



ATID Co.,Ltd

RFID API Reference Guide for Android Developers

Android Developer Guide

SW Team

2022-06-20

개정 이력

버전	개정일자	개정사유 ¹	개정내역 ²	작성자
v 1.0	2021-07-14	초안	신규 생성	SW Team
v 1.1	2022-06-20	추가	ISO 18000-6B 와 Rail tag API 추가	SW Team

¹ 개정사유 : 제정 또는 개정 내용이 이전 문서에 대해 추가/수정/삭제인지 선택 기입

² 개정내역 : 개정이 발생하는 페이지 번호와 변경 내용을 기술

목차

목차	3
1. Intro	5
2. Reference Library Guide	6
2.1. ATRfidManager Class	6
2.1.1. Method	6
2.2. ATRfidReader Class	8
2.2.1. Method	8
2.3. ATRfidATX00S1Reader Class	26
2.3.1. Method	26
2.4. ATRfid900MAREader Class	31
2.4.1. Method	31
2.5. RfidReaderEventListener Interface	34
2.5.1. Method	34
2.6. Parameter Classes	36
2.6.1. RangeValue Class	36
2.6.2. LockParam Class	36
2.6.3. SelectionMask6c Class	39
2.6.4. SelectionMask6b Class	42
2.6.5. EpcMatchParam Class	44
2.6.6. QValue Class	46
2.7. Enumerations	49
2.7.1. ActionState	49
2.7.2. BankType	49
2.7.3. TagType	49
2.7.4. ConnectionState	50
2.7.5. InventorySession	50
2.7.6. InventoryTarget	50
2.7.7. LockType	50
2.7.8. RfidModuleType	50
2.7.9. SelectFlagType	52
2.7.10. MaskMatchingType	52
2.7.11. GlobalBandType	52
2.7.12. MaskActionType	53
2.7.13. MaskTargetType	53



RFID API Reference Guide for Android Developers

Android Developer Guide

회사

ATID Co.,Ltd

문서이름

작성자

SW Team

날자

2022-06-20

버전

V1.1

2.7.14.	SingulationAlgorithm	54
2.7.15.	ResultCode	55

ATID CO.,Ltd

1. Intro

본 문서는 RFID SDK Library를 사용하여 응용 프로그램을 개발하고자 하는 Android개발자들을 위하여 SDK Library 사용 방법을 기술 하는데 그 목적이 있다.

본 문서에서 사용되는 개발 도구는 Android Studio 가 사용되었고 개발 대상 플랫폼은 Android 10 를 지원한다.

Dependency libraries description,

Library	설명
atid.dev.rfid	RFID Module을 제어하기 위한 Android용 Library
atid.system.comm	Module과의 통신을 제어하는 Android용 Library
atid.system.jcomm	Module과의 통신을 제어하는 Android용 Library
atid.system.ctrl	Module의 전원 제어를 위한 Android용 Library
atid.system.device	Module의 정보를 관리하는 Android용 Library
atid.util	SDK Library 내부에서 사용되는 Utility Library

2. Reference Library Guide

2.1. ATRfidManager Class

ATRfidManager 는 PDA에 장착된 RFID Instance의 생성 및 Resource를 관리하며 Activity간에 RFID의 Resource 관리를 하기 위해 제공 되는 Class이다.

2.1.1. Method

2.1.1.1. getInstance

RFID Reader Object를 생성하고, RFID Module과 RFID Reader Object와의 연결 한다.

➤ Syntax

```
public static ATRfidReader getInstance()
```

➤ Return value

RFID Reader의 Instance를 반환한다.

➤ Remarks

getInstance 메서드는 성공적으로 동작 시 Reader 의 인스턴스를 생성하여 반환한다.
Main Activity 의 onCreate 메서드에서 호출하도록 한다.

2.1.1.2. onDestroy

getInstance 메소드에서 생성된 RFID Reader를 RFID Module과 연결을 끊고 RFID Reader Object를 Release한다.

➤ Syntax

```
public static void onDestroy()
```

➤ Remarks

onDestroy 메서드는 RFID Reader Object 의 Resource 를 Free 하고, Release 작업을 수행한다.

Main Activity 의 onDestroy 메서드에서 호출하도록 한다.

2.1.1.3. wakeUp

RFID Module이 Sleep 상태에 있을 때, RFID Module을 wakeup하기 위해 호출한다.

➤ Syntax

```
public static void wakeUp()
```

➤ Remarks

App의 모든 Activity의 onStart 메서드에서 호출 한다. sleep메서드를 호출한 후, wakeup 메서드를 호출하지 않으면, RFID Module은 동작하지 않는다.

wakeUp 과 sleep 의 호출은 반드시 pair 로 이루어져야 한다.

2.1.1.4. **sleep**

RFID Module이 Wakeup 상태에 있을 때, RFID Module을 sleep 하기 위해 호출한다.

➤ **Syntax**

```
public static void sleep()
```

➤ **Remarks**

App의 모든 Activity의 onStop메서드에서 호출한다. onStop메서드에서 sleep을 호출하지 않으면 PDA의 전원 버튼을 눌러서 Sleep 모드에 들어 갔어도, Module은 계속 동작하게 된다.

wakeUp 과 sleep 의 호출은 반드시 pair 로 이루어져야 한다.

2.1.1.5. **getVersion**

라이브러리 버전 정보를 반환한다.

➤ **Syntax**

```
public static String getVersion()
```

➤ **Remarks**

사용중인 atid.dev.rfid.jar 의 버전 정보를 반환한다.

2.1.1.6. **checkAutoModule**

단말기에 장착된 RFID Module을 검색하여 Instance를 반환한다.

➤ **Syntax**

```
public static ATRfidReader checkAutoModule ()
```

➤ **Remarks**

단말기에서 지원하는 모든 RFID Module에 대해 검색을 해서, 검색된 RFID Module에 대한 RFID Reader의 Instance를 반환한다.

라이브러리 내부에서 사용하는 Method이므로, 임의로 사용해서는 안 된다.

2.1.1.7. **checkModule**

주어진 RfidModuleType에 대한 Instance를 반환한다.

➤ **Syntax**

```
public static ATRfidReader checkModule(RfidModuleType type)
```

➤ **Parameters**

type: 생성할 Instance의 Module Type.

➤ **Remarks**

type으로 입력한 Module Type의 Instance 생성을 시도한다.

라이브러리 내부에서 사용하는 Method이므로, 임의로 사용해서는 안 된다.

2.2. ATRfidReader Class

ATRfidReader Class는 RFID Reader의 Instance를 생성하여 RFID Reading 및 Configuration을 Instance설정한다.

2.2.1. Method

2.2.1.1. Reset

RFID Module을 reset 한다.

➤ Syntax

```
public void Reset()
```

➤ Remarks

특수한 상황이 아니면 임의로 사용하는 것을 권장하지 않는다.

2.2.1.2. setLogLevel

LogCat으로 출력되는 message의 출력 레벨을 설정한다.

➤ Syntax

```
public void setLogLevel(int level)
```

➤ Parameters

level: message 출력 레벨.

➤ Remarks

라이브러리 debugging 용도 이므로, 임의로 사용해서는 안된다.

2.2.1.3. destroy

destroy 메서드는 강제로 ATRfidReader의 Instance를 파괴하는 메서드이다.

➤ Syntax

```
public void destroy()
```

➤ Remarks

ATRfidManager 에서 호출되므로 따로 호출할 필요는 없다.

2.2.1.4. powerControl

powerControl 메서드는 RFID Module전원을 제어한다.

➤ Syntax

```
public void powerControl(boolean enabled)
```

➤ Parameters

enabled: true이면 RFID Module의 전원을 켜고, false이면 RFID Module의 전원을 끈다.

➤ Remarks

RFID Module 의 Power 는 ATRfidManager 내부에서 자동으로 조정을 한다. 특수한 상황이 아니면 임의로 사용하는 것을 권장하지 않는다.

2.2.1.5. connect

connect 메서드는 RFID Module과 연결을 수행한다.

➤ Syntax

```
public boolean connect()
```

➤ Return value

연결이 정상적으로 되었을 때 true를 반환하며 실패 시 false를 반환한다.

➤ Remarks

Connect 는 RFID Device 와 연결시 1 회만 실행을 하도록 한다.

2.2.1.6. disconnect

disconnect 메서드는 RFID Module과 연결을 해제한다.

➤ Syntax

```
public void disconnect()
```

➤ Remarks

disconnect 는 RFID Device 와 연결을 해제 시 1 회만 실행을 하도록 한다.

2.2.1.7. readEpc6cTag

readEpc6cTag 메서드는 ISO 18000-6C 태그를 Single 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode readEpc6cTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readEpc6cTag 메서드는 RFID Module 에서 ISO18000-6C Tag 에 대하여 Inventory 기능을 수행하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener 의 onReaderReadTag Method 를 통해서 태그 데이터가 전달 된다.

2.2.1.8. readEpc6bTag

readEpc6cTag 메서드는 ISO 18000-6B 태그를 Single 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode readEpc6cTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readEpc6bTag 메서드는 RFID Module 에서 ISO18000-6B Tag 에 대하여 Inventory 기능을 수행하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener 의 onReaderReadTag Method 를 통해서 태그 데이터가 전달 된다.

2.2.1.9. readEpcRailTag

readEpc6cTag 메서드는 AEI/Rail 태그를 Single 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode readEpcRailTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readEpcRailTag 메서드는 RFID Module 에서 AEI/Rail Tag 에 대하여 Inventory 기능을 수행하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener 의 onReaderReadTag Method 를 통해서 태그 데이터가 전달 된다.

