AsReader P51N

SDK リファレンス ガイド V1.2

Asterisk, Inc.

変更履歴

No.	バージョン	変更内容	日付
1	1.0	新規作成	2019/05/04
2	1.1	１、付録Ⅲを更新 ２、SetStaticIP メソッドを追加	2019/06/18
3	1.2	GetSdkVersion メソッドを追加	2019/06/24

目　次

前提条件

Windows バージョン：

開発ソフトウェア：VisualStdio バージョン 2012+

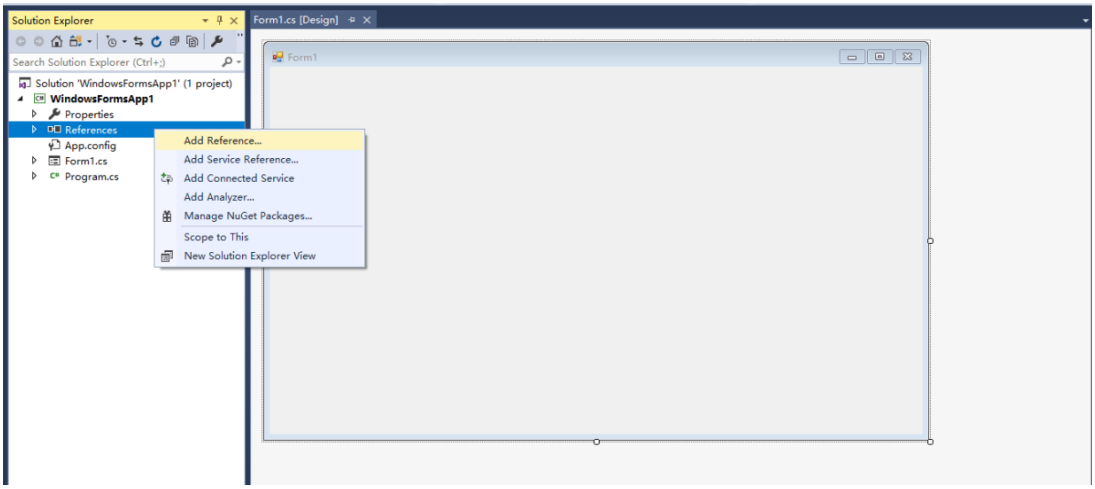
開発言語：C#

1 SDK の使用

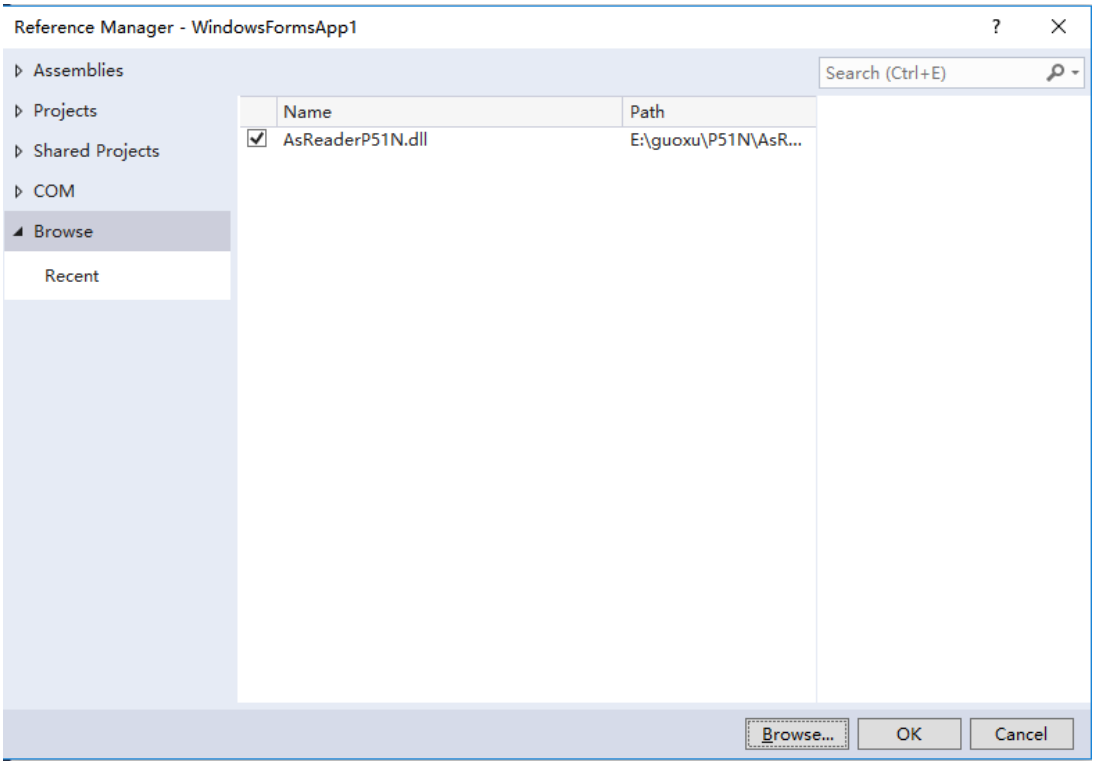
1.1 SDK 追加

1. プロジェクト（Windows フォームアプリケーション）を新規します。
   AsReaderP51N.dll と AsReader.dll をプロジェクトファイルにコピーします。

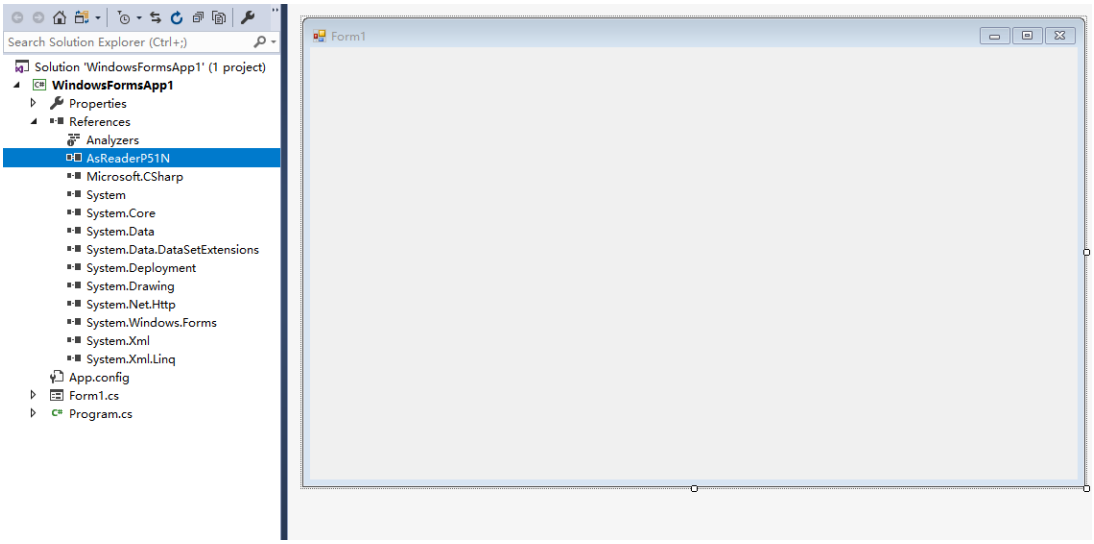
2. プロジェクトのインポートに AsReaderP51N.dll を追加します。
   インポート一覧で右クリックしてインポートを追加します。



