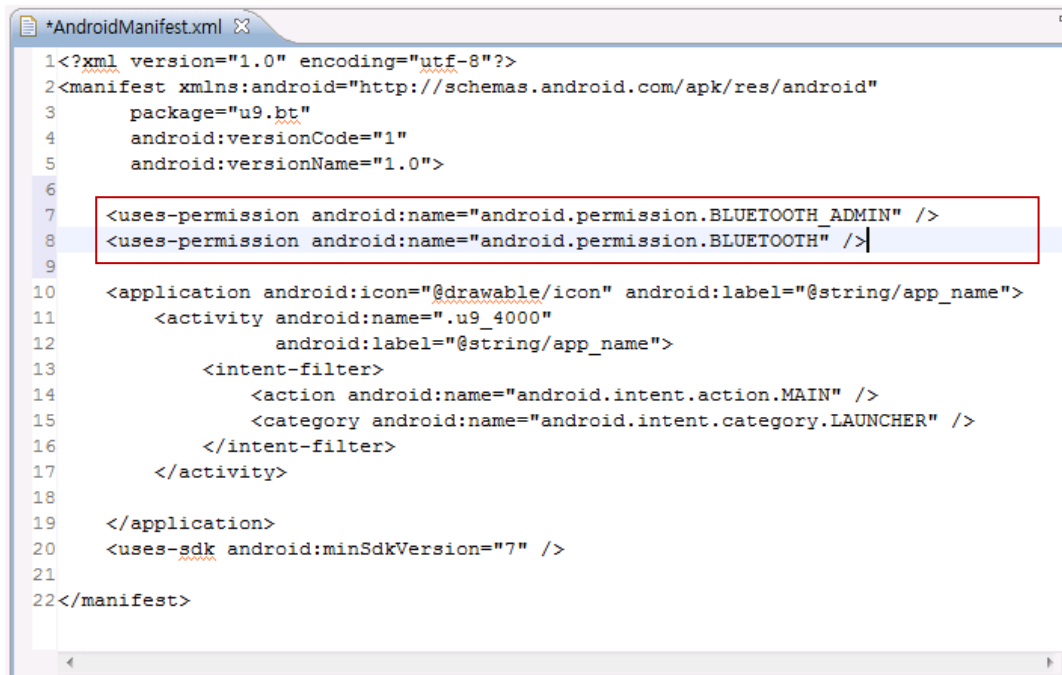
Workspace ▶ u9.3000 ▶ src ▶		
새 폴더		
이름	수정한 날짜	유형
com	2011-07-18 오후...	파일 폴더
u9	2011-07-18 오후...	파일 폴더
com.zip	2011-07-18 오후...	ALZip ZIP File

- ⑦ 이클립스에서 프로젝트를 선택한 뒤 메뉴의 [File]->[Refresh]를 선택하여 java 파일이 아래와 같이 로드 되는지 확인 합니다.



- ⑧ Bluetooth 를 사용하기 위해서 이클립스에서 프로젝트의 res 아래에 있는 AndroidManifest.xml 에 권한을 추가합니다.





```
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="u9.bt"
4    android:versionCode="1"
5    android:versionName="1.0">
6
7    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
8    <uses-permission android:name="android.permission.BLUETOOTH" />
9
10    <application android:icon="@drawable/icon" android:label="@string/app_name">
11        <activity android:name=".u9_4000"
12            android:label="@string/app_name">
13            <intent-filter>
14                <action android:name="android.intent.action.MAIN" />
15                <category android:name="android.intent.category.LAUNCHER" />
16            </intent-filter>
17        </activity>
18    </application>
19    <uses-sdk android:minSdkVersion="7" />
20
21
22</manifest>
```

위의 과정으로 Ubists U9 bluetooth 리더의 이클립스를 이용한 안드로이드 API 환경은 모두 설정 되었습니다.