2.2.1.10. inventory6cTag

inventory6cTag 메서드는 ISO 18000-6C 태그를 Multiple 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode inventory6cTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

inventory6cTag 메서드는 RFID Module 에서 ISO18000-6C Tag 에 대하여 Inventory 기능을 수행하며 정상적으로 수행되었을 경우, RfidReaderEventListener 의 onReaderReadTag Method 를 통해서 태그 데이터가 전달 된다.

2.2.1.11. inventory6bTag

inventory6cTag 메서드는 ISO 18000-6B 태그를 Multiple 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode inventory6bTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

inventory6cTag 메서드는 RFID Module 에서 ISO18000-6B Tag 에 대하여 Inventory 기능을 수행하며 정상적으로 수행되었을 경우, RfidReaderEventListener 의 onReaderReadTag Method 를 통해서 태그 데이터가 전달 된다.

2.2.1.12. inventoryRailTag

inventory6cTag 메서드는 AEI/Rai 태그를 Multiple 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode inventoryRailTag()
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

inventoryRailTag 메서드는 RFID Module에서 AEI/Rail Tag에 대하여 Inventory 기능을 수행하며 정상적으로 수행되었을 경우, RfidReaderEventListener의 onReaderReadTag Method를 통해서 태그 데이터가 전달 된다.

2.2.1.13. readMemory6c

readMemory6c 메서드는 ISO 18000-6C Tag에 대하여 Read Memory 기능을 수행한다.

➤ Syntax

```
public ResultCode readMemory6c(BankType bank, int offset, int length)
public ResultCode readMemory6c(BankType bank, int offset, int length,
String password)
public ResultCode readMemory6c(BankType bank, int offset, int length,
EpcMatchParam epc)
public ResultCode readMemory6c(BankType bank, int offset, int length,
String password, EpcMatchParam epc)
```

➤ Parameters

bank : Tag의 Memroy Bank를 지정한다.

offset : Tag Data의 시작 주소를 WORD 단위로 지정한다.

length : Tag Data의 길이를 WORD 단위로 지정한다.

password : Tag의 Access Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Read Memory를 수행할 때, 태그의 EPC 정보를 지정한다.

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readMemory6c 메서드는 RFID Module에서 ISO18000-6C Tag에 대하여 Read Memory 기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

2.2.1.14. readMemory6b

readMemory6b 메서드는 ISO 18000-6B Tag에 대하여 Read Memory 기능을 수행한다.

➤ Syntax

```
public ResultCode readMemory6c(int offset, int length)
```

➤ **Parameters**

offset : Tag Data의 시작 주소를 BYTE 단위로 지정한다.

length : Tag Data의 길이를 BYTE 단위로 지정한다.

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

readMemory6b 메서드는 RFID Module에서 ISO18000-6B Tag에 대하여 Read Memory 기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

이 메서드는 getModuleType의 반환 값이 I900MA 일 경우에만 사용할 수 있다.

2.2.1.15. writeMemory6c

writeMemory6c 메서드는 ISO 18000-6C Tag에 대하여 Write Memory 기능을 수행한다.

➤ **Syntax**

```
public ResultCode writeMemory6c(BankType bank, int offset, String data)
public ResultCode writeMemory6c(BankType bank, int offset, String data,
    String password)
public ResultCode writeMemory6c(BankType bank, int offset, String data,
    EpcMatchParam epc)
public ResultCode writeMemory6c(BankType bank, int offset, String data,
    String password, EpcMatchParam epc)
```

➤ **Parameters**

bank : Tag의 Memory Bank를 지정한다.

offset : Tag Data의 시작 주소를 WORD 단위로 지정한다.

data : Tag 기록하고자 하는 Data를 WORD 단위 Hex 문자열로 지정한다.

password : Tag의 Access Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Write Memory를 수행할 때, 태그의 EPC 정보를 지정한다.

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

writeMemory6c 메서드는 RFID Module이 ISO18000-6C Tag에 대하여 Write Memory 기능을 수행하게 하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

2.2.1.16. writeMemory6b

writeMemory6b 메서드는 ISO 18000-6B Tag에 대하여 Write Memory 기능을 수행한다.

➤ **Syntax**

```
public ResultCode writeMemory6b(int offset, String data)
```

➤ Parameters

offset : Tag Data의 시작 주소를 BYTE 단위로 지정한다.

data : Tag 기록 하고자 하는 Data를 BYTE 단위 Hex 문자열로 지정한다.

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

writeMemory6b 메서드는 RFID Module이 ISO18000-6B Tag에 대하여 Write Memory 기능을 수행하게 하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

이 메서드는 getModuleType의 반환 값이 I900MA 일 경우에만 사용할 수 있다.

2.2.1.17. lock6c

lock6c 메서드는 ISO 18000-6C Tag에 대하여 Lock 기능을 수행한다.

➤ Syntax

```
public ResultCode lock6c(LockParam param)
public ResultCode lock6c(LockParam param, String password)
public ResultCode lock6c(LockParam param, EpcMatchParam epc)
public ResultCode lock6c(LockParam param, String password, EpcMatchParam epc)
```

➤ Parameters

param : Lock을 수행하고자 하는 Memory Bank 영역에 대한 정보를 수록한 LockParam Class의 Instance.

password : Tag의 Access Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Lock을 수행할 때, 태그의 EPC 정보를 지정한다.

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

lock6c 메서드는 RFID Module에서 ISO18000-6C Tag에 대하여 Lock 기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

2.2.1.18. kill6c

kill6c 메서드는 ISO 18000-6C Tag에 대하여 Kill을 수행한다.

➤ Syntax

```
public ResultCode kill6c(String password)
public ResultCode kill6c(String password, EpcMatchParam epc)
```

➤ Parameters

password : Tag의 Kill Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Kill을 수행할 때, 태그의 EPC 정보를 지정한다.

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

kill6c 메서드는 RFID Module에서 ISO18000-6C Tag에 대하여 Kill기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

2.2.1.19. stop

stop 메서드는 모든 Action 계열(Inventory, Read / Write Memory, Lock, Kill)의 동작을 중지 시킨다.

➤ **Syntax**

```
public ResultCode stop()
```

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

RFID Module 이 동작 중인 모든 작업을 취소하고 상태를 중지 상태로 변경한다.

2.2.1.20. saveProperties

변경된 RFID Module의 속성값을 저장한다.

➤ **Syntax**

```
public ResultCode saveProperties()
```

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

사용자에 의한 변경된 RFID Module의 속성값들을 Module에 저장하여, 어플리케이션이 재 실행 되더라도 이전 값들을 유지 시킨다.

이 메서드는 getModuleType의 반환 값이 I900MA 일 경우에만 사용할 수 있다.

2.2.1.21. defaultProperties

defaultProperties 메서드는 RFID Module의 모든 속성값을 초기값으로 되돌린다.

➤ **Syntax**

```
public ResultCode defaultProperties()
```

➤ **Return value**

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ **Remarks**

defaultProperties메서드는 RFID Module에 설정되어 있는 모든 속성값을 공장 초기값으로 설정한다.

2.2.1.22. **getFirmwareVersion**

RFID Module의 Firmware 버전을 반환한다.

➤ **Syntax**

```
public String getFirmwareVersion() throws ATRfidReaderException
```

➤ **Return value**

Firmware version을 문자열로 반환한다.

➤ **Remarks**

getFirmwareVersion메서드는 반드시 getInstance메서드 호출 이후에 사용하여야 한다.

2.2.1.23. **getState**

getState 메서드는 Reader Object와 RFID Module의 연결 상태를 반환한다.

➤ **Syntax**

```
public ConnectionState getState()
```

➤ **Return value**

연결 상태를 반환하며 자세한 내용은 ConnectionState을 참조한다.

➤ **Remarks**

getState메서드는 반드시 getInstance메서드 호출 이후에 사용하여야 한다.

2.2.1.24. **getAction**

getState메서드는 RFID Module의 동작 상태를 반환한다.

➤ **Syntax**

```
public ActionState getAction()
```

➤ **Return value**

RFID Module의 동작 상태를 반환하며 자세한 내용은 ActionState을 참조한다.

➤ **Remarks**

getAction메서드는 반드시 getInstance메서드 호출 이후에 사용하여야 한다.

2.2.1.25. **getOperationTime**

getInventoryTime메서드는 RFID Module이 동작할 하는 시간을 반환한다.

➤ **Syntax**

```
public int getOperationTime() throws ATRfidReaderException
```

➤ **Return value**

Module 동작 시간을 ms단위로 반환한다.

➤ **Remarks**

Module의 동작 시간을 정수형으로 반환하며, 시간은 ms단위로 지정되어 있다. 이 값이 0으로 설정되면 Module에 중지 명령이 내려지거나 수행중인 명령이 완료 될 때까지 동작한다.

2.2.1.26. setOperationTime

setOperationTime메서드는 RFID Module이 동작할 시간을 설정한다.

➤ Syntax

```
public void setOperationTime(int time) throws ATRfidReaderException
```

➤ Parameters

time : Module 동작 시간. ms단위

➤ Remarks

Module의 동작 시간을 정수형으로 설정하며, 시간은 ms단위이다. 이 값이 0으로 설정되면 Module에 중지 명령이 내려지거나 수행중인 명령이 완료 될 때까지 동작한다.

2.2.1.27. getPowerRange

RFID Module의 Antenna의 출력 레벨의 Minimum, Maximum을 반환한다.

➤ Syntax

```
public RangeValue getPowerRange() throws ATRfidReaderException
```

➤ Return value

RFID Module의 국가 설정 값에 따른 출력 레벨의 Minimum, Maximum값을 가지고 있는 RangeValue Class의 Instance를 반환한다.

➤ Remarks

RFID Module의 국가 설정 상태에 따라 변경된다.