3. 「ブラウズ」を選択し、プロジェクトパスに AaReaderP51N.dll を選択し、「確定」をクリックします。



4. 追加完了後、下図通り：

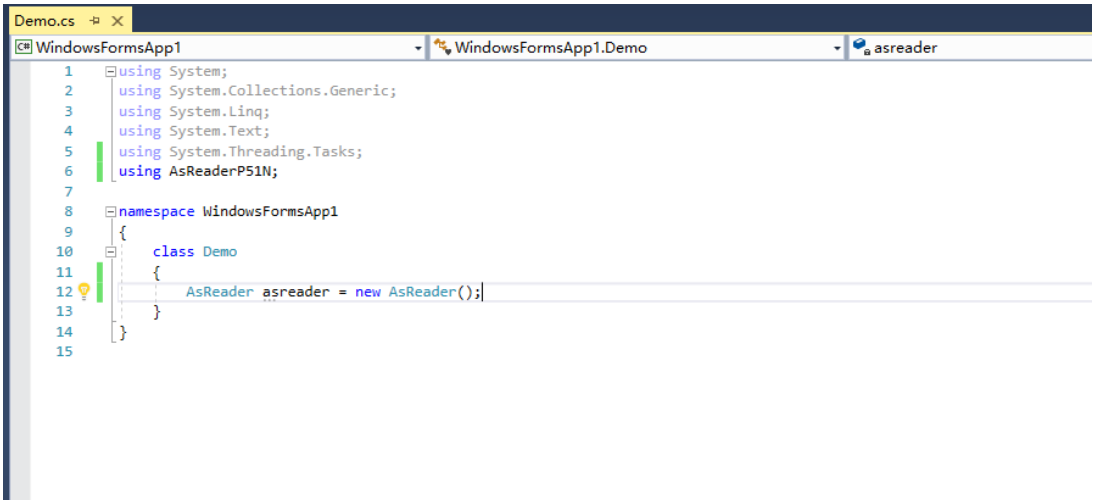


1.2 SDK インポート

1. 命名スペースをインポートします。

```
using AsReaderP51N;
```

2. オブジェクトをインスタンスします。



3. ConnectWithVCP メソッド（章 2.1.1 をご参照）を実行し、リーダライターを接続します。

注意：実行に成功した場合、0x0 を返す
実行に失敗した場合、0x2000002 を返す

```
asreader.ConnectWithVCP("COM1");
```

2 API 説明

2.1 AsReader Class

AsReader Class は RFID のインベントリー、読み取り、書き込み、ロックなどの API インターフェースを提供します。

2.1.1 ConnectWithVCP

関数名	UInt32 ConnectWithVCP(string comPort)		
引数名	IN/OUT	型	説明
comPort	IN	string	デバイスのポート番号
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： ポートを開き、ホストと IRI デバイスとの通信接続をさせ、デバイスの構造体を初期化し、ホストは IRI デバイスにすべての処理関数を登録します。			
例： ConnectWithVCP(“COM1”);			

2.1.2 ConnectWithTCP

関数名	UInt32 ConnectWithTCP(string ipAddress,string tcpPort)		
引数名	IN/OUT	型	説明
ipAddress	IN	string	デバイスの ip アドレス
tcpPort	IN	string	デバイス接続のポート
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： TCP/IP とポートを通じて、ホストとデバイスとの通信接続をさせ、デバイス構造体を初期化し、ホストはデバイスにすべての処理関数を登録します。			
例： ConnectWithTCP(“192.168.1.100”、“9600”); 注意：TCP 通信ポート番号：5000；UDP 通信ポート番号：50001			

2.1.3 Disconnect

関数名	UInt32 Disconnect()		
引数名	IN/OUT	型	説明
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： ホストとデバイスの接続を切断し、デバイスをリセットします。			
例：ポストとリーダライターの接続を切断します。 Disconnect();			

2.1.4 StartInventory

関数名	UInt32 StartInventory(bool epc,bool tid,bool pc,bool rssi,bool phase,bool channel,bool antenna)		
引数名	IN/OUT	型	説明
epc	IN	bool	true：表示される flase：表示されない
tid	IN	bool	true：表示される flase：表示されない
pc	IN	bool	true：表示される flase：表示されない
rssi	IN	bool	true：表示される flase：表示されない
phase	IN	bool	true：表示される flase：表示されない
channel	IN	bool	true：表示される flase：表示されない
antenna	IN	bool	true：表示される flase：表示されない
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： デバイスはタグのインベントリーを開始します。			
例： インベントリーされたタグデータに epc、tid を表示、pc、rssi、phase、channel、antenna を表示しません。 StartInventory(true,true,false,false,false,false,false);			

2.1.5 StopInventory

関数名	UInt32 StopInventory()		
引数名	IN/OUT	型	説明
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： デバイスがタグのインベントリーを停止します。			
例：インベントリーを停止します。 StopInventory();			

2.1.6 SetSelectMask

関数名	UInt32 SetSelectMask(MemBank memBank, Target target, Action action UInt32 startAddressWord, byte[] selectMask)		
引数名	IN/OUT	型	説明
memBank	IN	enum	マスク比較を行うタグのメモリバンク 章 2.2.1.7 をご参照
target	IN	enum	Selection Mask を適用するタグセッション 章 2.2.1.5 をご参照
action	IN	enum	タグがマークされた後のアクション 章 2.2.1.6 をご参照
startAddressWord	IN	UInt32	選択したタグのメモリバンクのオフセット（スタートアドレス） 単位：word
selectMask	IN	byte	マスクの値を選択[0-16] 単位：word
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： 選択されたマスクのパラメータを設定します。 複数のタグがある場合、この方法により特定のタグに対してインベントリー、読み取り、書き込み、ロックなどの操作をすることができます。			
例： タグをフィルターする条件： タグのメモリバンク： EPC セッション設定： SESSION_S0 アクション設定： ACTION_ASLINVA_DSLINVB オフセット：2 マスクを選択する内容は下記通り： byte[] selectMask= {0x12, 0x34, 0x56, 0x78, 0x12, 0x34, 0x56x, 0x78, 0x12, 0x34, 0x56, 0x78};			

SetSelectMask(Mem_EPC, SESSION_S0, ACTION_ASLINVA_DSLINVB, 0x02, selectMask)；

2.1.7 WriteMemory

関数名	UInt32 WriteMemory(MemBankType memBank, uint startAddressWord, uint accessPassword, byte[] writeData, byte[] epcData)		
引数名	IN/OUT	型	説明
memBank	IN	enum	章 2.2.1.7 をご参照
startAddressWord	IN	uint	ターゲットを書き込みバンクのオフセット 単位：Word
accessPassword	IN	uint	ターゲットタグのアクセスパスワード。(パスワードがを設定してない場合、0 である)
writeData	IN	byte	書き込みデータ
epcData	IN	byte	ターゲットタグの EPC 値
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： epcData でタグを選定し、タグのターゲットバンクにデータを書き込みます。 タグのメモリバンクに書き込むデータの長さは 32 ワード（Words）を超えないことです。すなわち 64 バイト（Bytes）です。			
例：タグを書き込みます。 メモリバンク：EPC オフセット：2 アクセスパスワード：0x12345678； 書き込むデータ：byte[] writedata = {0x12, 0x34}； ターゲットタグの EPC 値： byte[] epcData= {0x12, 0x34, 0x56, 0x78, 0x12, 0x34, 0x56x, 0x78, 0x12, 0x34, 0x56, 0x78}； WriteMemory(Mem_EPC, 0x02, 0x12345678, writeData, epcData)；			