## 2.3 샘플 소스 코드 실행

- ① 제공되는 u9.3000.zip 을 이클립스 워크스페이스에 복사 후 경로를 유지한 채 압축 해제 합니다.
- ② 해당 프로젝트를 메뉴의 [File]->[Import]를 통하여 이클립스에 로드한뒤 실행하면 데모 프로그램과 같은 동작을 하는 것을 확인 할 수 있습니다. 포함되는 소스코드는 위에서 설명한 환경을 모두 적용하고 있습니다.

## 2.4 API 클래스 이용

- ① 제공되는 API 에서 리더기와 통신 프로토콜에 관련된 기능은 JNI 에 포함되어 있으며, ConnectSocket.java 는 JNI 의 함수의 호출을 대신 해주는 역할을 하고 있습니다.
- ② 제공되는 API 에서 Bluetooth 의 제어, 통신의 연결, 해제 는 ConnectSocket.java 에 소스 코드가 그대로 포함되어 있으며 ConnectReader 함수의 동작은 필요에 따라서 수정 사용할 수 있습니다.
- ③ 단 ConnectSocket 클래스에서 private native 가 붙어 있는 함수는 JNI 와 연결된 함수 이므로 절대 변경하시면 안됩니다. 또한 public int read(byte[] buffer) 함수와 public int read(byte[] buffer) 함수는 JNI 에서 호출되어 사용하는 함수이므로 public 이지만 변경하시면 안됩니다.
- ④ 3 절의 Function Specification 에서는 ConnectSocket 클래스의 public 함수를 기준으로 해당 함수의 동작을 설명합니다.

---

## Chapter 3. Function Specification

### 3.1 ConnectReader

**public** Boolean ConnectReader()

리더와 bluetooth 를 통하여 연결을 시도하고 그 성공여부를 반환합니다. 리더와의 연결이 정상적으로 이루어지면, 초기화를 수행하고 리더와 통신에 필요한 내부 스레드를 생성합니다.

함수의 성공여부를 반환 합니다.

### 3.2 DisconnectReader

**public void** DisconnectReader()

리더와의 bluetooth 연결을 끊습니다.

### 3.3 GetBluetoothState

**public** connectBluetoothState GetBluetoothState()

ConnectReader 함수 호출 후 상태를 파악할 수 있는 함수 입니다.. ConnectReader 가 실패할 경우 실패 이유를 얻을 수 있습니다.

```
public enum connectBluetoothState{  
    btsNone,  
    btsNotAvalibleBluetooth,    // 블루투스가 장치에 없습니다.  
    btsNotEnableBluetooth,    // 블루투스를 활성화 시킬수 없습니다.  
    btsNotPairedBluetooth,    // 페어링에 실패 했습니다.  
    btsFailRfConnect,    // 블루투스 동작 자체에 문제가 있습니다.  
    btsFailBluetoothSocket,    // 블루투스 Socket 스트림을 받아 오지 못했습니다.  
    btsCompleteConnectBluetooth,    // 블루투스가 정상적으로 연결되었습니다.  
}
```

connectBluetoothState 열거자를 반환합니다.

### 3.4 Initialize

**public boolean** Initialize()

리더기와 연결후 리더기를 초기화 합니다. ConnectReader 함수가 성공을 반환할 경우 내부적으로 이미 호출 됩니다. 리더기와 연결이 끊기거나 할 때 초기화할 필요가 있을 경우 사용할 수 있습니다.

---

함수의 성공여부를 반환 합니다.

### 3.5 WriteRegister

**public boolean** WriteRegister(**int** nAddress, **int** nValue)

리더기가 가지고 있는 레지스터의 값을 변경할 때 사용합니다. 레지스터란 리더기가 가지고 있는 환경설정값의 리스트입니다. 레지스터에 관한 자세한 내용은 사용자 매뉴얼을 참고해주세요.