2.2.1.28. getPower

getPower메서드는 RFID Module의 Antenna 출력 레벨을 반환한다.

➤ Syntax

```
public int getPower() throws ATRfidReaderException
```

➤ Return value

Antenna의 출력 레벨을 *10의 정수형으로 반환한다.

➤ Remarks

반환되는 값은 출력 값에 *10을 한 값이 출력된다. 만약 현재 설정되어 있는 Antenna 출력값이 30dbm이라면 *10을 한 300이 반환된다. 반환되는 값은 getPowerRange메서드에서 반환한 최대값과 최소값 사이에서 반환된다.

2.2.1.29. setPower

setPower메서드는 RFID Module의 Antenna 출력 레벨을 설정한다.

➤ Syntax

```
public void setPower(int power) throws ATRfidReaderException
```

➤ Parameters

power : Antenna 출력 레벨 * 10의 정수형값

➤ Remarks

설정값은 Antenna 출력값의 *10을 한 값을 설정한다. 만약 설정하려고 하는 Antenna 출력값이 30dbm이라면 *10을 한 300을 설정한다. 설정할 수 있는 값은 getPowerRange메서드에서 반환한 최대값과 최소값 사이에서 설정한다.

2.2.1.30. getAntennaCycleCount

getAntennaCycleCount 메서드는 동작 수행에 필요한 Antenna Cycle count를 반환한다.

➤ Syntax

```
public int getAntennaCycleCount() throws ATRfidReaderException
```

➤ Return value

Antenna Cycle count를 정수형으로 반환한다.

➤ Remarks

설정 값은 0~65535 값이 되며, 이는 Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.31. setAntennaCycleCount

setAntennaCycleCount 메서드는 동작 수행에 필요한 Antenna Cycle count를 설정한다.

➤ Syntax

```
public void setAntennaCycleCount(int count) throws ATRfidReaderException
```

➤ Parameters

count : Antenna Cycle count

➤ Remarks

설정 값은 0~65535 값이 되며, 이는 Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.32. getDwellTime

getDwellTime 메서드는 antenna의 dwell time을 반환한다.

➤ Syntax

```
public int getDwellTime() throws ATRfidReaderException
```

➤ Return value

dwell time을 정수형으로 반환한다.

➤ **Remarks**

Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.33. **setDwellTime**

setDwellTime 메서드는 antenna의 dwell time을 설정한다.

➤ **Syntax**

```
public void setDwellTime(int time) throws ATRfidReaderException
```

➤ **Parameters**

time : dwell time

➤ **Remarks**

Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.34. **getInventoryRoundCount**

getInventoryRoundCount 메서드는 inventory round count를 반환한다.

➤ **Syntax**

```
public int getDwellTime() throws ATRfidReaderException
```

➤ **Return value**

dwell time을 정수형으로 반환한다.

➤ **Remarks**

Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.35. **setInventoryRoundCount**

setDwellTime 메서드는 inventory round count를 설정한다.

➤ **Syntax**

```
public void setDwellTime(int count) throws ATRfidReaderException
```

➤ **Parameters**

time : inventory round count

➤ **Remarks**

Module의 전반적인 성능에 영향을 주게 되므로, 임의로 사용을 해서는 안 된다.

2.2.1.36. **getInventoryTime**

getInventoryTime메서드는 RFID Module의 Inventory Round 시간 중에서 Antenna가 활성화 되어 있는 시간을 반환한다.

➤ **Syntax**

```
public int getInventoryTime() throws ATRfidReaderException
```

➤ **Return value**

Antenna가 활성화 되어 있는 시간을 반환한다.(ms단위)

➤ **Remarks**

RFID Module는 한 번의 Inventory Round Time동안 Inventory Time과 Idle Time을 가진다. Inventory Round Time은 최대 400ms로 Inventory Time과 Idle Time을 합쳐서 Inventory Round Time을 넘을 수 없다.

2.2.1.37. **setInventoryTime**

setInventoryTime메서드는 RFID Module의 Inventory Round 시간 중에서 Antenna가 활성화 되어 있는 시간을 설정한다.

➤ **Syntax**

```
public void setInventoryTime(int time) throws ATRfidReaderException
```

➤ **Parameters**

time : Antenna의 활성화 시간. (ms단위)

➤ **Remarks**

getInventoryTime을 참조한다.

2.2.1.38. **getIdleTime**

getIdleTime메서드는 RFID Module의 Inventory Round시간 중에서 Antenna의 유휴 시간을 반환한다.

➤ **Syntax**

```
public int getIdleTime() throws ATRfidReaderException
```

➤ **Return value**

Antenna의 유휴 시간을 반환한다. (ms단위)

➤ **Remarks**

RFID Module은 한 번의 Inventory Round Time동안 Inventory Time과 Idle Time을 가진다. Inventory Round Time은 최대 400ms로 Invenotry Time과 Idle Time은 합쳐서 Inventory Round Time을 넘을 수 없다.

2.2.1.39. **setIdleTime**

setIdleTime메서드는 RFID Module의 Inventory Round Time중에서 Antenna의 유휴시간을 설정한다.

➤ **Syntax**

```
public void setIdleTime(int time) throws ATRfidReaderException
```

➤ **Parameters**

time : Antenna의 유휴 시간. (ms단위)

➤ **Remarks**

getIdleTime을 참조한다.

2.2.1.40. **getReportRssi**

getReportRssi메서드는 RFID Module이 Inventory를 수행하면서 읽은 EPC값과 함께 RSSI 값을 같이 보고 할 것인지 여부를 반환한다.

➤ **Syntax**

```
public boolean getReportRssi() throws ATRfidReaderException
```

➤ **Return value**

RSSI값 보고 여부를 결정하는 boolean형

➤ **Remarks**

반환 값이 true이면 ReaderReadTag이벤트로 반환되는 데이터에 rssi값이 같이 보고 된다.

2.2.1.41. **setReportRssi**

setReportRssi메서드는 RFID Module의 Inventory를 수행하면서 읽은 EPC값과 함께 RSSI 값을 같이 보고 할 것인지 여부를 설정한다.

➤ **Syntax**

```
public void setReportRssi(boolean enabled) throws ATRfidReaderException
```

➤ **Parameters**

enabled : RSSI값 보고 여부를 결정하는 boolean형

➤ **Remarks**

설정 값이 true이면 ReaderReadTag이벤트로 반환되는 데이터에 rssi값이 같이 보고 된다.

2.2.1.42. **getInventorySession**

getInventorySession메서드는 RFID Module이 Inventory를 수행하면서 사용하는 Tag의 Session을 반환한다.

➤ **Syntax**

```
public InventorySession getInventorySession() throws ATRfidReaderException
```

➤ **Return value**

Tag Session을 나타내는 InventorySession 열거형

2.2.1.43. **setInventorySession**

setInventorySession메서드는 RFID Module의 Inventory를 수행하면 사용하는 Tag의 Session을 설정한다.

➤ **Syntax**

```
public void setInventorySession(InventorySession session) throws ATRfidReaderException
```

➤ **Parameters**

session : Tag Session을 나타내는 InvenotrySession 열거형

2.2.1.44. **getInventoryTarget**

getInventoryTarget메서드는 RFID Module이 Inventory를 수행하면서 사용하는 Tag의 Session에 상태를 반환한다.

➤ **Syntax**

```
public InventoryTarget getInventoryTarget() throws ATRfidReaderException
```

➤ **Return value**

Tag Session의 상태를 나타내는 InventoryTarget 열거형

2.2.1.45. **setInventoryTarget**

setInventoryTarget메서드는 RFID Module의 Inventory를 수행하면 사용하는 Tag의 Session에 상태를 설정한다.

➤ **Syntax**

```
public void setInventoryTarget(InventoryTarget target) throws ATRfidReaderException
```

➤ **Parameters**

target : Tag Session의 상태를 나타내는 InvenotryTarget 열거형

2.2.1.46. **getSelectFlag**

getSelectFlag 메서드는 Inventory를 수행하면서 사용하는 Tag의 SL 상태를 반환한다.

➤ **Syntax**

```
public SelectFlagType getSelectFlag() throws ATRfidReaderException
```

➤ **Return value**

Tag SL 상태를 나타내는 SelectFlagType 열거형

2.2.1.47. **setSelectFlag**

setSelectFlag 메서드는 Inventory를 수행하면 사용하는 Tag의 SL 상태를 설정한다.

➤ **Syntax**

```
public void setSelectFlag(SelectFlagType type) throws ATRfidReaderException
```

➤ **Parameters**

type : Tag SL 상태를 나타내는 SelectFlagType 열거형

2.2.1.48. **getUseSelectionMask**

getUseSelectionMask메서드는 RFID Module이 Inventory나 기타 Access명령을 수행하면서 Selection Mask를 사용할 것인지 여부를 반환한다.

➤ **Syntax**

```
public boolean getUseSelectionMask() throws ATRfidReaderException
```

➤ **Return value**

Selection Mask 사용 여부를 결정하는 boolean형

➤ **Remarks**

반환 값이 true이면 설정된 Selection Mask기능 사용하여 Inventory나 Access명령을 수행한다.

2.2.1.49. **setUseSelectionMask**

setUseSelectionMask메서드는 RFID Module이 Inventory나 기타 Access명령을 수행하면서 Selection Mask를 사용할 것인지 여부를 설정한다.

➤ **Syntax**

```
public void setUseSelectionMask(boolean used) throws ATRfidReaderException
```

➤ **Parameters**

enabled : Selection Mask 사용 여부를 결정하는 boolean형

➤ **Remarks**

설정 값이 true이면 설정된 Selection Mask기능 사용하여 Inventory나 Access명령을 수행한다.