2.1.8 ReadMemory

関数名	UInt32 ReadMemory(MemBankType memBank, uint startAddressWord, uint lengthWord, uint accessPassword, byte[] epcData)		
引数名	IN/OUT	型	説明
memBank	IN	MemBank	章 2.2.1.7 をご参照
startAddressWord	IN	UInt32	ターゲット読み取りバンクのオフセット 単位：Word
lengthWord	IN	UInt32	読み取りの長さ、単位：Word

accessPassword	IN	UInt32	ターゲットタグのアクセスパスワード。(パスワードがない場合、0 である)
epcData	IN	byte	ターゲットタグの EPC 値
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： epcData でタグを選定し、タグのターゲットバンクのデータを読み取ります。 タグのメモリバンクに読み取ったデータの長さは 32 ワード（Words）を超えないことです。すなわち 64 バイト（Bytes）です。			
例：タグを読み取ります。 ターゲットバンク：EPC オフセット：2 長さ：2 アクセスパスワード：0x12345678 ターゲットタグの EPC 値： byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78}; ReadMemory(Mem_EPC,0x02,0x02,0x12345678,epcData);			

2.1.9 Kill

関数名	UInt32 Kill(uint KillPassword,byte[] epcData)		
引数名	IN/OUT	型	説明
KillPassword	IN	uint	ターゲットタグのキルパスワード
epcData	IN	byte	ターゲットタグの EPC 値
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： epcData でターゲットタグを選定し、タグをキルします。 タグをキルする前にキルパスワードを RESERVED バンクに書き込む必要があります。 オフセット 00 から 2 Word を書き込みます。 注意：タグをキルすると、回復できなくなります。			
例：キルパスワード：0x12345678 ターゲットタグの EPC 値： byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78}; Kill(0x12345678,epcData);			

2.1.10 LockMemory

関数名	UInt32 LockMemory(TagMask tagMask,TagAction tagAction, uint accessPassword,byte[] epcData)		
引数名	IN/OUT	型	説明
tagMask	IN	TagMask	ロック操作のマスク（付録IVをご参照）
tagAction	IN	TagMask	ロック操作のデータ（付録IVをご参照）
accessPassword	IN	uint	ターゲットタグのアクセスパスワード（パスワードを設定してない場合、0 である。）
epcData	IN	byte	ターゲットタグの EPC 値
戻り値	OUT	UInt32	付録Ⅰをご参照

メソッド説明：

タグのメモリバンクに対してロック（Lock）、永久ロック（PermaLock）、アンロック（Unlock）永久アンロック（PermaUnlock）をします。

タグをロックする前に、アクセスパスワードを RESERVED バンクに書き込む必要があります。

オフセット 0X20 から 2 Word を書き込みます。

ロック操作パラメータ target の高 4 桁は保留桁であり、残りの 20 桁はロック操作の Payload であり、Mask と Action を含めて、高いから低い順に 10 桁ずつあります。

マスクは Mask 桁が 1 の Action だけに有効です。各データエリアの Action は 2 bits、00~11 があり、オープン、永久オープン、ロック、永久ロックの順番です。

Mask と Action の各桁の意味は以下通り：

Lock-Command Payload

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Kill Mask		Access Mask		EPC Mask		TID Mask		File_0 Mask		Kill Action		Access Action		EPC Action		TID Action		File_0 Action	

Masks and Associated Action Fields

	Kill pwd		Access pwd		EPC memory		TID memory		File_0 memory	
	19	18	17	16	15	14	13	12	11	10
Mask	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write
	9	8	7	6	5	4	3	2	1	0
Action	pwd read/ write	perma lock	pwd read/ write	perma lock	pwd write	perma lock	pwd write	perma lock	pwd write	perma lock

例：タグの EPC バンクをロックします。

ターゲットタグのアクセスパスワード：12345678。

ターゲットタグの EPC 値： byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78}; LockMemory(0x008020,0x12345678,epcData);

2.1.11 GetFirmwareVersion

関数名	UInt32 GetFirmwareVersion(ref string firmwareVersion)		
引数名	IN/OUT	型	説明
firmwareVersion	OUT	string	デバイスのファームウェアバージョン
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： デバイスのファームウェアバージョンを取得します。			
例：デバイスのファームウェアバージョンを取得します。 GetFirmVersion(firmwareVersion);			

2.1.12 SetRegion

関数名	UInt32 SetRegion(RegionType region)		
引数名	IN/OUT	型	説明
region	IN	enum	当面のリージョン（章 2.2.1.1 をご参照）
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： リージョンを設定します。			
例：リージョンを REGION_FCC_PART_15_247 に設定します。 SetRegion(REGION_FCC_PART_15_247);			

2.1.13 GetRegion

関数名	UInt32 GetRegion(ref RegionType region)		
引数名	IN/OUT	型	説明
region	OUT	enum	当面のリージョン（章 2.2.1.1 をご参照）
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： リージョンを取得します。			
例：リージョンを取得します。 GetRegion(Value);			

2.1.14 SetValue

関数名	UInt32 SetValue(UInt key, uint vaule)		
引数名	IN/OUT	型	説明
key	IN	unit	デバイスの関連パラメータ （章 2.2.1.8 をご参照）
vaule	IN	unit	当該パタメータの値
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： デバイスの関連パラメータを設定します。			
例：発射パワーを 30dbm に設定します。 SetValue(E_IPJ_KEY_ANTENNA_TX_POWER, 30)；			

2.1.15 GetValue

関数名	UInt32 GetValue(uint key, ref uint vaule)		
引数名	IN/OUT	型	説明
key	IN	unit	デバイスの関連パラメータ （章 2.2.1.8 をご参照）
vaule	OUT	unit	当該パタメータ値
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： デバイスの関連パラメータを取得します。			
例：発射パワーを取得します。 GetValue(E_IPJ_KEY_ANTENNA_TX_POWER, value)；			

2.1.16 GetAntennaPorts

関数名	GetAntennaPorts(ref bool ant1, ref bool ant2, ref bool ant3, ref bool ant4)		
引数名	IN/OUT	型	説明
ant1	OUT	bool	アンテナ１の状態（On または Close） [true,flase]
ant2	OUT	bool	アンテナ２の状態（On または Close） [true,flase]
ant3	OUT	bool	アンテナ３の状態（On または Close） [true,flase]
ant4	OUT	bool	アンテナ４の状態（On または Close） [true,flase]
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： 全てのアンテナポートの状態を取得します。（On または Close）			

例：全てのアンテナポートの状態を取得します。 GetAntennaPorts(ant1,ant2,ant3,ant4)；

2.1.17 SetAntennaPorts

関数名	SetAntennaPorts(bool ant1,bool ant2,bool ant3, bool ant4)		
引数名	IN/OUT	型	説明
ant1	IN	bool	アンテナ１の状態（On または Close） [true,flase]
ant2	IN	bool	アンテナ２の状態（On または Close） [true,flase]
ant3	IN	bool	アンテナ３の状態（On または Close） [true,flase]
ant4	IN	bool	アンテナ４の状態（On または Close） [true,flase]
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： 全てのアンテナポートの状態を設定します。（On または Close）			
例：アンテナ１をオンにし、アンテナ 2、3、4 をクローズにします。 SetAntennaPorts(true,flase,flase,flase)；			

2.1.18 GetTxPower

関数名	UInt32 GetTxPower(ref uint power)		
引数名	IN/OUT	型	説明
power	OUT	uint	発射パワーのパラメータ値 [0-30]
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： リーダライターの発射パワーを取得します。			
例：リーダライターの発射パワーを取得します。 GetTxPower(power)；			

2.1.19 SetTxPower

関数名	UInt32 SetTxPower(uint txPower)		
引数名	IN/OUT	型	説明
txPower	IN	uint	発射パワーのパラメータ値 [0-30]
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： リーダライターの発射パワーを設定します。			

例：発射パワーを 30dBm に設定します。 SetTxPower(30)；

2.1.20 SetRFMode

関数名	UInt32 SetRFMode(RFModeType mode)		
引数名	IN/OUT	型	説明
mode	IN	enum	リーダライターの RFID モード。（章 2.2.1.2 をご参照）
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： リーダライターの RFID モードを設定します。			
例：RFID モードを RFMODE0（自動モード）に設定します。 SetRFMode(RFMODE0)；			