**nAddress** : 레지스터의 주소 입니다. 서로 다른 주소로 레지스터를 구분합니다.

**nValue** : 해당 레지스터에 쓰고 싶은 값입니다.

함수의 성공여부를 반환 합니다.

### 3.6 ReaderRegister

**public int** ReadRegister(**int** nAddress)

리더기가 가지고 있는 레지스터의 값을 얻을 때 사용합니다. 레지스터란 리더기가 가지고 있는 환경설정값의 리스트입니다. 레지스터에 관한 자세한 내용은 사용자 매뉴얼을 참고해주세요.

**nAddress** : 알고 싶은 레지스터의 주소 입니다.

해당 레지스터의 값을 반환 합니다

### 3.7 BeginRead

**public boolean** BeginRead()

리더를 지속적인 읽기 모드로 동작 시킬 때 사용합니다. 이 함수가 호출된 이후 리더는 지속적인 UHF RFID 수집 기능을 활성화 시키며, FinishRead 명령이 호출될 때까지 계속 RFID 태그를 읽습니다. 태그는 리더기의 RF 필드 내에서 반응한 태그의 EPC Code 값을 라이브러리 내부에 지속적으로 쌓아 놓고 있습니다.

단 리더가 바코드 읽기 상태로 되어 있을 경우 이 명령은 무시 됩니다.

함수의 성공 여부를 반환 합니다.

### 3.8 FinishRead

**public void** FinishRead()

BeginRead 함수에 의해 태그 읽기를 수행중인 RFID 리더의 동작을 중지 합니다.

---

### 3.9 GetTagList

**public** String[] GetTagList()

리더기에 의해서 전달된 RFID 태그 코드나 바코드 데이터를 저장하고 있는 문자열 배열을 반환합니다. 이 문자열 배열에는 RFID 태그의 코드는 PC + EPC + CRC 순서로 16 진수값을 문자열로 각 바이트마다 SPACE(' ')를 포함하여 저장되어 있습니다. 바코드는 코드값이 아스키 문자열로 저장되어 있습니다.

String 객체의 배열을 반환 합니다.

### 3.10 GetCountList

**public** String[] GetCountList()

리더기에 의해 전달된 RFID 태그의 코드나 바코드가 읽힌 수량을 저장하는 문자열 배열을 반환합니다. 읽힌 수량은 각 배열에 아스키코드로 저장 되어 있습니다.

String 객체의 배열을 반환 합니다.

### 3.12 GetTagListSize

**public int** GetTagListSize()

현재 저장된 각 태그 리스트의 크기를 반환하는 함수입니다. GetTagList 나 GetCountList 함수에서 반환된 배열의 크기와 같습니다.

현재 저장된 데이터의 리스트의 크기를 반환 합니다.

### 3.13 GetTagByIndex

**public** String GetTagByIndex(int nIndex)

현재 저장된 리스트내에서 RFID 태그 코드나 바코드 정보를 한 개씩 개별로 얻을 때 사용합니다. 인자로 주어지는 int 변수는 직전에 호출된 GetTagListSize 의 크기 이상이 될 수 없습니다.

nIndex : 얻기를 원하는 배열의 위치 index 값 입니다.

해당 Index 의 코드 데이터를 문자열로 반환 합니다.

### 3.14 GetCountByIndex

**public** String GetCountByIndex(int nIndex)

현재 저장된 리스트내에서 RFID 태그 코드나 바코드 정보를 읽힌 횟수를 개별로 얻을 때 사용합니다. 인자로 주어지는 int 변수는 직전에 호출된 GetTagListSize 의 크기 이상이 될 수 없습니다.

---

nIndex : 얻기를 원하는 배열의 위치 index 값 입니다.

해당 Index 의 데이터의 읽힌 회수를 문자열로 반환 합니다.

### 3.15 GetRSSIByIndex

```
public int GetRSSIByIndex(int nIndex)
```

U9-4000 에서는 사용하지 않습니다.