2.2.1.50. **getSelectionMask6c**

getSelectionMask6c메서드는 RFID Module에 설정되어 있는 Selection Mask값을 반환한다.

➤ **Syntax**

```
public SelectionMask6c getSelectionMask6c(int index) throws ATRfidReaderException
```

➤ **Parameters**

index : 반환 받고자 하는 Selection Mask 배열의 Index (0 ~ 7)

➤ **Return value**

지정된 index에 설정되어 있는 Selection Mask의 정보를 가지고 있는 SelectionMask6c Class의 Instance

➤ **Remarks**

RFID Module은 최대 8개까지 Selection Mask를 설정할 수 있다.

2.2.1.51. **setSelectionMask6c**

setSelectionMask6c메서드는 RFID Module에 Selection Mask를 설정한다.

➤ **Syntax**

```
public void setSelectionMask6c(int index, SelectionMask6c mask) throws ATRfidReaderException
```

➤ **Parameters**

index : 설정하고자 하는 Selection Mask 배열의 Index (0 ~ 7)

mask : 설정하고자 하는 Selection Mask의 정보를 가지고 있는 SelectionMask6c Class의 Instance

➤ **Remarks**

RFID Module은 최대 8개까지 Selection Mask를 설정할 수 있다.

2.2.1.52. **getSelectionMask6cList**

getSelectionMask6cList 메서드는 RFID Module에 설정되어 있는 Selection Mask 값들의 List를 반환한다.

➤ **Syntax**

```
public SelectionMask6cList getSelectionMask6cList() throws ATRfidReaderException
```

➤ **Return value**

설정되어 있는 Selection Mask들의 정보를 가지고 있는 SelectionMask6cList Class의 Instance

➤ **Remarks**

RFID Module은 최대 8개까지 Selection Mask를 설정할 수 있다.

2.2.1.53. **setSelectionMask6cList**

setSelectionMask6cList 메서드는 RFID Module에 Selection Mask 값들의 List를 설정한다.

➤ **Syntax**

```
public void setSelectionMask6cList(SelectionMask6cList masks) throws ATRfidReaderException
```

➤ **Parameters**

masks : 설정하고자 하는 Selection Mask들의 정보를 가지고 있는 SelectionMask6cList Class의 Instance

➤ **Remarks**

RFID Module은 최대 8개까지 Selection Mask를 설정할 수 있다.

2.2.1.54. **getGlobalBand**

getGlobalBand메서드는 RFID Module의 국가 주파수 향질을 반환한다.

➤ **Syntax**

```
public GlobalBandType getGlobalBand() throws ATRfidReaderException
```

➤ **Return value**

국가 향질을 나타내는 GlobalBandType 열거형

➤ **Remarks**

주파수 향질을 나타내는 국가 정보를 반환한다.

2.2.1.55. **getFreqChannelCount**

getFreqChannelCount메서드는 RFID Module의 주파수 채널 테이블의 주파수 채널 최대 개수를 반환한다.

➤ **Syntax**

```
public int getFreqChannelCount() throws ATRfidReaderException
```

➤ **Return value**

주파수 채널 테이블의 최대 개수를 나타내는 Integer형

➤ **Remarks**

RFID Module은 국가 주파수 향질에 따라 여러 개의 주파수 채널을 지원한다. getFreqChannelCount메서드는 RFID Module에 저장되어 있는 주파수 채널 테이블의 최대 개수를 반환한다.

2.2.1.56. **isUseFreqChannel**

isUseFreqChannel메서드는 주파수 채널 중에 해당 인덱스의 채널을 사용 중인지 여부를 반환한다.

➤ **Syntax**

```
public boolean isUseFreqChannel(int index) throws ATRfidReaderException
```

➤ **Parameters**

index : 채널 사용여부를 반환 받으려는 주파수 채널 테이블의 인덱스를 나타내는 Integer형

➤ **Return value**

주파수 채널의 사용 여부를 나타내는 boolean형

➤ **Remarks**

주파수 채널 테이블에서 인덱스에 해당하는 주파수 채널의 사용 여부를 반환한다. 인덱스는 0보다 크고, getFreqChannelCount메서드로 반환 받은 값 보다 작아야 한다.

2.2.1.57. **setUseFreqChannel**

setUseFreqChannel메서드는 주파수 채널 중에 해당 인덱스의 채널을 사용할 것인지 여부를 설정한다.

➤ **Syntax**

```
public void setUseFreqChannel(int index, boolean isUsed) throws ATRfidReaderException
```

➤ **Parameters**

index : 채널 사용 여부를 설정하고자 하는 주파수 채널 테이블의 인덱스를 나타내는 Integer형

isUsed : 채널 사용 여부를 지정하는 Boolean형

➤ **Remarks**

주파수 채널 테이블에서 인덱스에 해당하는 주파수 채널의 사용 여부를 설정한다. 인덱스는 0보다 크고, getFreqChannelCount메서드로 반환 받은 값 보다 작아야 한다.

2.2.1.58. getChannelFreq

getChannelFreq메서드는 주파수 채널 중에 해당 인덱스의 주파수 값을 반환한다.

➤ Syntax

```
public int getChannelFreq(int index) throws ATRfidReaderException
```

➤ Parameters

index : 주파수 값을 반환 받고자 하는 주파수 채널 테이블의 인덱스를 나타내는 Integer형

➤ Return value

주파수 값을 나타내는 Integer형

➤ Remarks

주파수 채널 테이블에서 인덱스에 해당하는 주파수 값을 반환한다. 인덱스는 0보다 크고, getFreqChannelCount메서드를 반환 받은 값 보다 작아야 한다.

2.2.1.59. setEventListener

응용프로그램에서 이벤트를 사용할 수 있도록 설정한다.

➤ Syntax

```
public void setEventListener(RfidReaderEventListener listener)
```

➤ Parameters

listener : 응용프로그램에서 특정 이벤트(RFID)를 처리하도록 생성된 interface이다.

➤ Remarks

setEventListener메서드는 반드시 getInstance메서드 호출 이후에 사용하여야 한다.

2.2.1.60. removeEventListener

응용프로그램에서 이벤트를 사용할 수 없도록 설정한다.

➤ Syntax

```
public void removeEventListener(RfidReaderEventListener listener)
public void removeEventListener()
```

➤ Parameters

listener : 응용프로그램에서 특정 이벤트(RFID)를 처리하도록 생성된 interface이다.

2.3. ATRfidATX00S1Reader Class

ATRfidATX00S1Reader Class는 ATRfidReader를 상속한 Class이며, R2000 Module에 종속적인 기능들을 추가로 제공한다.

이 Class를 사용하기 위해서는 ATRfidManager Class의 getInstance Method로 반환된 ATRfidReader Instance를 ATRfidATX00S1Reader로 Type Casting 하면 된다.

2.3.1. Method

2.3.1.1. readMemory6cEx

지정된 시간만큼 CW를 유지하면서 Read Memory기능을 연속적 또는 단발적으로 수행한다.

➤ Syntax

```
public ResultCode readMemory6cEx(boolean isContinuousMode, BankType bank, int offset, int length, String password, EpcMatchParam epc, int delay);
```

➤ Parameters

isContinuousMode : 연속 수행 여부를 지정한다.

bank : Tag의 Memroy Bank를 지정한다.

offset : Tag Data의 시작 주소를 WORD 단위로 지정한다.

length : Tag Data의 길이를 WORD 단위로 지정한다.

password : Tag의 Access Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Read Memory를 수행할 때, 태그의 EPC 정보를 지정한다.

delay : CW를 유지 시킬 시간을 지정한다.(millisecond 단위)

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readMemory6c 메서드는 RFID Module 에서 ISO18000-6C Tag 에 대하여 Read Memory 기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener 의 onReaderResult Method 를 통하여 전달된다.

2.3.1.2. readMemory6cSync

readMemory6cEx와 동일한 기능을 하지만, 읽혀진 태그의 데이터를 Method 실행 결과로 반환한다.

➤ Syntax

```
public String readMemory6cSync(boolean isContinuousMode, BankType bank, int offset, int length, SelectionMask6cList masks, String password, EpcMatchParam epc, int delay);
```

➤ Parameters

isContinuousMode : 연속 수행 여부를 지정한다.

bank : Tag의 Memroy Bank를 지정한다.

offset : Tag Data의 시작 주소를 WORD 단위로 지정한다.

length : Tag Data의 길이를 WORD 단위로 지정한다.

masks : Read 할 태그의 selection mask list.

password : Tag의 Access Password를 4Byte Hex 문자열로 지정한다.

epc : 특정 태그에 대해서 Read Memory를 수행할 때, 태그의 EPC 정보를 지정한다.

delay : CW를 유지 시킬 시간을 지정한다.(millisecond 단위)

➤ **Return value**

Read memory가 정상적으로 수행되었다면 태그 데이터를 반환하고 실패/에러가 발생했을 경우에는 null이 반환 된다.

➤ **Remarks**

이 method는 기능 수행이 끝날 때까지 리턴하지 않으므로, setOperationTime method를 사용해서 기능 수행 시간을 설정해 놓고 사용하기를 권장한다.

2.3.1.3. readOemData

R2000 Module의 OEM Register 값을 읽는다.

➤ **Syntax**

```
public int readOemData(int address);
```

➤ **Parameters**

address : access 할 OEM Register 주소 (32bits value)

➤ **Return value**

입력된 address의 값 (32bits value)

➤ **Remarks**

R2000 Module의 Register 값을 직접 읽는 것이므로, 일반적인 용도로는 사용하지 않는다.

2.3.1.4. writeOemData

R2000 Module의 OEM Register 에 값을 기록한다.

