unitech

ScanServer/ScanAgent 操作マニュアル

V1.02

2010.11.17

目　　次

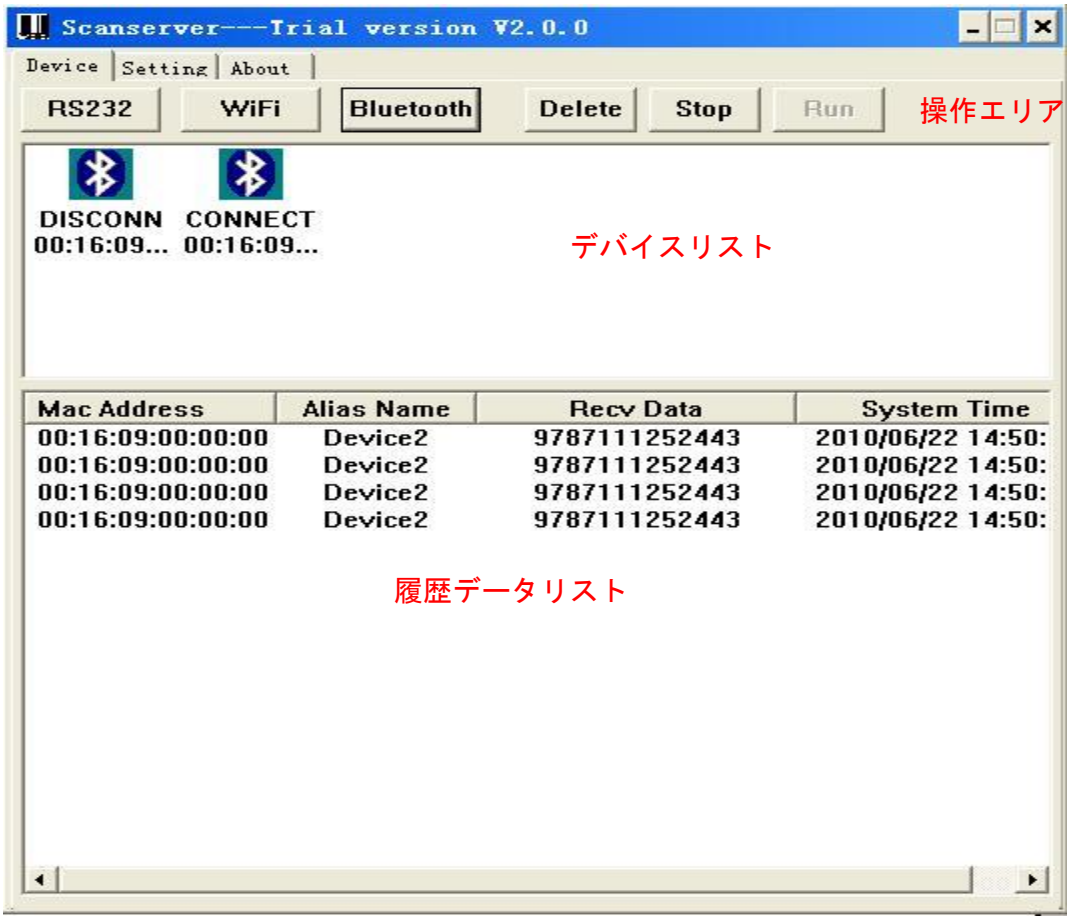
1. 概要 4
1.1. ユーザインターフェース 4
1.2. 実行環境 5
2. クイックスタートガイド 5
3. ScanServer の操作 6
3.1. 一般的な操作 6
3.2. RS232 デバイスをデバイスリストに追加 7
3.3. WiFi デバイスをデバイスリストに追加 9
3.4. Bluetooth デバイスをデバイスリストに追加 10
3.5. ScanServer の詳細設定 11
4. ScanAgent の操作 13
4.1. 一般的な操作 13
4.2. ScanAgent の詳細設定 14
5. ScanServer APIs プログラミングについて 15
5.1. ScanServer API の概要 15
5.2. データ構造と定義 15
5.2.1. メッセージ定義 15
5.2.2. 構造体定義 15
5.3. ScanServer ライブラリの初期化 20
5.3.1. ライブラリのパラメータを初期化 20
5.3.2. ライブラリへ登録 20
5.3.3. ライブラリのパラメータを検索 20
5.3.4. ライブラリのパラメータを変更 20
5.4. デバイス管理と操作 21
5.4.1. RS232 デバイスをデバイスリストに追加 21
5.4.2. WiFi デバイスをデバイスリストに追加 21