2.1.21 GetRFMode

関数名	UInt32 GetRFMode(ref RFModeType mode)		
引数名	IN/OUT	型	説明
mode	OUT	enum	リーダライターの RFID モード。（章 2.2.1.2 をご参照）
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： リーダライターの RFID モードを取得します。			
例：RFID モードを取得します。 GetRFMode(mode)；			

2.1.22 SetSession

関数名	UInt32 SetSession(SessionType session)		
引数名	IN/OUT	型	説明
session	IN	enum	RFID Session（章 2.2.1.3 をご参照）
戻り値	OUT	UInt32	付録 I をご参照
メソッド説明： RFID Session を設定します。			
例： RFID Session を Session_S0 に設定します。 SetSession(Session_S0)；			

2.1.23 GetSession

関数名	UInt32 GetSession(ref SessionType session)		
引数名	IN/OUT	型	説明
session	OUT	enum	RFID Session（章 2.2.1.3 をご参照）
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： RFID Session を取得します。			
例：RFID Session を取得します。 GetSession(session)；			

2.1.24 SetSearchMode

関数名	UInt32 SetSearchMode(SearchModeType mode)		
引数名	IN/OUT	型	説明
mode	IN	enum	RFID のサーチモード（章 2.2.1.4 をご参照）
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： リーダライターの RFID サーチモードを設定します。			
例：RFID のサーチモードを INVENTORY_SEARCH_MODE_AUTO_SEARCH に設定します。 SetSearchMode(INVENTORY_SEARCH_MODE_AUTO_SEARCH)；			

2.1.25 GetSearchMode

関数名	UInt32 GetSearchMode(ref SearchModeType mode)		
引数名	IN/OUT	型	説明
mode	OUT	enum	RFID のサーチモード（章 2.2.1.4 をご参照）
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： RFID のサーチモードを取得します。			
例：RFID のサーチモードを取得します。 GetSession(mode)；			

2.1.26 SetCWOn

関数名	UInt32 SetCWOn()		
引数名	IN/OUT	型	説明
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： CW 搬送波をオンにします。 CW 搬送波：順方向リンク変調がない場合、コンフィグされたチャンネルまたは周波数でレーダーの信号を発送します。			
例：CW 搬送波をオンにします。 SetCWOn()；			

2.1.27 SetCWOff

関数名	UInt32 SetCWOff()		
引数名	IN/OUT	型	説明
戻り値	OUT	UInt32	付録Ⅰをご参照
メソッド説明： CW 搬送波をオフにします。 CW 搬送波：順方向リンク変調がない場合、コンフィグされたチャンネルまたは周波数でレーダーの信号を発送します。			
例：CW 搬送波をオフにします。 SetCWOff()；			

2.1.28 GetDeviceInformation

関数名	void GetDeviceInformation(DeviceInformation deviceInformation)		
引数名	IN/OUT	型	説明
deviceInformation	OUT	DeviceInformation	デバイス情報を取得 付録Ⅲをご参照
戻り値	OUT	Void	
メソッド説明： デバイス情報を取得します。			
例：デバイス情報を取得します。 GetDeviceInformation(mode)；			

2.1.29 SetStaticIP

関数名	bool SetStaticIP(string ip,string subnet,string gateway,string dnsServer)		
引数名	IN/OUT	型	説明
ip	IN	string	IP アドレス
subnet	IN	string	サブネットマスク
gateway	IN	string	ゲートウェイ
dnsServer	IN	string	DNS サーバー
戻り値	OUT	bool	
メソッド説明： 静的 IP アドレスを設定します。			
例： SetStaticIP(“192.168.3.1”,” 255.255.255.0”,” 192.168.3.1”,” 192.168.3.1”)；			

2.1.30 GetSdkVersion

関数名	void GetSdkVersion(ref string sdkVersion)		
引数名	IN/OUT	型	説明
sdkVersion	OUT	string	SDK バージョンを取得
戻り値	OUT	Void	
メソッド説明： SDK バージョンを取得します。			
例： string sdkVersion; GetSdkVersion(sdkVersion)；			

2.1.31 SetDelegate

関数名	Void SetDelegate(CallBackReadTagData readTagData,CallBackError errorString)		
引数名	IN/OUT	型	説明
readTagData	IN	CallBackReadTagData	データ処理のコールバック関数
errorString	IN	CallBackError	実行にエラーが出る場合のコールバック関数
戻り値	OUT	Void	
メソッド説明： デリゲート関数を設定します。			

例： CallBackReadTagData Rec = null; CallBackError Rec1 = null; Void test(InventoryResult ReadTagStruct); Void test1(string Error); Rec = test; Rec1 = test1; SetDelegate(Rec,Rec1);

注意：データを処理するデリゲートと実行にエラーが出る場合の出力用デリゲートを定義します。
public delegate void CallBackReadTagData(InventoryResult tagcallbackdata);
public delegate void CallBackErrorString(string error);
InventoryResult　付録Ⅱをご参照
error　付録Ⅰをご参照

2.2 Types Class

Types クラスはリージョン、RFID モード、セッションモード、サーチモード、Selection Mask のタグセッション、タグのセッション状態、メモリバンク、パラメータ設定を定義します。

2.2.1 列挙型

2.2.1.1 RegionType

リージョン	パラメータ
REGION_FCC_PART_15_247	0x0
REGION_HONG_KONG_920_925_MHZ	0x3
REGION_TAIWAN_922_928_MHZ	0x4
REGION_ETSI_EN_302_208_V1_4_1	0x7
REGION_KOREA_917_921_MHZ	0x8
REGION_MALAYSIA_919_923_MHZ	0x9
REGION_CHINA_920_925_MHZ	0xA
REGION_SOUTH_AFRICA_915_919_MHZ	0xC
REGION_BRAZIL_902_907_AND_915_928_MHZ	0xD
REGION_THAILAND_920_925_MHZ	0xE

REGION_SINGAPORE_920_925_MHZ	0xF
REGION_AUSTRALIA_920_926_MHZ	0x10
REGION_INDIA_865_867_MHZ	0x11
REGION_URUGUAY_916_928_MHZ	0x12
REGION_VIETNAM_920_925_MHZ	0x13
REGION_ISRAEL_915_917_MHZ	0x14
REGION_PHILIPPINES_918_920_MHZ	0x15
REGION_INDONESIA_923_925_MHZ	0x17
REGION_NEW_ZEALAND_921P5_928_MHZ	0x18
REGION_JAPAN_916_921_MHZ_NO_LBT	0x19
REGION_PERU_916_928_MHZ	0x1A
REGION_RUSSIA_916_921_MHZ	0x1B
REGION_CUSTOM	0x100

2.2.1.2 RfModeType

RFID モード

RFID モード	パラメータ	説明
RFMODE0	0x0	自動モード（FCC モード）
RFMODE1	0x1	FCC モード（902～928MHz）
RFMODE2	0x2	ETSI モード(865～868MHz)
RFMODE3	0x3	高速度インベントリーモード
RFMODE4	0x4	ノイズ低減敏感モード

2.2.1.3 SessionType

セッションモード

セッションモード	パラメータ
Session_S0	0x0
Session_S1	0x1
Session_S2	0x2
Session_S3	0x3

2.2.1.4 SearchModeType

サーチモード

サーチモード	パラメータ

INVENTORY_SEARCH_MODE_AUTO_SEARCH	0x0
INVENTORY_SEARCH_MODE_DUAL_TARGET	0x1
INVENTORY_SEARCH_MODE_SINGLE_TARGET_A_TO_B	0x2
INVENTORY_SEARCH_MODE_SINGLE_TARGET_B_TO_A	0x3