➤ **Syntax**

```
public boolean writeOemData(int address, int value);
```

➤ **Parameters**

address : access 할 OEM Register 주소 (32bits value)

value : address에 write할 값 (32bits value)

➤ **Return value**

Method 실행의 성공 여부를 나타내는 boolean형

➤ **Remarks**

R2000 Module의 Register 값을 직접 읽는 것이므로, 일반적인 용도로는 사용하지 않는다.

2.3.1.5. isUseLinkProfile

Link Profile의 사용 여부를 반환한다.

➤ **Syntax**

```
public boolean isUseLinkProfile();
```

➤ **Return value**

Link Profile의 사용 여부를 나타내는 boolean형

2.3.1.6. getLinkProfileCount

Link Profile의 개수를 반환한다.

➤ **Syntax**

```
public int getLinkProfileCount() throws ATRfidReaderException
```

➤ **Return value**

Link Profile의 개수를 나타내는 Integer형

2.3.1.7. getActiveLinkProfile

현재 사용중인 Link Profile의 Index를 반환한다.

➤ **Syntax**

```
public int getActiveLinkProfilelet() throws ATRfidReaderException
```

➤ **Return value**

Link Profile의 Index를 나타내는 Integer형

2.3.1.8. setActiveLinkProfile

현재 사용중인 Link Profile의 Index를 변경한다.

➤ **Syntax**

```
public boolean setActiveLinkProfilelet(int index) throws ATRfidReaderException
```

➤ **Parameters**

index : 변경할 Link Profile index

➤ **Return value**

Method 실행의 성공 여부를 나타내는 boolean형

2.3.1.9. getUseDefaultLinkProfile

Default Link Profile 사용 여부 값을 반환한다.

➤ **Syntax**

```
public boolean getUseDefaultLinkProfilelet(int index) throws ATRfidReaderException
```

➤ **Parameters**

index : Link Profile index

➤ **Return value**

값이 1이면 Default Link Profile 값을 사용하고, 0이면 F/W의 default 값을 사용한다.

2.3.1.10. **setDefaultLinkProfile**

Default Link Profile의 Index를 설정 한다.

➤ **Syntax**

```
public boolean setDefaultLinkProfile(int index, int used) throws
ATRFidReaderException
```

➤ **Parameters**

index : Default로 설정할 Link Profile index

used : index로 설정한 값을 Default로 사용할지를 결정 할 값(1:사용, 0:사용 안 함)

➤ **Return value**

Method 실행의 성공 여부를 나타내는 boolean형

2.3.1.11. **setCarrierWaveDelay**

CW의 유지 시간을 설정한다.

➤ **Syntax**

```
public boolean setCarrierWaveDelay(int delay) throws ATRFidReaderException
```

➤ **Parameters**

delay : 유지 시간(0~255 millisecond 단위)

➤ **Return value**

Method 실행의 성공 여부를 나타내는 boolean형

2.3.1.12. **getCurrentSingulationAlgorithm**

inventory에 사용하는 q 알고리즘을 반환한다.(default: DYNAMICQ)

➤ **Syntax**

```
public SingulationAlgorithm getCurrentSingulationAlgorithm() throws
ATRFidReaderException
```

➤ **Return value**

현재 사용중인 알고리즘을 나타내는 SingulationAlgorithm 형.

2.3.1.13. **setCurrentSingulationAlgorithm**

inventory에 사용하는 q 알고리즘을 설정 한다.

➤ **Syntax**

```
public void setCurrentSingulationAlgorithm(SingulationAlgorithm algorithm) throws
ATRFidReaderException
```

➤ **Parameters**

algorithm : inventory에 사용할 q 알고리즘

➤ **Remarks**

RFID 성능에 영향이 있으므로 사용에 주의가 필요 하다.

2.3.1.14. **getQValue**

inventory에 사용하는 Q 값을 반환한다.

➤ **Syntax**

```
public QValue getQValue() throws ATRfidReaderException
```

➤ **Return value**

Q 값을 나타내는 QValue 형.

2.3.1.15. **setQValue**

inventory에 사용하는 Q 값을 설정한다.

➤ **Syntax**

```
public void setQValue(QValue q) throws ATRfidReaderException
```

➤ **Parameters**

q : 변경할 Q Value

➤ **Remarks**

RFID 성능에 영향이 있으므로 사용에 주의가 필요 하다.

FIXEDQ를 사용한다면 start q 만을 고정으로 사용하며,

DYNAMICQ를 사용한다면, start, min, max q를 동적으로 사용한다.

이 값들은 아래와 같은 조건으로 설정되어 있어야 한다.

- Start q는 min q 보다 크거나 같고 max q 보다 작거나 같아야 한다.
- Min q는 start q, max q 보다 작거나 같아야 한다.
- Max q는 start q, min q 보다 크거나 같아야 한다.

2.4. ATRfid900MAREader Class

ATRfid900MAREader Class는 ATRfidReader를 상속한 Class이며, AMS Module에 종속적인 기능들을 추가로 제공한다.

이 Class를 사용하기 위해서는 ATRfidManager Class의 getInstance Method로 반환된 ATRfidReader Instance를 ATRfid900MAREader 로 Type Casting 하면 된다.

2.4.1. Method

2.4.1.1. readEpcRailTag

Rail 태그를 Single 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode readEpcRailTag();
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readEpcRailTag 메서드는 RFID Module에서 Rail Tag에 대하여 Inventory 기능을 수행하도록 하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener의 onReaderReadTag Method를 통해서 태그 데이터가 전달 된다.

이 method를 사용하기 위해서는 Rail Tag 전용 F/W가 적용된 Module을 사용해야 한다. getFirmwareVersion Method로 반환된 버전이 'R'로 시작되면 Rail Tag 전용 F/W이며 'M'으로 시작된다면 Standard F/W 이다.

2.4.1.2. inventoryRailTag

Rail 태그를 Multiple 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode inventoryRailTag();
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

inventoryRailTag 메서드는 RFID Module에서 Rail Tag에 대하여 Inventory 기능을 수행하도록 하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener의 onReaderReadTag Method를 통해서 태그 데이터가 전달 된다.

이 method를 사용하기 위해서는 Rail Tag 전용 F/W가 적용된 Module을 사용해야 한다. getFirmwareVersion Method로 반환된 버전이 'R'로 시작되면 Rail Tag 전용 F/W이며 'M'으로 시작된다면 Standard F/W 이다.

2.4.1.3. readEpcAnyTag

태그 타입을 지정하지 않고 single 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode readEpcAnyTag();
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

readEpcAnyTag 메서드는 RFID Module에서 ISO 18000-6B, ISO 18000-6C, Rail Tag에 대하여 Inventory 기능을 수행 하도록 하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener의 onReaderReadTag Method를 통해서 태그 데이터가 전달 된다. Standard F/W가 적용되어 있다면 이 method가 동작은 하겠지만, Rail Tag에 대해서는 반응하지 않는다.(6B, 6C 태그에 대해서만 반응함)
getFirmwareVersion Method로 반환된 버전이 'R'로 시작되면 Rail Tag 전용 F/W이며 'M'으로 시작된다면 Standard F/W 이다.

2.4.1.4. inventoryAnyTag

태그 타입을 지정하지 않고 multiple 모드로 Inventory 한다.

➤ Syntax

```
public ResultCode inventoryAnyTag();
```

➤ Return value

명령어 수행 결과를 ResultCode 열거형으로 반환한다.

➤ Remarks

inventoryAnyTag 메서드는 RFID Module에서 ISO 18000-6B, ISO 18000-6C, Rail Tag에 대하여 Inventory 기능을 수행 하도록 하며 정상적으로 태그를 읽었을 경우, RfidReaderEventListener의 onReaderReadTag Method를 통해서 태그 데이터가 전달 된다. Standard F/W가 적용되어 있다면 이 method가 동작은 하겠지만, Rail Tag에 대해서는 반응하지 않는다. (6B, 6C 태그에 대해서만 반응함)
getFirmwareVersion Method로 반환된 버전이 'R'로 시작되면 Rail Tag 전용 F/W이며 'M'으로 시작된다면 Standard F/W 이다.

2.4.1.5. readMemory6b

Tag를 선택해서 ISO 18000-6B Read Memory기능을 수행한다.

➤ Syntax

```
public ResultCode readMemory6b(int offset, int length, SelectionMask6b mask);
```

➤ Parameters

offset : Read를 수행할 Tag Data의 시작 주소를 BYTE 단위로 지정한다.

length : Read를 수행할 Tag Data의 길이를 BYTE 단위로 지정한다.

mask : Read가 수행되기를 원하는 Tag의 메모리 정보를 지정한다.

➤ **Return value**

명령어 수행 결과를 ResultCode열거형으로 반환한다.

➤ **Remarks**

readMemory6b메서드는 RFID Module에서 ISO18000-6B Tag에 대하여 Read Memory기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다.

2.4.1.6. writeMemory6b

Tag를 선택해서 ISO 18000-6B Write Memory기능을 수행한다.

➤ **Syntax**

```
public ResultCode writeMemory6b(int offset, String data, SelectionMask6b mask);
```

➤ **Parameters**

offset : Write를 수행할 Tag Data의 시작 주소를 BYTE 단위로 지정한다.

data : Tag 기록 하고자 하는 Data를 BYTE 단위 Hex 문자열로 지정한다.

mask : Write가 수행되기를 원하는 Tag의 메모리 정보를 지정한다.

➤ **Return value**

명령어 수행 결과를 ResultCode열거형으로 반환한다.