5.4.3. 近くの Bluetooth デバイスを検索 21
5.4.4. Bluetooth デバイス情報を検索 22
5.4.5. Bluetooth デバイスをデバイスリストに追加 22
5.4.6. デバイスの数を検索 22
5.4.7. デバイスリストからデバイスを削除 22
5.4.8. デバイスを停止 23
5.4.9. デバイスを開始 23
5.4.10. デバイスのステータスを検索 23
5.4.11. デバイス情報を検索 23
5.4.12. デバイスに正しいメッセージを応答 23
5.4.13. デバイスに正しくないメッセージを応答 24
5.4.14. インデックスによってデバイス ID を検索 24
5.4.15. ID によってデバイスインデックスを検索 24
5.4.16. RS232 デバイスをデバイスリストに追加 25
6. Dot Net Framework 用の ScanServerDllCF 26
7. Web ページでデータ受信するための PcScannerActiveX コントロールの使い方 26
7.1 レジスターコントロール 26
7.2. Html に埋め込む 26
7.3. Channel1 を有効/無効 26
7.4. DataReady イベント処理 27
8. FAQ 27
8.1 "Translate to ActiveX Control"をチェックしても Web ページでデータを受け取れないのはどうしてですか 27

1. 概要

ScanServer は、PC 側で動作するミドルウェアで、デバイスの管理とリモートデバイスからデータを受信して保存します。ユニテックは、お客様のシステムに組み込むために API もユーザに提供しています。ScanAgent は、ターミナル側で実行するミドルウェアです。そして、データの収集、一時的なデータの保存、そして RS232、Bluetooth そして WiFi を通して通信を行います。

1.1. ユーザインターフェース



ScanServer

ScanAgent

1.2. 実行環境

ScanServer: Windows XP, Vista, Windows7

ScanAgent: Unitech ポータブルコンピュータ(HT、PA シリーズ)

2. クイックスタートガイド

1) ScanServer をインストールするには、”Scanserver Setup V2.0.0.7.exe”をダブルクリックします。

2) C:¥Unitech¥ScanserverV2.0.0.7¥ScanAgent フォルダの全ファイルをターミナルの Flash Storage フォルダにコピーします。そして、 ScanAgent を起動します。

3) 初めて起動する場合、 ScanAgent を適当に設定しなければなりません。 ScanAgent の操作エリアにある”Setup”ボタンをクリックします。そして通信インターフェースを選択します。

ノート：無線インターフェースを選択する場合、先ずネットワークに接続しなければなりません。そしてローカルの IP アドレスが表示されます。Bluetooth インターフェースを選択した場合、Bluetooth モジュールの電源が入り、ローカル MAC アドレスが表示されるまで待って下さい。

3）PC で ScanServer を実行します。

4)選択した通信インターフェースにより、ScanServer にデバイスを追加するために正しいボタンを選択してクリックします。

5）複数のデバイスがある場合に、上記の操作を繰り返して ScanServer にデバイスを追加します。

3. ScanServer の操作

3.1. 一般的な操作

各デバイスは、三つの状態：Stop(停止)、Disconnect(切断) と Connect(接続)があります。

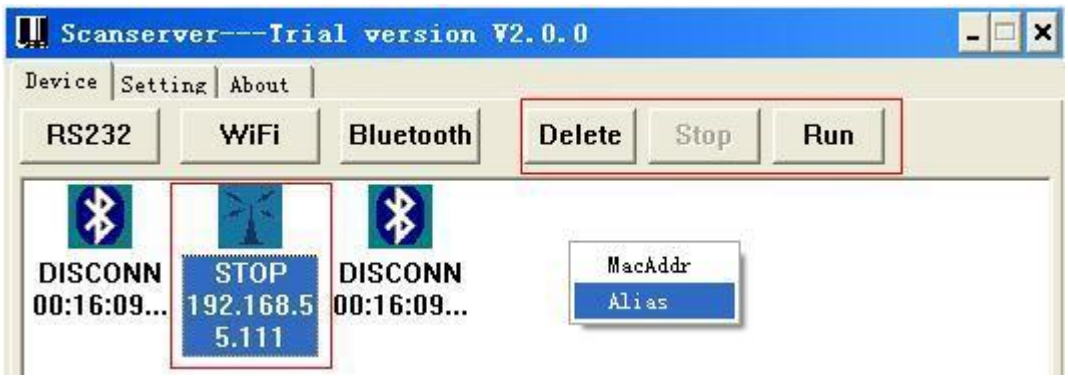
Stop：この状態では、デバイスは ScanServer に接続しようとはせず、データを ScanServer に送信することができません。