### 3.16 GetRSSILevel

```
public int GetRSSILevelByIndex(int nIndex)
```

U9-4000 에서는 사용하지 않습니다.

### 3.17 GetReadTime

```
public String GetReadTimeByIndex(int nIndex)
```

현재 저장된 리스트내에서 RFID 태그 코드나 바코드 정보를 읽힌 마지막 시간을 개별로 얻을 때 사용합니다. 인자로 주어지는 int 변수는 직전에 호출된 GetTagListSize 의 크기 이상이 될 수 없습니다.

nIndex : 얻기를 원하는 배열의 위치 index 값 입니다.

해당 Index 의 데이터의 읽힌 시간을 "HH:MM:SS"로 반환 합니다.

### 3.18 GetReadTypeByIndex

```
public int GetReadTypeByIndex(int nIndex)
```

현재 저장된 리스트내에서 RFID 태그 코드나 바코드 정보의 종류를 개별로 얻을 때 사용합니다. 인자로 주어지는 int 변수는 직전에 호출된 GetTagListSize 의 크기 이상이 될 수 없습니다.

```
public enum TagType{  
    tagC1G2,           // EPC Gen2 태그를 의미 합니다.  
    tag6B,             // ISO18000-6B 태그를 의미 합니다.  
    tagBarcode,        // 바코드 데이터를 의미 합니다.  
}
```

nIndex : 얻기를 원하는 배열의 위치 index 값 입니다.

TagType 열거자를 반환 합니다.

---

### 3.19 ClearTagList

**public void** ClearTagList()

현재 API 가 저장하고 있는 태그 데이터 리스트를 삭제 합니다.

### 3.20 IsUpdate

**public boolean** IsUpdate()

IsUpdate 함수가 직전이 호출되고 난후 태그 리스트에 추가된 데이터가 있는 확인 하는 함수 입니다.

추가된 데이터가 있으면 true 를 반환합니다.

### 3.21 SendString

**public boolean** SendString(int nSize, byte[] SendData)

U9-4000 의 경우 LCD 에 태그 데이터 16 진수가 표시되는 부분에 사용자가 원하는 문자열을 표시 할 수 있도록 하는 함수입니다. 한글은 허용되지 않습니다. 원하는 ASCII 코드를 byte 배열로 만들어 인자로 넘겨주면 됩니다.

nSize : 표시하고자하는 데이터의 크기입니다.

SendData : 표시하고자하는 ASCII 데이터의 byte 배열 입니다.

성공여부를 반환 합니다.

### 3.22 GetReaderVersion

**public** String GetReaderVersion()

현재 리더기의 Firmware 버전을 얻을 수 있습니다..

"X.X.X" 형태로 리더기의 firmware 버전을 반환합니다.

### 3.23 GetLibVersion

**public** String GetLibVersion()

JNI 라이브러리의 버전을 얻을 수 있습니다.

"X.X.X" 형태로 리더기의 firmware 버전을 반환합니다.

### 3.24 ReadTagMemory

**public boolean** ReadTagMemory(int nBytePos, int nByteSize, byte[] readData)

RFID 에서 태그의 메모리의 특정 위치를 읽을 때 사용합니다. 이 함수를 사용하기 전에 원하는

---

태그의 **BANK** 가 현재 선택된 **BANK** 가 아니라면 **SetMemoryBank** 함수를 먼저 호출해야 합니다.

**nBytePos** : 읽기는 원하는 메모리의 시작 위치를 지정합니다. (함수는 허용되지 않습니다.)

**nByteSize** : 읽기를 원하는 메모리의 크기를 지정 합니다. (함수는 허용되지 않습니다.)

**readData** : 읽혀올 데이터의 크기를 **nByteSize** 만큼 **byte** 배열로 만들어 인자로 넘겨주면 함수의 수행이 성공했을 경우 해당 배열에 읽힌 데이터가 복사되어 있습니다..

함수의 성공여부를 반환합니다.