2.2.1.5 TargetType

Selection Mask を適用するタグのセッションを設定します。

Target	パラメータ	説明
SESSION_S0	0x0	inventoried S0
SESSION_S1	0x1	inventoried S1
SESSION_S2	0x2	inventoried S2
SESSION_S3	0x3	inventoried S3
SL_FLAG	0x4	Selection Flags

2.2.1.6 ActionType

タグのセッション状態を設定します。
Selection Mask 使う時に、Selection Mask にマッチング、マッチングしない場合、タグの Session と Session Flag の変化をそれぞれに表示します。

Action	パラメータ	説明
ACTION_ASLINVA_DSLINVB	0x0	Tag Matching: assert SL or inventoried → A Tag Not-Matching：retract SL or inventoried → B
ACTION_ASLINVA_NOTHING	0x1	Tag Matching ： assert SL or inventoried → A Tag Not-Matching ： do nothing
ACTION_NOTHING_DSLINVB	0x2	Tag Matching ： do nothing Tag Not-Matching：retract SL or inventoried → B
ACTION_NSLINVS_NOTHING	0x3	Tag Matching ： negate SL or (A → B, B → A) Tag Not-Matching ： do nothing
ACTION_DSLINVB_ASLINVA	0x4	Tag Matching ： retract SL or inventoried → B Tag Not-Matching：assert SL or inventoried → A
ACTION_DSLINVB_NOTHING	0x5	Tag Matching ： retract SL or inventoried → B Tag Not-Matching ： do nothing
ACTION_NOTHING_ASLINVA	0x6	Tag Matching ： do nothing Tag Not-Matching：assert SL or inventoried → A
ACTION_NOTHING_NSLINVS	0x7	Tag Matching ： do nothing

		Tag Not-Matching : negate SL or (A → B, B → A)

2.2.1.7 MemBankType

Selection Mask 比較を行うタグのメモリバンクを設定します。

MemBank	パラメータ	説明
MEM_RESERVED	0x0	保留バンド
MEM_EPC	0x1	EPC バンド
MEM_TID	0x2	TID バンド
MEM_USER	0x3	ユーザーバンド

2.2.1.8 KeyValueType

Key_Value	パラメータ	説明
KEY_BOOTSTRAP_VERSION	0x1	Indy Module にロードされたブートストラップイメージのバージョン
KEY_BOOTSTRAP_CRC	0x2	Indy Module のブートストラップイメージの CRC（サイクル冗長検査）値。
KEY_APPLICATION_VERSION	0x3	Indy Module にロードされてたアプリケーションイメージのバージョン
KEY_APPLICATION_CRC	0x4	Indy モジュールのアプリケーションイメージの CRC（サイクル冗長検査）値
KEY_SECONDARY_IMAGE_VERSION	0x5	セカンダリー（アプリケーション以外）のイメージバージョン
KEY_SECONDARY_IMAGE_CRC	0x6	セカンダリー（アプリケーション以外）のサイクル冗長値
KEY_SECONDARY_IMAGE_TYPE	0x7	セカンダリー（アプリケーション以外）イメージタイプ
KEY_APPLICATION_REVISION_ID	0x8	セカンダリー（アプリケーション以外）のイメージリビジョン ID
KEY_APPLICATION_BUILD_ID	0x9	アプリケーションイメージのビルド ID
KEY_PRODUCT_ID	0xA	Indy モジュール製品の ID
KEY_SERIAL_NUMBER	0xB	Indy モジュールのシリアルコード
KEY_TRANSCEIVER_ID	0xC	モジュール内に Indy リーダライター

		チップの ID
KEY_MICROPROCESSOR_ID	0xD	Indy モジュールにシングルマイクロコントローラ唯一の ID
KEY_CUSTOMER_VERSION	0xE	修正したモジュールのハードウェアのバージョン識別子
KEY_CUSTOMER_ID	0xF	修正したモジュールのハードウェアの識別子
KEY_CUSTOMER_PRODUCT_ID	0x10	修正したモジュールのハードウェアの識別子
KEY_CALIBRATION_INFO	0x11	キャリブレーションソース情報
KEY_TEST_INFO	0x12	実行されたテストコマンドの回数
KEY_PRODUCT_SKU	0x13	モジュールのハードウェア SKU
KEY_LOT_DATE_CODE	0x14	モジュールのロットと日付コード
KEY_PRODUCT_KEY	0x15	各パラメータに基づいて作成された唯一のモジュール識別子
KEY_SECONDARY_IMAGE_LOCATION	0x16	マイナーイメージの位置
KEY_SECONDARY_IMAGE_SIZE	0x17	マイナーイメージのサイズ
KEY_UNIQUE_ID	0x18	デバイスの 64 桁唯一の ID。
KEY_HARDWARE_REVISION	0x19	モジュールのハードウェアバージョン
KEY_SYSTEM_TIMESTAMP_MS	0x1A	当面のシステムタイムスタンプ
KEY_ACTIVE_ACTIONS	0x1B	デバイスで自発的に実行するコマンド
KEY_REGION_ID	0x20	モジュールが使用する監督リジョーンのパラメータを設定します。
KEY_REGION_CHANNEL_TABLE	0x21	コンフィグされたカスタムモニターチャネルテーブル
KEY_REGION_CHANNEL_TABLE_SIZE	0x22	カスタムリジョーンチャネルホップテーブルのサイズ
KEY_REGION_ON_TIME_NOMINAL	0x23	カスタム監督リージョンの滞留タイム（ms）
KEY_REGION_ON_TIME_ACCESS	0x24	アクセスオペレーションを実行した時、カスタム監督リージョンの滞留タイム（ms）
KEY_REGION_OFF_TIME	0x25	カスタム監督リージョン内に違うチャンネルにホッピングした時のオフタイム（ms）
KEY_REGION_OFF_TIME_SAME_CHANNEL	0x26	カスタム監督リージョン内の同じチャンネルに切り替えた時のオフタイム（ms）
KEY_REGION_START_FREQUENCY_KHZ	0x27	カスタム監督リージョン 1 の周波数

		（kHz）
KEY_REGION_CHANNEL_SPACING_KHZ	0x28	カスタム監督リージョンチャンネルのスペーシング（kHz）。
KEY_REGION_RANDOM_HOP	0x29	カスタム監督リージョンのランダムホップを起動
KEY_REGION_INDY_PLL_R_DIVIDER	0x2A	Indy PLL パラメータ
KEY_REGION_RF_FILTER	0x2B	ユーザーにコンフィグしたカスタム監督リージョンに RF SAW フィルターを選択
KEY_ANTENNA_TX_POWER	0x31	アンテナ送信電力（cdBm）（例： 2300 cdBm = 23 dBm）。
KEY_ANTENNA_SEQUENCE	0x32	リーダが循環するアンテナシーケンス
KEY_INVENTORY_TAG_POPULATION	0x40	RF 電界に基づいて、rfid タグの種類を見積もりします。
KEY_INVENTORY_SELECT_FLAG	0x41	在庫選択のフラッグ
KEY_INVENTORY_SESSION	0x42	在庫セッション番号（0 - 3）
KEY_INVENTORY_SEARCH_MODE	0x43	在庫サーチモード
KEY_FAST_ID_ENABLE	0x45	Monza タグのために、FastID 機能を起動する
KEY_TAG_FOCUS_ENABLE	0x46	Monza タグのために、TagFocus 機能を起動します。セッションは S1 必須、サーチモードは A-> B 必須。
KEY_TAG_OPERATION_ENABLE	0x47	インベントリー中にタグオペレーションを行います。
KEY_TAG_OPERATION_RETRIES	0x48	失敗したタグのアクセスオペレーションでの最大再試行回数。
KEY_SELECT_ENABLE	0x50	インベントリーワンラウンドの前に Select オペレーションの伝送を行います。
KEY_SELECT_TARGET	0x51	Select コマンドがタグの選定フラグまたは在庫フラグを修正するかどうかを判断します。
KEY_SELECT_ACTION	0x52	Select コマンドを実行中のフラグの修正内容を指定します。
KEY_SELECT_MEM_BANK	0x53	Select コマンドを適用したタグのメモリバンド
KEY_SELECT_POINTER	0x54	メモリバンド内のオフセットを指定します。Select コマンドのタグマスクが始まります。
KEY_SELECT_MASK_LENGTH	0x55	このキーコードが SELECT_POINTER キーと合わせて Select コマンドマスク