Disconnect：この状態では、デバイスは ScanServer に接続しようとしますが、ScanServer にデータを送信することはできません。

Connect: この状態では、デバイスはデータを ScanServer に送信することが出来、そして ScanServer から応答文字列を受け取ることが出来ます。

Delete(削除)、Stop(停止) と Run(実行) の操作:

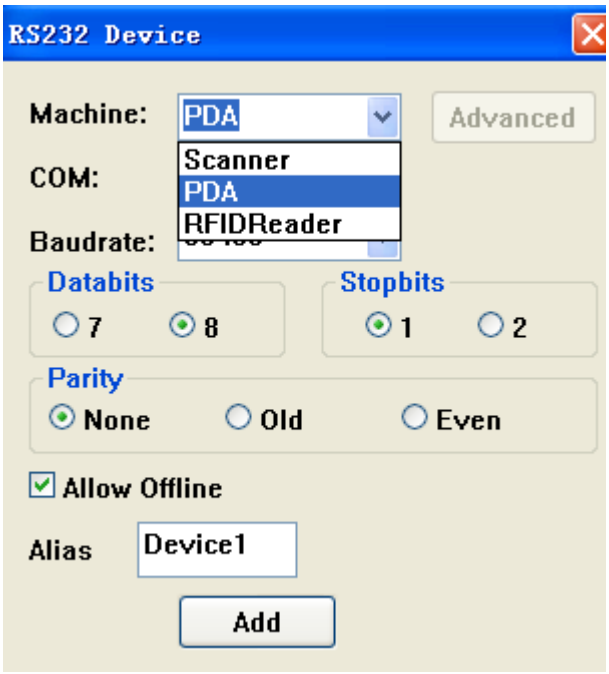
デバイスリストのデバイスを選択して、削除、停止、あるいは実行するデバイスのボタンをクリックします。



ノート: デバイスリストで右クリックすると、ユーザは MAC アドレスとわかりやすい名前を切り換えて表示することが出来ます。

3.2. RS232 デバイスをデバイスリストに追加

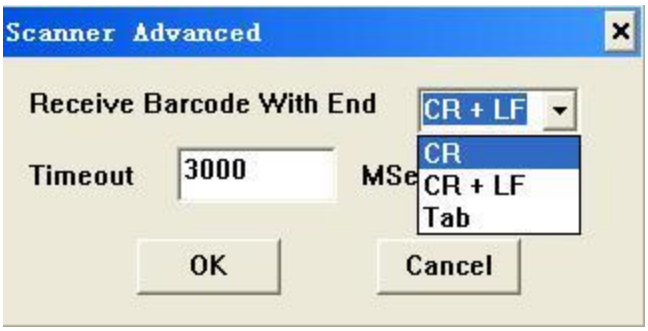
RS232 を通してデバイスと接続するには、ScanServer の操作エリアの "RS232" ボタンをクリックします。



RS232 デバイスを追加

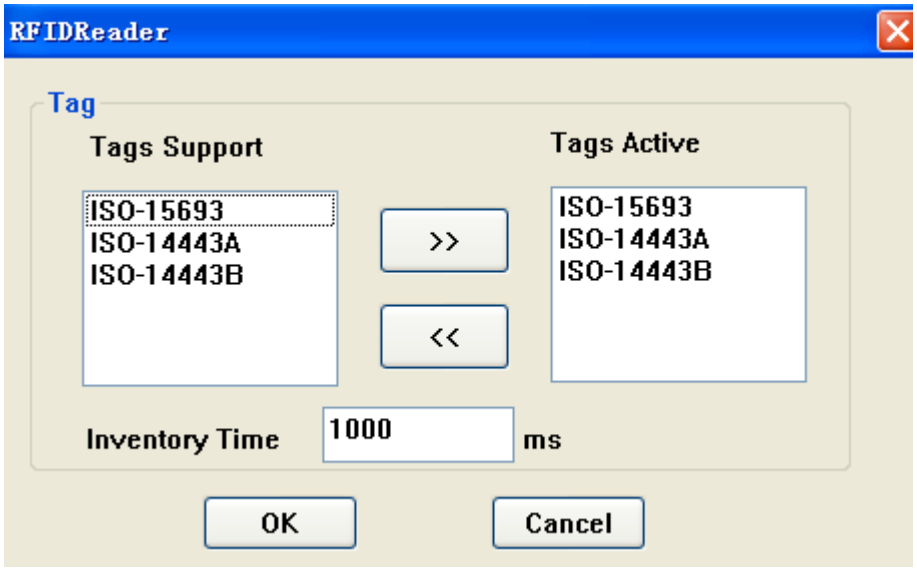
Machine:Scanner/PDA/RFIDReader

ここで、接続するデバイスのタイプを選択します。ターゲットデバイスがスキャナの場合、"Advanced" ボタンをクリックして、"Scanner Advanced" ダイアローグに入り、スキャナと通信をする前にパラメータを設定しなければなりません。