➤ **Remarks**

writeMemory6b메서드는 RFID Module에서 ISO18000-6B Tag에 대하여 Write Memory기능을 수행하며 정상적으로 수행되었을 경우, 결과는 RfidReaderEventListener의 onReaderResult Method를 통하여 전달된다

2.5. RfidReaderEventListener Interface

2.5.1. Method

2.5.1.1. onReaderStateChanged

onReaderStateChange메서드는 RFID Module의 연결 상태를 반환한다.

➤ **Syntax**

```
void onReaderStateChanged(ATRfidReader reader, ConnectionState state);
```

➤ **Parameters**

reader : event를 발생시킨 Reader Object

state : RFID의 연결 상태를 나타내는 ConnectionState 열거형

➤ **Remarks**

연결 상태가 변경되면 RFID Module과 연결된 Reader Object에서 호출된다.

2.5.1.2. onReaderActionChanged

onReaderActionChange메서드는 RFID Module의 동작상태를 반환한다.

➤ **Syntax**

```
void onReaderActionChanged(ATRfidReader reader, ActionState action);
```

➤ **Parameters**

reader : event를 발생시킨 Reader Object

state : RFID의 동작 상태를 나타내는 ActionState 열거형

➤ **Remarks**

동작 상태가 변경되면 RFID Module과 연결된 Reader Object에서 호출된다.

2.5.1.3. onReaderReadTag

onReaderReadTag메서드는 readEpc6cTag메서드나 inventory6c메서드로 읽은 Tag의 EPC 값을 반환한다.

➤ **Syntax**

```
void onReaderReadTag(ATRfidReader reader, String tag, float rssi, float phase);
```

➤ **Parameters**

reader : event를 발생시킨 Reader Object

tag : Inventory 로 읽은 Tag의 EPC를 나타내는 Hex형 문자열

rssi : RSSI 값을 나타내는 float형 값

phase : Phase 값을 나타내는 float형 값

➤ **Remarks**

RFID Module이 Inventory기능으로 Tag의 EPC데이터를 읽으면 Reader Object에서 호출된다.

2.5.1.4. onReaderResult

onReaderResult메서드는 Read Memory나 Write Memory, Lock, Kill 등의 Access 명령을 수행한 결과를 반환한다.

➤ Syntax

```
void onReaderResult(ATRfidReader reader, ResultCode code, ActionState action, String epc, String data, float rssi, float phase);
```

➤ Parameters

reader : event를 발생시킨 Reader Object

code : Access 명령의 수행 결과를 나타내는 ResultCode 열거형

action : 수행한 Access명령을 나타내는 ActionState 열거형

epc : Access Tag의 EPC 데이터를 나타내는 Hex형 문자열

data : 수행한 Access명령이 ReadMemory인 경우 읽은 Tag 데이터를 나타내는 Hex형 문자열

rssi : RSSI 값을 나타내는 float형 값

phase : Phase 값을 나타내는 float형 값

➤ Remarks

Access 관련 명령이 수행되면 RFID Module 과 연결된 Reader Object 에서 호출된다.

2.6. Parameter Classes

2.6.1. RangeValue Class

2.6.1.1. Constructor

범위를 나타내는 RangeValue의 새 Instance를 초기화 한다.

➤ Syntax

```
public RangeValue()  
public RangeValue(int min, int max)
```

➤ Parameters

min : 최소값을 나타내는 정수형

max : 최대값을 나타내는 정수형

➤ Remarks

getPowerRange 등에서 반환되는 범위를 나타내는 값 등에 사용된다.

2.6.1.2. Property Methods

2.6.1.2.1. getMin

설정된 범위의 최소값을 반환한다.

➤ Syntax

```
public int getMin()
```

➤ Return value

Instance에 설정되어 있는 최소값을 나타내는 정수형

2.6.1.2.2. getMax

설정된 범위의 최대값을 반환한다.

➤ Syntax

```
public int getMax()
```

➤ Return value

Instance에 설정되어 있는 최대값을 나타내는 정수형

2.6.2. LockParam Class

2.6.2.1. Constructor

범위를 나타내는 LockParam의 새 Instance를 초기화 한다.

➤ Syntax

```
public LockParam()  
public LockParam(LockType killPassword, LockType accessPassword,  
LockType epc, LockType tid, LockType user)
```

➤ Parameters

killPassword : Kill Password 영역의 Lock동작을 나타내는 LockType 열거형.

accessPassword : Access Password 영역의 Lock동작을 나타내는 LockType 열거형.

epc : EPC Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

tid : TID Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

user : User Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

➤ **Remarks**

lock6c 메시드의 Parameter 로 사용한다.

2.6.2.2. Property Methods

2.6.2.2.1. getKillPassword

Kill Passwrod 영역의 Lock 동작을 반환한다.

➤ **Syntax**

```
public LockType getKillPassword()
```

➤ **Return value**

Kill Password영역의 Lock동작을 나타내는 LockType 열거형

2.6.2.2.2. setKillPassword

Kill Password 영역의 Lock동작을 설정한다.

➤ **Syntax**

```
public void setKillPassword(LockType killPassword)
```

➤ **Parameters**

killPassword : Kill Password 영역의 Lock동작을 나타내는 LockType 열거형.

2.6.2.2.3. getAccessPassword

Access Passwrod 영역의 Lock 동작을 반환한다.

➤ **Syntax**

```
public LockType getAccessPassword()
```

➤ **Return value**

Access Password영역의 Lock동작을 나타내는 LockType 열거형

2.6.2.2.4. setAccessPassword

Access Password 영역의 Lock동작을 설정한다.

➤ **Syntax**

```
public void setAccessPassword(LockType accessPassword)
```

➤ **Parameters**

accessPassword : Access Password 영역의 Lock동작을 나타내는 LockType 열거형.

2.6.2.2.5. getEPC

EPC Memory Bank 영역의 Lock 동작을 반환한다.

➤ Syntax

```
public LockType getEPC()
```

➤ Return value

EPC Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형

2.6.2.2.6. setEPC

EPC Memory Bank 영역의 Lock동작을 설정한다.

➤ Syntax

```
public void setEPC(LockType epc)
```

➤ Parameters

epc : EPC Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

2.6.2.2.7. getTID

TID Memory Bank 영역의 Lock 동작을 반환한다.

➤ Syntax

```
public LockType getTID()
```

➤ Return value

TID Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형

2.6.2.2.8. setTID

TID Memory Bank 영역의 Lock동작을 설정한다.

➤ Syntax

```
public void setTID(LockType tid)
```

➤ Parameters

tid : TID Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

2.6.2.2.9. getUser

User Memory Bank 영역의 Lock 동작을 반환한다.

➤ Syntax

```
public LockType getUser()
```

➤ Return value

User Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형

2.6.2.2.10. setUser

User Memory Bank 영역의 Lock동작을 설정한다.

➤ **Syntax**

```
public void setUser(LockType user)
```

➤ **Parameters**

user : User Memory Bank 영역의 Lock동작을 나타내는 LockType 열거형.

2.6.3. SelectionMask6c Class

2.6.3.1. Constructor

Selection Mask를 나타내는 SelectionMask6c Class의 새 Instance를 초기화 한다.

➤ **Syntax**

```
public SelectionMask6c()
public SelectionMask6c(MaskTargetType target, MaskActionType action,
    BankType bank, int pointer, int length, String mask,
    boolean truncate)
```

➤ **Parameters**

target : Mask 대상이 되는 Tag의 Session을 나타내는 MaskTargetType 열거형

action : Mask 조건에 대한 Session설정을 결정하는 MaskActionType 열거형

bank : Mask 조건의 대상이 되는 Tag의 Memory Bank를 나타내는 BankType 열거형

pointer : Mask 값을 비교하기 시작할 시작 주소를 나타내는 정수형 (bit단위)

length : Mask값을 비교할 길이를 지정하는 정수형 (bit단위)

mask : Mask값을 나타내는 Hex형 문자열

truncate : Mask값을 길이 만큼 잘라낼지 여부를 나타내는 boolean형

➤ **Remarks**

getSelectionMask6c 나 setSelectionMask6c 메소드에서 사용한다.

2.6.3.2. Property Methods

2.6.3.2.1. isUsed

현재 설정된 Selection Mask 조건을 사용할 것인지 여부를 반환한다.

➤ **Syntax**

```
public boolean isUsed()
```

➤ **Return value**

Selection Mask정보를 사용할 것인지 여부를 나타내는 boolean형

2.6.3.2.2. setUsed

현재 설정된 Selection Mask 조건을 사용할 것인지 여부를 설정한다.

➤ **Syntax**

```
public void setUsed(boolean used)
```

➤ **Parameters**

used : Selection Mask정보를 사용할 것인지 여부를 나타내는 boolean형

2.6.3.2.3. getTarget

Selection Mask 대상이되는 Tag의 Session을 반환한다.

➤ **Syntax**

```
public MaskTargetType getTarget()
```

➤ **Return value**

Mask 대상이 되는 Tag의 Session을 나타내는 MaskTargetType 열거형

2.6.3.2.4. setTarget

Selection Mask 대상이되는 Tag의 Session을 설정한다.

➤ **Syntax**

```
public void setTarget(MaskTargetType target)
```

➤ **Parameters**

target : Mask 대상이 되는 Tag의 Session을 나타내는 MaskTargetType 열거형

2.6.3.2.5. getAction

Selection Mask 조건에 대한 Session설정의 방법을 반환한다.

➤ **Syntax**

```
public MaskActionType getAction()
```

➤ **Return value**

Mask 조건에 대한 Session설정을 결정하는 MaskActionType 열거형

2.6.3.2.6. setAction

Selection Mask 조건에 대한 Session설정의 방법을 설정한다.

➤ **Syntax**

```
public void setAction(MaskActionType action)
```

➤ **Parameters**

action : Mask 조건에 대한 Session설정을 결정하는 MaskActionType 열거형

2.6.3.2.7. getBank

Selection Mask의 비교 대상이 되는 Tag Memory Bank를 반환한다.