		のメモリ範囲を決定します。（SELECT_MASK_VALUE キーでコンフィグしました）
KEY_SELECT_MASK_VALUE	0x56	Select コマンドのタグフィルタがマッチングする必要なビットパターンを定義します。
KEY_TAG_OPERATION	0x60	TAG_OPERATION_ENABLE キーを使用する場合、アクセスコマンドを起動する時タグに発送する特定のアクセスコマンドをコンビィグします。（タグの操作）
KEY_ACCESS_PASSWORD	0x61	タグをセーフ状態にする場合、アクセスコマンドと合わせて使用される 32 ビットのアクセスパスワードを指定します。
KEY_KILL_PASSWORD	0x62	タグキルを行う時の 32 桁キルパスワードを指定します。
KEY_READ_MEM_BANK	0x63	Access 操作で Read コマンドにタグメモリバンドを指定します。
KEY_READ_WORD_POINTER	0x64	Access 操作で Read コマンドのワードポインタです。
KEY_READ_WORD_COUNT	0x65	Access 操作で読み取ったデータの長さ（単位：Word）
KEY_WRITE_MEM_BANK	0x66	Access 操作で書き込みコマンドのタグメモリバンドをコンフィグします。
KEY_WRITE_WORD_POINTER	0x67	Access 操作で書き込みコマンドにアクセスのワードポインタ
KEY_WRITE_WORD_COUNT	0x68	Access 操作で書き込みコマンドに書き込んだデータの長さ（単位：word）
KEY_WRITE_DATA	0x69	指定された 16 ビットワードで書き込みコマンドに書き込むタグメモリデータです。
KEY_LOCK_PAYLOAD	0x6A	ロックコマンドのペイロードフィールドです。
KEY_BLOCKPERMALOCK_ACTION	0x6B	BlockPermalock のアクション
KEY_BLOCKPERMALOCK_MEM_BANK	0x6C	BlockPermalock 操作でタグのターゲットメモリバンド
KEY_BLOCKPERMALOCK_BLOCK_POINTER	0x6D	BlockPermalock 操作のブロックポインター
KEY_BLOCKPERMALOCK_BLOCK_RANGE	0x6E	BlockPermalock 操作で単位は 16 blocks の BlockPermalock のマスク範囲です。

KEY_BLOCKPERMALOCK_MASK	0x6F	BlockPermalock マスク
KEY_WRITE_EPC_LENGTH_CONTROL	0x70	TAG_OPERATION キーを選択する場合、WRITE_EPC タグの操作で、WRITE_EPC の長さを制御できます。
KEY_WRITE_EPC_LENGTH_VALUE	0x71	WRITE_EPC_LENGTH_CONTROL キーを選択する場合、ユーザーが書き込みタグの EPC の長さを指定できます。
KEY_WRITE_EPC_AFI_CONTROL	0x72	TRUE に設定した場合、Write EPC 操作中にタグ EPC メモリ内の AFI ビットの書き込みが有効されます。
KEY_WRITE_EPC_AFI_VALUE	0x73	EPC 書き込み操作で AFI 制御を起動する場合、ユーザーAFI ビットを指定できます。
KEY_QT_ACTION	0x74	QT オペレーション。QT の書き込みと読み取りを選択できます。
KEY_QT_PERSISTENCE	0x75	QT 操作の持続性
KEY_QT_DATA_PROFILE	0x76	QT データをコンフィグします。
KEY_QT_ACCESS_RANGE	0x77	QT アクセス範囲
KEY_QT_TAG_OPERATION	0x78	QT コマンドの後に、タグ操作が有効になった時、特定のアクセスコマンドを指定します。
KEY_AUTOSTOP_DURATION_MS	0x89	持続インベントリーラウンドの実行タイムを指定します。指定すると、インベントリーが停止されます。
KEY_AUTOSTOP_TAG_COUNT	0x8B	複数の持続インベントリーラウンド中で棚卸しする最大のタグ数を指定します。
KEY_AUTOSTOP_ROUND_COUNT	0x8C	ストップする前に実行される最大のインベントリーラウンドを指定します。
KEY_REPORT_CONTROL_TAG	0xA1	タグオペレーションのレポートに表示されるタグフィールドを制御します。
KEY_REPORT_CONTROL_STATUS	0xA2	どんなステータスのレポートを生成されることを制御します。
KEY_REPORT_CONTROL_TIMESTAMP	0xA3	モジュールからレポートにタイムスタンプを含めるかどうかを制御します。
KEY_RESPONSE_CONTROL_TIMESTAMP	0xA	モジュールからのレスポンスがタイムスタンプを含めるかどうかを制御します。
KEY_TEMPERATURE_INTERNAL	0xB0	モジュール中のマイクロコントロー

		ラのセンサーによって測定された温度値。（摂氏温度）
KEY_TEMPERATURE_EXTERNAL	0xB1	推定された当面のモジュールの外部温度（摂氏温度）
KEY_TEMPERATURE_PA	0xB2	パワーアンプ（PA）温度センサーによって測定された当面の温度（摂氏温度）
KEY_GPIO_MODE	0xC0	モジュールの GPIO ピンモード（インプット、インプットアクション、アウトプットなど）を選択します。
KEY_GPIO_STATE	0xC1	GPIO ロジックレベルの出力(+3.3 V/0.0 V)を制御します。
KEY_GPIO_HI_ACTION	0xC2	GPIO が High ロジックステータスに転換した時にモジュールで実行されるアクションを制御します。
KEY_GPIO_LO_ACTION	0xC3	GPIO が LOW ロジックステータスに転換した時にモジュールで実行されるアクションを制御します。
KEY_GPIO_PULSE_DURATION	0xC4	GPIO アクションの脈動持続時間を制御します。
KEY_GPIO_DEBOUNCE_MS	0xC5	GPIO アクションの内部デバウンスタイムアウトを制御します。
KEY_GPIO_CURRENT_STATE	0xC6	当面の GPIO ピンのロジックレベル（+3.3 V/0.0 V）を表示します。。
KEY_RF_MODE	0xD0	インベントリーオペレーションの RF モード
KEY_FIRST_ERROR	0xE0	value_index 0 で最後に該当キーをクリアしてから発生した最初のエラー及び value_index 1～4 中に他のパラメータエラーを含めています。
KEY_LAST_ERROR	0xE1	最後に発生したエラーを含めています。
KEY_SYSTEM_ERROR	0xE2	最後に発生したシステムレベルのエラー(ハードフォールト)を含みます。
KEY_CAL_DATA_1	0xF0	DATA1。
KEY_CAL_DATA_2	0xF1	DATA2。
KEY_CAL_DATA_3	0xF2	DATA3。
KEY_CAL_DATA_4	0xF3	DATA4。
KEY_CAL_DATA_5	0xF4	DATA5。
KEY_CAL_DATA_6	0xF5	DATA6。
KEY_CAL_DATA_7	0xF6	DATA7。