Scanner Advanced

ターゲットデバイスが RFIDReader の場合、"Advanced" ボタンをクリックし、" RFIDReader Advanced" ダイアローグに入り、RFIDReader と通信を始める前にパラメータを設定しなければなりません。

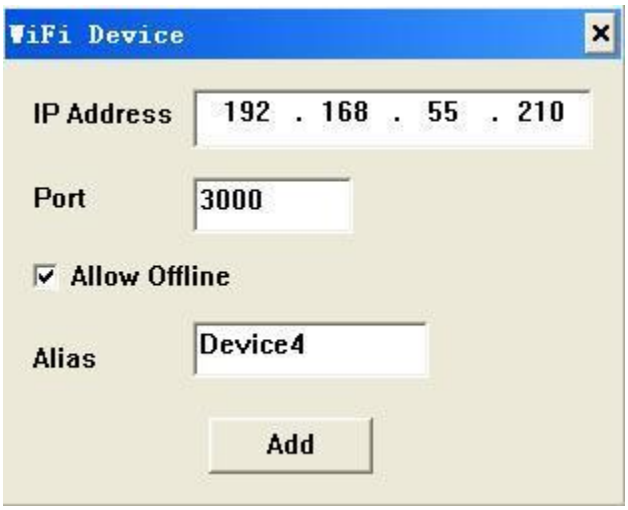


RFIDReader Advanced

Allow Offline: "allow offline" がチェックされると、デバイスは動作していなくてもかまいません。ユーザは前もって ScanServer にデバイスを追加することが出来、デバイスが物理的に接続されると、 ScanServer はデバイスを接続しすぐにデータの受信を開始します。

3.3. WiFi デバイスをデバイスリストに追加

WiFi を通してデバイスを接続するために、操作エリアの "WiFi" ボタンをクリックします。



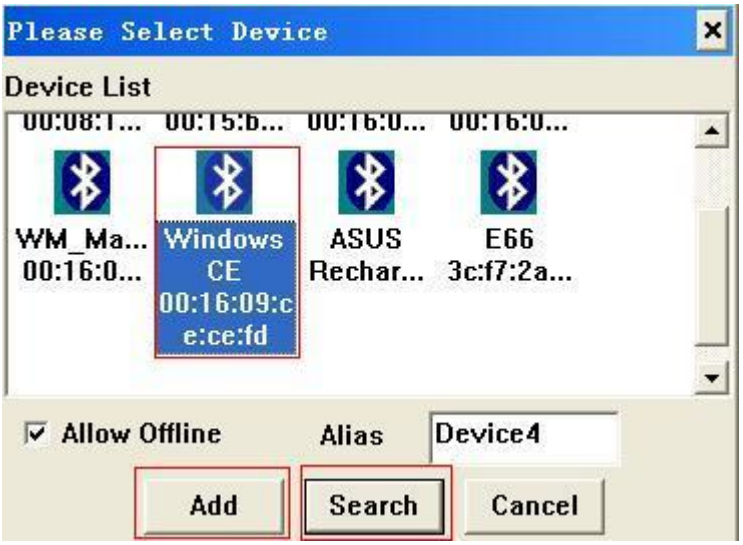
WiFi デバイスの追加

リモートデバイスの IP アドレスと通信ポート(ポート番号はリモートデバイスと同じでなければなりません)を入力します。

Allow Offline: "Allow offline" がチェックされると、デバイスは動作していなくてもかまいません。ユーザは前もって ScanServer にデバイスを追加することが出来、デバイスが物理的に接続されると、 ScanServer はデバイスを接続しすぐにデータの受信を開始します。