### 3.25 WriteTagMemory

**public boolean** WriteTagMemory(**int** nBytePos, **int** nByteSize, **byte**[] writeData)

RFID 에서 태그의 메모리의 특정 위치에 데이터를 쓸때 사용합니다. 이 함수를 사용하기 전에 원하는 태그의 **BANK** 가 현재 선택된 **BANK** 가 아니라면 **SetMemoryBank** 함수를 먼저 호출해야 합니다.

**nBytePos** : 쓰기를 원하는 메모리의 시작 위치를 지정합니다. (함수는 허용되지 않습니다.)

**nByteSize** : 쓰기를 원하는 메모리의 크기를 지정 합니다. (함수는 허용되지 않습니다.)

**writeData** : 쓰고자 하는 데이터의 크기를 **nByteSize** 만큼 **byte** 배열로 만들어 인자로 넘겨 줍니다.

함수의 성공여부를 반환 합니다.

### 3.26 LockTag

**public boolean** LockTag(**int** nPayload, **byte**[] AccessPassword)

Tag의 각 메모리의 접근 권한을 설정하는 함수 입니다.

Lock 명령에 사용되는 Payload값의 자세한 설명은 ISO Spec을 참고 하시기 바라며 대략적인 것은 표와 같습니다.

19	18	17	16	15	14	13	12	11	10
Kill Mask		Access Mask		EPC Mask		TID Mask		User Mask	
09	08	07	06	05	04	03	02	01	00
Kill Action		Access Action		EPC Action		TID Action		User Action	

위의 20bit를 lpPayload 인자의 LSB에 00비트부터 19비트까지 값으로 넘겨주면 됩니다. 예를 들어 Mask : 1010100000, Action : 1010100000 이라면 0x000A82A0 가 Payload 가 됩니다.

본 함수에 의해 전달되는 Password는 Access Password 이며, 이 Password는 Lock 되고자 하는

---

Tag의 Memory의 Reserved Bank에 미리 Write되어 있어야 합니다. Access Password가 0x00000000의 값을 가지면, Lock은 정상적으로 수행되지 않습니다.

Tag의 Reserved 영역의 Memory Map이 아래와 같다면,

주소	종류	Memory 값
0x30-0x3F	Access Password	0x5678
0x20-0x2F	Access Password	0x1234

전달될 Access Password : 0x12345678 가 되어야 합니다.

IpPayload : Tag Memory 의 Lock 상태를 하는 payload 의 배열입니다.

nAccessPassword : Tag Memory 의 Reserved Bank 에 지정되어 있는 Access Password(32bit)

함수의 성공여부를 반환 합니다.

### 3.27 KillTag

**public boolean** KillTag(**byte**[] AccessPassword, **byte**[] KillPassword)

Tag의 기능을 완전히 정지 시킬 때 사용하는 함수 입니다. 본 함수가 정상적으로 수행되고 나면, 해당 태그는 더 이상 어떤 리더기의 명령에도 응답하지 않게 되며, Tag의 기능을 상실 하게 됩니다.

본 함수에 의해 전달되는 Password는 두 가지 이며, 이 Password는 Kill 되고자 하는 Tag의 Memory의 Reserved Bank에 미리 Write되어 있어야 합니다. 단, 두 가지 Password중 한 개라도 0x00000000의 값을 가지면, Kill은 정상적으로 수행되지 않습니다.

Tag의 Reserved 영역의 Memory Map이 아래와 같다면,

주소	종류	Memory 값
0x30-0x3F	Access Password	0xDEF0
0x20-0x2F	Access Password	0x9ABC
0x10-0x1F	Kill Password	0x5678
0x00 -0x0F	Kill Password	0x1234

전달될 Kill Password 는 0x12345678, Access Password 는 0x9ABCDEFO 가 되어야 합니다.

nAccessPassword : Tag Memory 의 Reserved Bank 에 지정되어 있는 Access Password(32bit)

nKillPassword : Tag Memory 의 Reserved Bank 에 지정되어 있는 Kill Password(32bit)

함수의 성공여부를 반환 합니다.