KEY_CAL_DATA_8	0xF7	DATA8。
KEY_DEVICE_BAUDRATE	0x100	モジュール UART1（IRI）インタフェースが実行した後のシリアルボーレート
KEY_DEVICE_IDLE_POWER_MODE	0x101	デバイスがアイドル状態のパワー消耗モード
KEY_ONBOOT_START_ACTION	0x102	モジュールが起動直後に実行されるアクション
KEY_ENABLE_LT_REPORTS	0x103	UART1（IRI）インターフェイスで IRI-LT 形式のレポートを自動的に送信できるようにモジュールをコンフィグします。
KEY_IN_BOOTSTRAP	0x104	リーダーがブートストラップにある場合、true を返します。ブートストラップバージョンが 1.8.x.x 以上の場合のみ動作できます。
KEY_EXTERNAL_ANTENNA_MUX_ENABLE	0x200	外部アンテナマルチプレックス可能
KEY_EXTERNAL_ANTENNA_MUX_NUM_ANTENNAS	0x201	外部アンテナマルチプレックスをサポートする数
KEY_EXTERNAL_ANTENNA_MUX_PHYSICAL_PORT	0x202	外部アンテナマルチプレックスの共通ポートが接続するフィジカルポートです。
KEY_EXTERNAL_ANTENNA_MUX_DELAY_MICROSECONDS	0x203	外部アンテナマルチプレックスの RF 間のアクティビィティディレイはマイクロ秒です。
KEY_ANTENNA_SEQUENCE_OPTION	0x220	アンテナ記述子を使用して順序付けされた設定が異なるフィジカルアンテナポートに接続されているか、または同じフィジカルアンテナに対する複数ロジックの記述子であるかを設定します。
KEY_ANTENNA_DESCRIPTORS	0x221	ロジックアンテナパラメータの key-value を制御するバンクキーです。
KEY_ANTENNA_PHYSICAL_PORT	0x222	フィジカルアンテナポートが接続されているアンテナ記述子シーケンス内の特定のアンテナポートです。
KEY_TEST_ID	0x400	テスト ID であり、特定のテストコマンドを選択する時に使います。
KEY_TEST_PARAMETERS	0x401	テストコマンドパラメータであり、特定のテストコマンドをインプットするときに使います。

KEY_TEST_RESULT_1	0x402	テストコマンドの結果は 1
KEY_TEST_RESULT_2	0x403	テストコマンドの結果は 2
KEY_TEST_RESULT_3	0x404	テストコマンドの結果は 3
KEY_TEST_DATA	0x405	テストコマンドデータ
KEY_TEST_FREQUENCY	0x406	最後にロックされた頻度
KEY_TEST_POWER	0x407	最新の送信電力
KEY_TEST_RF_MODE	0x408	RF プロファイル ID
KEY_TEST_TIME	0x409	時間通りに最後の送信
KEY_TEST_EVENT	0x40A	イベント情報
KEY_TEST_REPORTS	0x40B	レポート情報
KEY_TEST_SYSTEM	0x40C	システム情報
KEY_TEST_DEBUG_PORT	0x40D	シリアルデバッグポートのコンフィグ
KEY_GENERIC_DATA	0xC00	カスタムデータをデバイスに追加するための第三者向けの汎用データストレージ
KEY_OEM_DATA	0xC01	キャリブレーション中にカスタムデータをデバイスに追加するための OEM データストレージ
KEY_BLOCK_WRITE_OVERRIDE	0x1018	ブロック書き込みモードの制御
KEY_SJC_CONTROL	0x101B	
KEY_STORED_SETTINGS_LOAD_STATUS	0x1021	ストレージ設定のステータスをローディングします。
KEY_PA_DIE_TEMPERATURE_TX_DUTY_CYCLE_LIMIT	0x1022	RS1000 限定。PA ダイの温度がこの限界を超えると、PA ダイの最大熱限界 150℃に当たらないように送信機のオンタイムデューティサイクルが比例して減少します。
KEY_TEMP_COMP_OFFSETS	0x1023	各 TempComp パワーカーブに定数のオフセットを追加します。

付録 I:

返したパラメータ	パラメータ	説明
E_IPJ_ERROR_SUCCESS	0x0	前回の操作が成功に実行、または起動後エラーが出なかった。
E_IPJ_ERROR_GENERAL_ERROR	0x1	想定外のエラーが発生する時に出られます。テストコマンドを使う時に発生したエラーも含めます。
E_IPJ_ERROR_SET_KEY_INVALID	0x2	無効のキーID を使用する時、または無効のパラメータでキーを操作する時に出られます。
E_IPJ_ERROR_SET_KEY_READ_ONLY	0x3	読み取り専用のキーを設定する時に出られます。
E_IPJ_ERROR_SET_KEY_OUT_OF_RANGE	0x4	きーを有効範囲外に設定する場合、出られます。
E_IPJ_ERROR_GET_KEY_INVALID	0x5	無効のキーID を使用する時、または無効のパラメータでキーの取得(ロット取得も含み)を実行する時に出られます。
E_IPJ_ERROR_GET_KEY_WRITE_ONLY	0x6	書き込み専用のキーを取得する場合に出られます。
E_IPJ_ERROR_COMMAND_INVALID	0x7	start () API 関数が無効のパラメータを呼び出す時に出られます。
E_IPJ_ERROR_COMMAND_START_FAILURE	0x8	start () API 関数がリスト以外の原因で呼び出失敗した場合に出られます。
E_IPJ_ERROR_COMMAND_DECODE_FAILURE	0x9	IRI ホストライブラリがモジュールパケットのデコードに失敗した場合に出られます。
E_IPJ_ERROR_COMMAND_ENCODE_FAILURE	0xA	IRI ホストライブラリがモジュールに発送するパケットをエンコードできない場合に出られます。
E_IPJ_ERROR_COMMAND_STALLED	0xB	IRI ホストがモジュール内でコマンドが指定された時間に完成できなかった場合に出られます。
E_IPJ_ERROR_VALUE_INVALID	0xC	モジュール内のロジックはコンフィグされた内容の一部が無効だと判断した場合に出られます。

E_IPJ_ERROR_MORE_THAN_ONE_COMMAND_RECEIVED	0xD	モジュールが複数の衝突コマンドを受け取った場合に出られます。
E_IPJ_ERROR_NOT_IMPLEMENTED	0xE	IRI ホストがモジュールのアプリケーションにサポートしてない機能を起動する場合に出られます。
E_IPJ_ERROR_INVALID_PRODUCT_CONFIGURATION	0xF	モジュールに無効な設定が含まれて start（）API 関数を呼び出す場合に出られます。
E_IPJ_ERROR_INVALID_FACTORY_SETTINGS	0x10	モジュールに無効な出荷設定が含まれて start（）API 関数を呼び出す場合に出られます。
E_IPJ_ERROR_RESPONSE_ENCODE_FAILURE	0x11	レポートまたはレスポンスをエンコードしようとしている時にモジュールがエラーを検出した場合に出られます。
E_IPJ_ERROR_COMMAND_VERIFY_FAILURE	0x12	モジュールが正しく書き込まれなかったデータを検出した場合に出られます。
E_IPJ_ERROR_INTERNAL_NON_RECOVERABLE	0x13	モジュール内部で回復できないエラーが発生した時に出られます。
E_IPJ_ERROR_TEMPLATE_DECODE_FAILURE	0x14	モジュールがコマンドテンプレートを正しくデコードできなかった場合に出られます。
E_IPJ_ERROR_SYSTEM_IN_ERROR_STATE	0x15	モジュールがエラーの状態に回復できないエラーが発生した場合に出られます。
E_IPJ_ERROR_TEST_ERROR	0x16	IRI ポストが設定によって、エラー制御テストを実行できるようにモジュールコンフィグした場合に出られます。
E_IPJ_ERROR_STORED_SETTING_DECODE	0x17	デコードストレージ設定が特定のキーコードで失敗した時に出られます。
E_IPJ_ERROR_VALUE_INDEX_OUT_OF_RANGE	0x18	無効な value_index でキーを設定または取得する場合に出られます。
E_IPJ_ERROR_BANK_INDEX_OUT_OF_RANGE	0x19	無効な bank_index でキーを設定または取得する場合に出られます。
E_IPJ_ERROR_INVALID_PRODUCT_CALIBRATION	0x1A	無効なキャリブレーションデータを含むモジュールに対して start () API 関数を呼び出す場合に出られます。
E_IPJ_ERROR_REPORT_SIZE_WOULD_OVERFLOW	0x1B	要請したレポートフィールドとデータが IRI タグオペレーションレポートのバッファーにオーバーフローす