3.4. Bluetooth デバイスをデバイスリストに追加

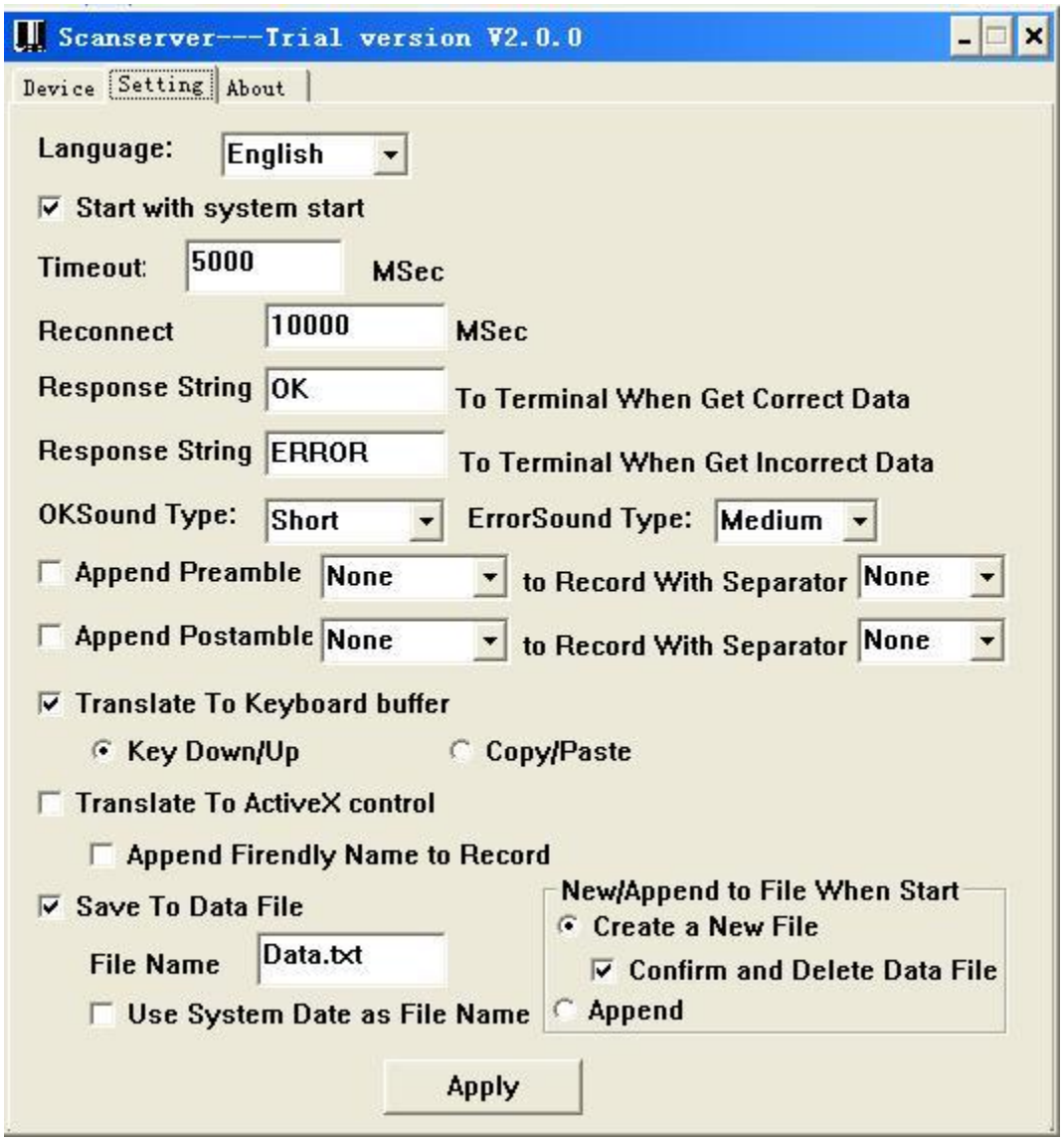
Bluetooth を通してデバイスを接続する場合、ScanServer の動作エリアの"Bluetooth"ボタンをクリックします。



Bluetooth デバイスの追加

Bluetooth デバイスを探すために最初に "Search" ボタンをクリックします。次にターゲットデバイスを選択して "Add" ボタンをクリックします。

ノート: ターゲットデバイスが" デバイスリスト" に無い場合、再度検索して下さい。デバイスを表示するのに二三回検索が必要な場合があります。



3.5. ScanServer の詳細設定

詳細設定

1)Language：English

2)Windows 起動時に ScanServer を開始

3)Timeout:データの最大応答時間

4)Reconnect:デバイスが ScanServer から切断された場合、ScanServer は 10 秒ごとに

再接続を試みます。

5)Response string: ScanServer がデータを受信するときにデバイスに送信する標準の文字列

6)OK/ERROR sound Type:リモートデバイスは、ScanServer が受信しているデータに応答したときにビープ音を発します。

7)Append prefix/suffix: データをデータファイルに保存するとき、どのデバイスから来たデータかを区別するためにプリフィクス/サフィックスを使用することができます。

8)Translate to keyboard buffer: データをキーボードバッファに転送すると、ユーザは受信データを MS-Word、メモ帳、ERP 等のアプリケーションでで使用することができます。