➤ **Syntax**

```
public BankType getBank()
```

➤ **Return value**

Mask 조건의 대상이 되는 Tag의 Memory Bank를 나타내는 BankType 열거형

2.6.3.2.8. setBank

Selection Mask의 비교 대상이 되는 Tag Memory Bank를 설정한다.

➤ **Syntax**

```
public void setBank(BankType bank)
```

➤ **Parameters**

- **bank** : Mask 조건의 대상이 되는 Tag의 Memory Bank를 나타내는 BankType 열거형

2.6.3.2.9. getPointer

Selection Mask의 Mask값을 비교하기 시작할 시작 주소를 반환한다.

➤ **Syntax**

```
public int getPointer()
```

➤ **Return value**

Mask 값을 비교하기 시작할 시작 주소를 나타내는 정수형 (bit단위)

2.6.3.2.10. setPointer

Selection Mask의 Mask값을 비교하기 시작할 시작 주소를 설정한다.

➤ **Syntax**

```
public void setPointer(int pointer)
```

➤ **Parameters**

- **pointer** : Mask 값을 비교하기 시작할 시작 주소를 나타내는 정수형 (bit단위)

2.6.3.2.11. getLength

Selection Mask의 Mask값을 비교할 길이를 반환한다.

➤ **Syntax**

```
public int getLength()
```

➤ **Return value**

Mask값을 비교할 길이를 지정하는 정수형 (bit단위)

2.6.3.2.12. setLength

Selection Mask의 Mask값을 비교할 길이를 설정한다.

➤ **Syntax**

```
public void setLength(int length)
```

➤ **Parameters**

- **length** : Mask값을 비교할 길이를 지정하는 정수형 (bit단위)

2.6.3.2.13. getMask

Selection Mask의 비교 대상이 될 Mask값을 반환한다.

➤ **Syntax**

```
public String getMask()
```

➤ **Return value**

Mask값을 나타내는 Hex형 문자열

2.6.3.2.14. setMask

Selection Mask의 비교 대상이 될 Mask값을 설정한다.

➤ **Syntax**

```
public void setMask(String mask)
```

➤ **Parameters**

- **mask** : Mask값을 나타내는 Hex형 문자열

2.6.3.2.15. getTruncate

Selection Mask의 Mask값을 길이 만큼 자를 것인지 여부를 반환한다.

➤ **Syntax**

```
public boolean getTruncate()
```

➤ **Return value**

Mask값을 길이 만큼 잘라낼지 여부를 나타내는 boolean형

2.6.3.2.16. setTruncate

Selection Mask의 Mask값을 길이 만큼 자를 것인지 여부를 설정한다.

➤ **Syntax**

```
public void setTruncate(boolean truncate)
```

➤ **Parameters**

- **truncate** : Mask값을 길이 만큼 잘라낼지 여부를 나타내는 boolean형

2.6.4. SelectionMask6b Class

2.6.4.1. Constructor

Selection Mask를 나타내는 SelectionMask6b Class의 새 Instance를 초기화 한다.

➤ **Syntax**

```
public SelectionMask6b()  
public SelectionMask6b(int pointer, String mask, MaskActionType action)
```

➤ **Parameters**

pointer : Mask 값을 비교하기 시작할 주소. 항상 0으로 설정해야 한다.

mask : Mask값을 나타내는 Hex형 문자열. 항상 UID(64bits) 전체를 입력해야 한다.

action : Mask 조건. 항상 MaskMatchingType.Match로 설정해야 한다.

➤ **Remarks**

readMemory6b, writeMemory6c method 에서 사용한다.

2.6.4.2. Property Methods

2.6.4.2.1. getPointer

현재 설정된 Selection Mask의 pointer 값을 반환한다.

➤ **Syntax**

```
public int getPointer()
```

➤ **Return value**

Mask 값을 비교하기 시작할 시작 주소를 나타내는 정수형

2.6.4.2.2. setPointer

Selection Mask의 pointer 값을 설정한다.

➤ **Syntax**

```
public void setPointer(int pointer)
```

➤ **Parameters**

pointer : Mask 값을 비교하기 시작할 시작 주소를 나타내는 정수형

2.6.4.2.3. getAction

Selection Mask 조건을 반환한다.

➤ **Syntax**

```
public MaskActionType getAction()
```

➤ **Return value**

Mask 조건을 결정하는 MaskMatchingType 열거형

2.6.4.2.4. setAction

Selection Mask 조건을 설정한다.

➤ **Syntax**

```
public void setAction(MaskActionType action)
```

➤ **Parameters**

action : Mask 조건을 결정하는 MaskMatchingType 열거형

2.6.4.2.5. getMask

Selection Mask의 비교 대상이 될 Mask값을 반환한다.

➤ **Syntax**

```
public String getMask()
```

➤ **Return value**

Mask값을 나타내는 Hex형 문자열

2.6.4.2.6. setMask

Selection Mask의 비교 대상이 될 Mask값을 설정한다.

➤ **Syntax**

```
public void setMask(String mask)
```

➤ **Parameters**

- **mask** : Mask값을 나타내는 Hex형 문자열

2.6.5. EpcMatchParam Class

2.6.5.1. Constructor

EpcMatchParam의 새 Instance를 초기화 한다.

➤ **Syntax**

```
public EpcMatchParam()  
public EpcMatchParam(MaskMatchingType match, int offset, int length, String data)
```

➤ **Parameters**

match : tag 값과 data가 matching/non-matching 하는지를 설정.

offset : tag 값과 비교될 data가 시작되는 offset(EPC 시작 위치가 0이며 bit 단위이다.)

length : tag 값과 비교될 data의 길이 (bit 단위)

data : tag 값과 비교될 data

➤ **Remarks**

Read/write/lock/kill 등의 Access 명령에 사용된다.

2.6.5.2. Property Methods

2.6.5.2.1. getMatch

설정된 MaskMatchingType을 반환한다.

➤ **Syntax**

```
public MaskMatchingType getMatch()
```

➤ **Return value**

Instance에 설정되어 있는 MaskMatchingType

2.6.5.2.2. setMatch

MaskMatchingType 설정한다.

➤ **Syntax**

```
public void setMatch(MaskMatchingType match)
```

➤ **Parameters**

match : MaskMatchingType

2.6.5.2.3. getOffset

설정된 data의 offset 값을 반환한다.

➤ **Syntax**

```
public int getOffset()
```

➤ **Return value**

Instance에 설정되어 있는 offset을 나타내는 정수형 (bit단위)

2.6.5.2.4. setValue

data의 offset 값을 설정한다.

➤ **Syntax**

```
public int setValue(int offset)
```

➤ **Parameters**

offset : tag 값과 비교될 data가 시작되는 offset(EPC 시작 위치가 0이며 bit 단위이다.)

2.6.5.2.5. getLength

설정된 data의 length 값을 반환한다.

➤ **Syntax**

```
public int getOffset()
```

➤ **Return value**

Instance에 설정되어 있는 length를 나타내는 정수형 (bit 단위)

2.6.5.2.6. setLength

data의 length 값을 설정한다.

➤ **Syntax**

```
public void setLength(int length)
```

➤ **Parameters**

length : tag 값과 비교될 data의 길이 (bit 단위)

2.6.5.2.7. getData

Instance에 설정된 data를 반환한다.

➤ **Syntax**

```
public String getData()
```

➤ **Return value**

tag 값과 비교될 data

2.6.5.2.8. setData

Selection Mask의 Mask값을 비교하기 시작할 시작 주소를 설정한다.

➤ Syntax

```
public void setData(String data)
```

➤ Parameters

data : tag 값과 비교될 data

2.6.5.2.9. getValue

Instance에 설정된 값들을 RFID Module에 전달될 값으로 변환해서 반환한다.

➤ Syntax

```
public int getValue()
```

➤ Return value

변환 된 설정 값.

➤ Remarks

라이브러리 내부에서 사용하는 method 이므로, 임의로 사용하면 안 된다.

2.6.6. QValue Class

2.6.6.1. Constructor

QValue의 새 Instance를 초기화 한다.

➤ Syntax

```
public QValue()  
public QValue(int start, int min, int max)
```

➤ Parameters

start : starting q value

min : minimum q value (SingulationAlgorithm이 FIXED 일 때는 해당 없음.)

max : maximum q value(SingulationAlgorithm이 FIXED 일 때는 해당 없음.)

➤ Remarks

RFID 성능에 영향이 있으므로 사용에 주의가 필요 하다.

2.6.6.2. Property Methods

2.6.6.2.1. getStartQ

설정된 starting q value를 반환한다.

➤ Syntax

```
public int getStartQ()
```

➤ Return value

Instance에 설정되어 있는 starting q value

➤ **Remarks**

SingulationAlgorithm으로 FIXEDQ를 사용할 때는, start q value를 고정으로 사용한다.
DYNAMICQ를 사용할 때에 start q는 min q 보다 크거나 같아야 하며, max q 보다 작거나 같아야 한다.

2.6.6.2.2. setStartQ

starting q value를 설정한다.

➤ **Syntax**

```
public void setStartQ(int start)
```

➤ **Parameters**

start : 설정 할 start q 값(0~15)

➤ **Remarks**

SingulationAlgorithm으로 FIXEDQ를 사용할 때는, start q value를 고정으로 사용한다.
DYNAMICQ를 사용할 때에 start q는 min q 보다 크거나 같아야 하며, max q 보다 작거나 같아야 한다.

2.6.6.2.3. getMinQ

설정된 minimum q value를 반환한다.