		る場合に出られます。
E_IPJ_ERROR_FIXED_HARDWARE_SETTINGS_LOAD_ERROR	0x1C	モジュールのファームウェアがキャリブレーション時に組み込んだ特定のハード設定をロードできなかった場合に出られます。
E_IPJ_ERROR_GEN2_TAG_OTHER_ERROR	0x1000001	タグエラーcatch-allが他のコードに覆われないタグエラーを表します。
E_IPJ_ERROR_GEN2_TAG_MEMORY_OVERRUN	0x1000002	無効のパラメータでタグをアクセスする場合に出られます。（例：無効のメモリロケーション、無効の EPC 長さフィールドなど）
E_IPJ_ERROR_GEN2_TAG_MEMORY_LOCKED	0x1000003	タグのメモリロケーションをアクセスできません。
E_IPJ_ERROR_GEN2_TAG_INSUFFICIENT_POWER	0x1000004	タグメモリに書き込みを行う時、かつタグが応答しなかったエラーが出る場合に出られます。
E_IPJ_ERROR_GEN2_TAG_NON_SPECIFIC_ERROR	0x1000005	タグがサポートしないエラーコードが出る場合に出られます。
E_IPJ_ERROR_API_DEVICE_NOT_INITIALIZED	0x2000001	IRI ポストライブラリーが iri_device 構造を正しく初期化してない場合に出られます。
E_IPJ_ERROR_API_CONNECTION_PORT_ERROR	0x2000002	IRI ホストライブラリがモジュールとの IRI 通信の実装に使用される UART シリアルポートを制御できない場合に出られます。
E_IPJ_ERROR_API_CONNECTION_READ_TIMEOUT	0x2000003	IRI ホストライブラリがモジュールからの UART トラフィックを待っている間にタイムアウトが発生した場合に出られます。
E_IPJ_ERROR_API_CONNECTION_WRITE_TIMEOUT	0x2000004	IRI ホストライブラリがデータを UART に書き込む時にタイムアウトが発生した場合出られます。
E_IPJ_ERROR_API_CONNECTION_WRITE_ERROR	0x2000005	IRI ホストライブラリが UART で IRI パケットを送信失敗した時に出られます。
E_IPJ_ERROR_API_RX_BUFF_TOO_SMALL	0x2000006	IRI ホストが受け取ったバッファーが小さすぎる場合に出られます。
E_IPJ_ERROR_API_MESSAGE_INVALID	0x2000007	IRI ホストが無効のメッセージを検出した場合に出られます。
E_IPJ_ERROR_API_NO_HANDLER	0x2000008	IRI ホストライブラリの iri_device が有効なハンドラ関数を定義してない場合に出られます。

E_IPJ_ERROR_API_INVALID_LOADER_BLOCK	0x2000009	ブートロード操作中に、IRI ホストライブラリの ipj_flash_handle_loader_block () API が無効の CRC を含んでいるデータブロックを渡した場合に出られます。
E_IPJ_ERROR_API_RESPONSE_MISMATCH	0x200000A	IRI ホストライブラリがコマンドを発送した後、違うレスポンスを受け取る場合に出られます。
E_IPJ_ERROR_API_INVALID_PARAMETER	0x200000B	無効のパラメータで IRI ホストライブラリの API を呼び出す時に出られます。
E_IPJ_ERROR_API_NON_LT_PACKET_DETECTED	0x200000C	ホストを IRI-LT にコンフィグして、モジュールが IRI-LT 以外のパケットを発送した場合に出られます。
E_IPJ_ERROR_IRI_FRAME_DROPPED	0x3000001	IRI ホストライブラリがモジュールから適応のパケットを検出できなかった場合に出られます。
E_IPJ_ERROR_IRI_FRAME_INVALID	0x3000002	計算された IRI フレーム CRC またはパリティが送信された値と一致しなかった場合に出られます。
E_IPJ_ERROR_MAC_GENERAL	0x4000001	モジュール内で不明な通信エラーが発生した場合に出られます。
E_IPJ_ERROR_MAC_CRC_MISMATCH	0x4000002	モジュールが検出されたタグのレスポンス CRC がマッチングしなかった場合に出られます。
E_IPJ_ERROR_MAC_NO_TAG_RESPONSE	0x4000003	モジュールがタグのレスポンスを検出できなかった場合に出られます。
E_IPJ_ERROR_MAC_TAG_LOST	0x4000004	モジュールはタグがトランザクション中に失ったことを検出しました。
E_IPJ_ERROR_BTS_DEVICE_WATCHDOG_RESET	0x5000001	モジュールのウォッチドッグリセット機能が回復できないファームウェア動作状態になっていることを検出して、リセットを行った場合に出られます。
E_IPJ_ERROR_BTS_VALUE_INVALID	0x5000002	モジュールのブートロードがコマンドは無効のコマンド引数を含むポストから送信されたことを検出した場合に出られます。
E_IPJ_ERROR_BTS_FLASH_WRITE	0x5000003	モジュールのブートローダが IRI ホストの指示どおりにフラッシュ書き込みを実行できない場合に出られます。

E_IPJ_ERROR_BTS_FLASH_READ	0x5000004	ジュールのブートローダが IRI ホストの指示どおりにフラッシュ読み取りを実行できない場合に出られます。
E_IPJ_ERROR_BTS_FLASH_ADDRESS	0x5000005	アドレスが保護されるためモジュールのブートローダが IRI ホストの指示どおりにフラッシュ消去を実行できない場合に出られます。
E_IPJ_ERROR_BTS_FLASH_ERASE	0x5000006	モジュールのブートローダが IRI ホストの指示どおりにフラッシュ消去を実行できない場合に出られます。
E_IPJ_ERROR_BTS_UNKNOWN_COMMAND	0x5000007	モジュールのブートローダが IRI ホストから不明なコマンドを受け取った場合に出られます。
E_IPJ_ERROR_BTS_COMMAND_DECODE_FAILURE	0x5000008	モジュールのブートローダが IRI ホストから受け取ったコマンドをデコードできない場合に出られます。
E_IPJ_ERROR_TRANSCEIVER_FAILURE	0x6000001	モジュール内の無線トランシーバとの内部通信が失敗した時に出られます。
E_IPJ_ERROR_LIMIT_PA_TEMPERATURE_MAX	0x7000001	パワーアンプの温度が最大許容値を超える場合に出られます。

付録Ⅱ：

1. InventoryResult

タグをインベントリーする時に表示される項目
rssi
channel
phase
antenna
TagData tagData

2. TagData

タグのデータ
pc
epc
tid
data

付録Ⅲ：

デバイス情報
MacAddress
version
status
ipAddress
subnetAask
gateway
dns
dhcp
password

付録Ⅳ：

TagMask
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2

TagAction
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2