9)Translate to ActiveX control: ActiveX Control へデータを転送すると、ユーザは受信データを Web ページで扱うことが出来ます。

ノート: ユーザは、この機能をテストするためにサンプルページ "ScannerCtl_Channel.HTM" を試すことができます。

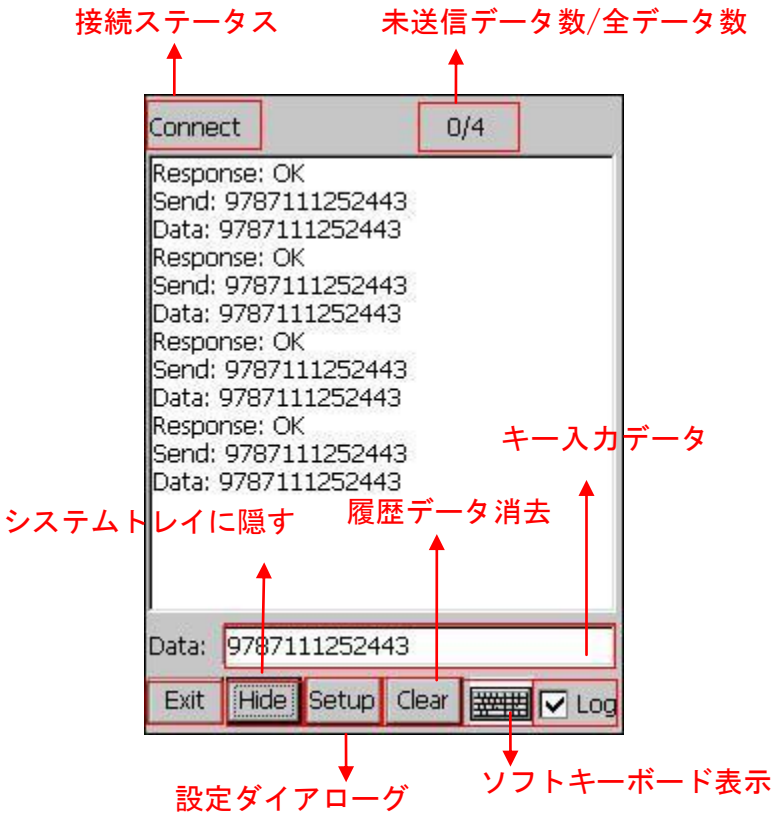
10)Append Friendly Name to Record: どのデバイスから来たデータかを区別するために使用します。

11)Save to data file: データをファイルに保存します。

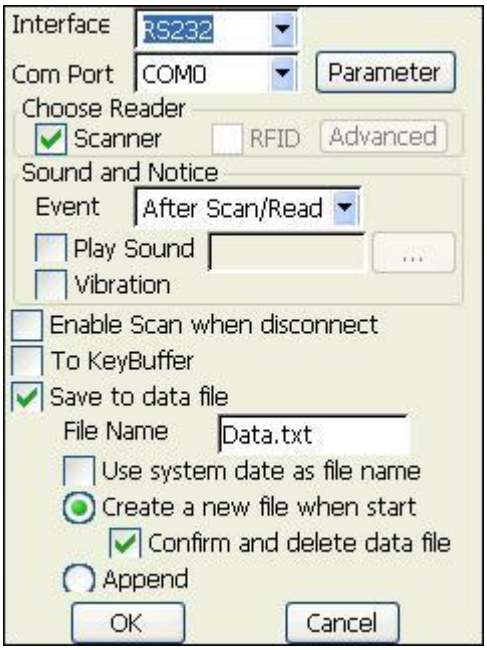
12)データを新しいファイルに保存するか、あるいは既存のファイルに追加するかどうか指定します。

4. ScanAgent の操作

4.1. 一般的な操作



4.2. ScanAgent の詳細設定



1）インターフェース: RS232/WiFi/Bluetooth

2）サポートしているリーダ: Scanner/RFID

3）Sound and Notice: あるイベントが発生したときに音またはバイブレーションで知らせる。

Event: イベントは、After Scan/Read（スキャン/読み取りの後）、Correct Data（正しいデータ）、Incorrect Data（正しくないデータ）、Disconnect（切断）、Connect（接続）

Play Sound：音を鳴らすサウンドファイルを指定します。

Vibration：バイブレーションを使うかどうか

4）Enable Scan when disconnect：この項目がチェックされた場合、デバイスはステータスがConnect（接続）の場合のみデータを読みます。