➤ **Syntax**

```
public int getMinQ()
```

➤ **Return value**

Instance에 설정되어 있는 minimum q value

➤ **Remarks**

SingulationAlgorithm으로 DYNAMICQ를 사용할 때에 유효하며,
Min q는 start q, max q 보다 작거나 같아야 한다.

2.6.6.2.4. setMinQ

minimum q value를 설정한다.

➤ **Syntax**

```
public void setMinQ(int min)
```

➤ **Parameters**

start : 설정 할 minimum q 값(0~15)

➤ **Remarks**

SingulationAlgorithm으로 DYNAMICQ를 사용할 때에 유효하며,
Min q는 start q, max q 보다 작거나 같아야 한다.

2.6.6.2.5. getMaxQ

설정된 maximum q value를 반환한다.

➤ Syntax

```
public int getMaxQ()
```

➤ Return value

Instance에 설정되어 있는 maximum q value

➤ Remarks

SingulationAlgorithm으로 DYNAMICQ를 사용할 때에 유효하며,
Max q는 start q, min q 보다 크거나 같아야 한다.

2.6.6.2.6. setMaxQ

maximum q value를 설정한다.

➤ Syntax

```
public void setMaxQ(int max)
```

➤ Parameters

start : 설정 할 maximum q 값(0~15)

➤ Remarks

SingulationAlgorithm으로 DYNAMICQ를 사용할 때에 유효하며,
Max q는 start q, min q 보다 크거나 같아야 한다.

2.7. Enumerations

2.7.1. ActionState

현재 RFID Module의 동작 상태를 나타낸다.

플래그	값	설명
Unknown	0x20	알 수 없는 상태
Inventory6bMulti	0x62	Multiple Inventory 6B Type
Inventory6bSingle	0x61	Single Inventory 6B Type
Inventory6cMulti	0x66	Multiple Inventory 6C Type
Inventory6cSingle	0x65	Single Inventory 6C Type
Inventory6cSelect	0x64	Select Inventory 6C Type
InventoryAnyMulti	0x6B	Multiple Inventory Any Type
ReadMemory6c	0x72	Read Memory 6C Type
ReadMemory6b	0x52	Read Memory 6B Type
WriteMemory6c	0x77	Write Memory 6C Type
WriteMemory6b	0x63	Write Memory 6B Type
Lock	0x6C	Lock Tag
Kill	0x6B	Kill Tag
Stop	0x33	Stop

2.7.2. BankType

Tag의 Memory Bank를 지정한다.

플래그	값	설명
Reserved	0	RESERVED Bank
EPC	1	EPC Bank
TID	2	TID Bank
User	3	User Bank

2.7.3. TagType

Inventory Tag type를 지정한다

Flag	Value	Description
Tag6C	0	ISO18000 6C
Tag6B	1	ISO18000 6B
TagRail	2	AEI/Rail
TagAny	3	N/A

2.7.4. ConnectionState

RFID Module과 Reader Object와 연결 상태를 나타낸다.

플래그	값	설명
Disconnected	0	연결이 끊어짐
Connecting	2	연결 중
Connected	3	연결 됨

2.7.5. InventorySession

Inventory 수행 시, 대상 Tag의 Session을 지정한다.

플래그	값	설명
S0	0	Session 0
S1	1	Session 1
S2	2	Session 2
S3	3	Session 3

2.7.6. InventoryTarget

Inventory 수행 시, Inventory 대상이 되는 Tag의 Session 상태를 나타낸다.

플래그	값	설명
A	0	Session 상태 A
B	1	Session 상태 B
AB	2	A or B

2.7.7. LockType

지정된 영역에 Lock동작을 나타낸다.

플래그	값	설명
NoChange	0	아무런 동작을 하지 않음.
Unlock	1	Unlock 수행
Lock	2	Lock 수행
PermaLock	3	영구 Lock 수행

2.7.8. RfidModuleType

현재 사용중인 RFID Module을 나타낸다.

플래그	값	설명
None	0	Module 없음.
I900MA	1	AMS Module (supported ISO 18000 6B/6C)



RFID API Reference Guide for Android Developers

Android Developer Guide

회사

ATID Co.,Ltd

문서이름

작성자

SW Team

날자

2022-06-20

버전

V1.1

AT6EM_1	2	Not supported
AT9200P_1	3	Not supported
ATX00S_1	4	R2000 Module (supported ISO 18000 6C)

ATID CO.,Ltd

2.7.9. SelectFlagType

Inventory 수행 시, Inventory 대상이 되는 Tag의 SL 상태를 나타낸다.

플래그	값	설명
All	0	assert 또는 deassert
Deassert	1	deassert
Assert	2	assert

2.7.10. MaskMatchingType

입력한 mask와 태그의 데이터를 비교했을 때, matching/non-matching 상태를 나타낸다.

플래그	값	설명
Match	0x30	Matching
NonMatch	0x31	Non-Matching

2.7.11. GlobalBandType

모듈의 국가 주파수 항질에 대한 국가를 나타낸다.

플래그	값	설명
Korea	0	한국 주파수
Europe	1	유럽 주파수
NorthAmerica	2	북아메리카 주파수
China	3	중국 주파수
Taiwan	4	타이완 주파수
Brazil	5	브라질 주파수
Malaysia	6	말레이시아 주파수
Hongkong	7	홍콩 주파수
Japan1W	8	일본 (1W) 주파수
Japan250mW	9	일본 (250mW) 주파수
India	10	인도 주파수
Indonesia	11	인도네시아 주파수
Japan125mW	12	일본 (125mW) 주파수
Israel	13	이스라엘 주파수
Australia	14	오스트레일리아 주파수
Newzealand	15	뉴질랜드 주파수
Philippines	16	필리핀 주파수
Singapore	17	싱가폴 주파수

Thailand	18	태국 주파수
Uruguay	19	우루과이 주파수
Vietnam	20	베트남 주파수
SouthAfrica	21	남아프리카 주파수
Morocco	22	모로코 주파수
Europe_B1	23	EN 302 208 Sub-bands b1
Europe_B2	24	EN 302 208 Sub-bands b2
Europe_B3	25	EN 302 208 Sub-bands b3
Peru	26	페루 주파수

2.7.12. MaskActionType

6c 타입의 태그에 Selection Mask를 사용할 때, 조건에 부합하는 태그의 Session 상태를 변경할 동작 방법을 나타낸다.

플래그	값	설명
Assert_Deassert	0	Matcing : assert SL or inventoried → A Not Matcing : deassert SL or inventoried → B
Assert_DoNothing	1	Matcing : assert SL or inventoried → A Not Matcing : do nothing
DoNothing_Deassert	2	Matcing : do nothing Not Matcing : deassert SL or inventoried → B
Negate_DoNothing	3	Matcing : negate SL or (A → B, B → A) Not Matcing : do nothing
Deassert_Assert	4	Matcing : deassert SL or inventoried → B Not Matcing : assert SL or inventoried → A
Deassert_DoNothing	5	Matcing : deassert SL or inventoried → B Not Matcing : do nothing
DoNothing_Assert	6	Matcing : do nothing Not Matcing : assert SL or inventoried → A
DoNothing_Negate	7	Matcing : do nothing Not Matcing : negate SL or (A → B, B → A)

2.7.13. MaskTargetType

6c 타입의 태그에 Selection Mask를 사용할 때, 조건에 부합하는 태그가 조작될 Session 을 지정한다.

플래그	값	설명
-----	---	----

S0	0	Session 0
S1	1	Session 1
S2	2	Session 2
S3	3	Session 3
SL	4	Session Flag

2.7.14. SingulationAlgorithm

Inventory에 사용할 Q 관리 알고리즘

플래그	값	설명
FIXEDQ	0	Start Q를 고정 Q값으로 사용
DYNAMICQ	1	Min, Max 값에 의한 동적 관리.

2.7.15. ResultCode

본 SDK에서 메서드나 이벤트의 동작 결과를 나타낸다.

플래그	값	설명
NoError	0x0000	No error
OtherError	0x0001	Other error
Undefined	0x0002	Undefined
MemoryOverrun	0x0003	Memory overrun
MemoryLocked	0x0004	Memory locked
InsufficientPower	0x000B	Insufficient power
NonSpecificError	0x000F	Non-Specific error
InvalidResponse	0xE001	Invalid response
InOperation	0xE002	In operation
OutOfRange	0xE003	Out of range
NotConnected	0xE004	Disconnected
InvalidParameter	0xE010	Invalidate parameter
InvalidInstance	0xE100	Invalid instance
FailSendControlPacket	0xEE00	Failed to send control packet
FailReceivePacket	0xEE01	Failed to receive packet
InvalidControlResponse	0xEE02	Invalidate control response packet
UnknownControlResponse	0xEE0F	Unknown control response
InvalidRegisterParameter	0xEE10	Invalidate register parameter
InvalidRegisterResponse	0xEE11	Invalidate register response
UnknownRegisterResponse	0xEE12	Unknown register response
FailSendRegisterPacket	0xEE11	Failed to send register packet
NotSupported	0xEF00	Not Supported
Timeout	0xEFFF	Timeout
HandleMismatch	0xF001	Handle mismatch
CRCErrror	0xF002	CRC error on tag response
NoTagReply	0xF003	No tag reply
InvalidPassword	0xF004	Invalid password
ZeroKillPassword	0xF005	Zero kill password
TagLost	0xF006	Tag lost
CommandFormatError	0xF007	Command format error
ReadCountInvalid	0xF008	Read count invalid
OutOfRetries	0xF009	Out of retries

ParamError	0xFFFB	Parameter error
Busy	0xFFFC	Busy
InvalidCommand	0xFFFD	Invalid command
LowBattery	0xFFFE	Low battery
OperationFailed	0xFFFF	Operation failed