---

### 3.28 SetMemoryBank

**public boolean** SetMemoryBank(**int** nBank)

EPC Gen2를 사용하는 경우 Read/Write 하고자 하는 Tag의 Bank 영역을 지정합니다. 두 번째 인자인 Bank값의 의미는 다음과 같습니다.

Bank값	Ban
0	Reserved
1	EPC
2	TID
3	User Memory

nBank : Bank 를 위에 표에 맞추어 int 값으로 지정합니다.

함수의 성공여부를 반환 합니다.

### 3.29 SetGen2QValue

**public boolean** SetGen2QValue(**int** nQValue)

Q Value는 Reader가 Tag를 읽을 때 발생하는 충돌(Collision)을 해결하기 위한 내부 알고리즘에 필요한 Parameter입니다.

Tag 는 Reader 의 RF 출력에 의해 깨어나서 응답을 하도록 되어 있는데, Collision 을 피하기 위해서 Random Number 를 생성합니다. 이 Random Number 의 값의 범위를 결정하는 것이 Q Value 입니다.

nQValue : Q 값을 0-15 사이의 값으로 지정합니다

함수의 성공여부를 반환 합니다.

### 3.30 SetRfPowerAttenuation

**public boolean** SetRfPowerAttenuation(**int** nAttenuation)

RF 출력을 조절할 수 있는 RF Attenuation값을 설정합니다.

RF Attenuation 값은 1 씩 조정 가능합니다. 30dBm(1W)의 출력이 나오는 Attenuation 값은 운용 주파수 마다 다릅니다. 이 값은 비례적으로 약 1dBm 의 RF 출력의 차이를 나타냅니다.

nAttenuation: Attenuation 값을 0-30 사이의 값으로 지정합니다

함수의 성공여부를 반환 합니다.

---

### 3.31 GetMemoryBank

**public int** GetMemoryBank()

현재 리더에 설정되어 있는 Bank 값을 가져옵니다.

Bank 값을 반환합니다.

### 3.32 GetGen2QValue

**public int** GetGen2QValue()

현재 리더에 설정되어 있는 Q Value 값을 가져옵니다.

Q 값을 반환합니다.

### 3.33 GetRfPowerAttenuation

**public int** GetRfPowerAttenuation()

현재 리더에 설정되어 있는 RF Attenuation 값을 가져옵니다.

RF Attenuation 값을 반환합니다.

### 3.34 GetReaderAlive

**public boolean** GetReaderAlive()

현재 리더와의 연결이 유지가 되고 있는 확인 하는 함수 입니다. 정해진 시간주기로 리더와 API 가 서로 Alive 신호를 교환하도록 하여 연결을 체크 합니다.

연결이 유지되고 있으면 true 를 반환합니다.

### 3.35 SetReaderAliveTimeOut

**public void** SetReaderAliveTimeOut(**int** nTimeOut)

Alive 체크를 위한 시간 주기를 지정합니다. 시간은 초입니다.

nTimeOut : 초단위의 시간을 지정합니다.

### 3.36 SavRegister

**public boolean** SaveRegister()

현재 리더에 지정된 register 의 값을 영구 저장합니다. 레지스터는 리더의 전원이 OFF-ON 되면 모두 초기값으로 지정되는데, 이 함수를 호출하여 성공하면 이 함수를 호출한 시점의 register 값으로 On 초기화로 지정됩니다.

---

함수의 성공여부를 반환합니다.

### 3.37 DefaultRegister

**public boolean** DefaultRegister()

레지스터의 값을 공장 초기치로 돌려놓는 함수 입니다.

함수의 성공여부를 반환합니다.