5）To KeyBuffer：データをキーボードバッファに転送します。

6）Save to data file：データファイルに保存するかどうか。

7）Create a new file：新しいファイルにデータを保存するか、あるいは既存のファイルに追加するかどうか。

5. ScanServer APIs プログラミングについて

5.1. ScanServer API の概要

ScanserverDll は、ScanServer と共にリリースされています。 この DLL は、本機能を使用したシステムの開発と組込を容易に行うことが出来ます。

実装されている概要は次の通りです:

1. ライブラリとパラメータを初期化します
2. ライブラリを登録する。アプリケーションはメッセージの受信が可能になります。
3. デバイスをライブラリに追加します
4. デバイスのメッセージと応答を処理します

5.2. データ構造と定義

5.2.1. メッセージ定義

#define WM_CHECKDATA WM_USER + 997

新しいデータ入力メッセージ。このメッセージを受信したときに、データが正しいかどうかを検証しなければなりません。データが正しい場合、デバイスに応答するために SendCorrectMsg をコールし、そしてアプリケーションは WM_SAVEDATA メッセージを受け取ります。データが正しくない場合は、デバイスに応答するために SendIncorrectMsg をコールし、そしてアプリケーションは WM_SAVEDATA メッセージを受信しません。

#define WM_SAVEDATA WM_USER + 996

アプリケーションがこのメッセージを受信したとき、データを保存しなければなりません。

#define WM_STATUSCHANGE WM_USER + 995

デバイスのステータスが変わったときに、アプリケーションはこのメッセージを受信します。

5.2.2. 構造体定義

以下は最初に初期化されなければならない ScanServer ライブラリの構造体です。

```
typedef struct _SETTING_INFO{
    BOOL        m_bIndividual;
        char        m_szCorrectRes[66]; //default message
        char        m_szIncorrectRes[66]; //default message
        int         m_nRetryTimes; //Reserve
        int         m_nTimeout;
        int         m_nReconnect;
        int         m_nRecvEnd; //ASCII value
        int         m_nBarcodeTimeout;
        BOOL        m_bPreamble;
        BOOL        m_bPostamble;
        int         m_nPreambleMode;
        int         m_nPostambleMode;
        int         m_nPreambleSpara;
        int         m_nPostambleSpara;
        BOOL        m_bToKeyBuffer; //translate to key buffer
        BOOL        m_bKeyEvent;//Key Down/Up
        BOOL        m_bCopy;//Copy/Paste
        BOOL        m_bActiveX;//ActiveX
        BOOL        m_bAppendFirendly;
        BOOL        m_bSaveToFile;    //save to data file
        char        m_szSaveFileName[MAX_PATH];
        int         m_nOKSoundType;
        int         m_nErrorSoundType;
        MACHINE_TYPE m_nMachineType;
        int         m_nInventoryTime;
}SETTING_INFO, *LPSETTING_INFO;
```

m_bIndividual：このパラメータが FALSE にセットされた場合、ScanServer ライブラリは、WM_CHECK メッセージを受信したときにデバイスにメッセージを自動応答します。
そうでなければ、アプリケーションはデバイスにメッセージを応答するために SendCorrectMsg/SendIncorrectMsg をコールしなければなりません。

m_szCorrectRes[66]： ScanServer ライブラリが正しいデータを受信した時にデバイスに送信される標準メッセージ

m_szIncorrectRes[66]：ScanServer ライブラリが正しくないデータを受信したときにデバイスに送信される標準メッセージ

m_nRetryTimes：//予約(未使用、デバイスは常に再送信)

m_nTimeout： m_bIndividual が TRUE にセットされた場合、アプリケーションは m_nTimeout ミリ秒以内にデバイスは応答しなければなりません。そうでなければ、デバイスはデータを再送信します。

m_nReconnect：デバイスが ScanServer から切断された場合、ScanServer は m_nReconnect ミリ秒ごとに再接続を試みます(m_nReconnect は、m_nTimeoutより大きくなければなりません。)

m_nRecvEnd：バーコードのターミネータ文字(デバイスがスキャナの場合､このパラメータをセットしなければなりません。m_nRecvEnd = 13 は、ターミネータ文字が CRであることを示します。)

m_nBarcodeTimeout:バーコードのタイムアウト(デバイスがスキャナの場合、このパラメータをセットしなければなりません。)

m_bPreamble：データにプリフィックスを付けて、データファイルに保存するかどうか。

m_bPostamble：データにサフィックスを付けて、データファイルに保存するかどうか。

m_nPreambleMode:プリフィックスデータ 0:MacAddr 1:Alias 2.MacAddr + Alias 3. Alias+ MacAddr

m_nPostambleMode:サフィックス 0:MacAddr 1:Alias 2.MacAddr + Alias 3. Alias+ MacAddr

m_nPreambleSpara:データとプリフィックスの区切り文字

0：なし 1：, 2： ; 3：Tab

m_nPostambleSpara：データとサフィックスの区切り文字

0：なし 1：, 2： ; 3：Tab

m_bToKeyBuffer:キーボードバッファに転送される文字

m_bKeyEvent: キーボードバッファ転送時にキーダウン/アップイベントを生成

m_bCopy: キーボードバッファ転送時にコピー/ペーストイベントを生成

m_bActiveX:ActiveX コントロールにデータを転送

m_bAppendFirendly:データにわかりやすい名前を付けて、ActiveX コントロール転送

m_bSaveToFile:データファイルに保存

m_szSaveFileName[MAX_PATH]: データファイル名

m_nOKSoundType:標準の警告ビープ音 0:なし 1: 0x03: 中ビープ 0x04:二つの中ビープ 0x05:長いビープ

m_nErrorSoundType: 標準の警告ビープ音

nMachineType: Scanner/PDA/RFIDReader.

m_nInventoryTime: RFIDReader インベントリ時刻

```
typedef struct RS232_PARAMETER_
{
    DWORD dwBaudrate;
    WORD nDatabits;
    WORD nStopbits;
    WORD nParity;
}RS232_PARAMETER, *LPRS232_PARAMETER;

typedef enum _MACHINE_TYPE
{
    MAC_UNKNOW = 0,
    PDA,
    SCANNER,
    READER,
```

```
}MACHINE_TYPE;

typedef struct TAG_
{
    BOOL        bTag_15693;
    BOOL        bTag_14443A;
    BOOL        bTag_14443B;
}TAG;

typedef enum    TAG_TYPE_
{
    ISO_15693 =1,
    ISO_14443A,
    ISO_14443B,
}TAG_TYPE;

typedef struct DEVICE_INFO_
{
    DEVICE_TYPE m_nDeviceType;
    UINT nDeviceID;//デバイスハンドラー
    int nPort;          //RS232 では COM ポート番号、WiFi では IP ポート番号;
    RS232_PARAMETER pParam; //RS232 Com ポートパラメータ
    BOOL bMulti; //スキャナはFalse; PDA は True
    char szName[20];    //デバイス名(使用しない)
    char szFriendlyName[20]; //わかりやすい名前
    char szIPAddr[20];   /WiFi
    char szMacAddr[20]; //Bluetooth MAC アドレス
    MACHINE_TYPE   nMachineType;
```

}DEVICE_INFO, *LPDEVICE_INFO;

5.3. ScanServer ライブラリの初期化

5.3.1. ライブラリのパラメータを初期化

関数コール: BOOL InitManager(LPSETTING_INFO pstInfo);

パラメータ: LPSETTING_INFO pstInfo: パラメータを初期化するためのパラメータの構造体

5.2.2.構造体定義を参照

戻り値: BOOL: TRUE/FALSE

5.3.2. ライブラリへ登録

ライブラリからメッセージを受信と処理をするウインドウまたはスレッドを登録する

関数コール: void SetRecvDataWnd(HWND hWnd);

パラメータ: HWND hWnd: メッセージを処理するウインドウのハンドル

戻り値: void

関数コール: void SetRecvThreadID(DWORD dwThreadID);

パラメータ:DWORD dwThreadID:メッセージを処理するスレッドの ID

戻り値: void

5.3.3. ライブラリのパラメータを検索

関数コール: void GetManagerParam(LPSETTING_INFO pstInfo);

パラメータ: LPSETTING_INFO pstInfo: ライブラリを初期化するためのパラメータの構造体

5.2.2.構造体定義を参照

戻り値: void

5.3.4. ライブラリのパラメータを変更

関数コール: BOOL ChangeManagerParam(LPSETTING_INFO pstInfo);

パラメータ: LPSETTING_INFO pstInfo: ライブラリを初期化するためのパラメータの構造体

5.2.2. 構造体定義を参照

戻り値: BOOL: TRUE/FALSE

5.4. デバイス管理と操作

5.4.1. RS232 デバイスをデバイスリストに追加

関数コール: int AddRS232Device(int nPort, LPRS232_PARAMETER pszParameter, BOOL bMulti = TRUE, char *pszFriendlyName = NULL, BOOL bOnline = TRUE);

パラメータ: int nPort: Com ポート番号

LPRS232_PARAMETER pszParameter:RS232 の転送速度と他のパラメータ

BOOL bMulti:スキャナは FALSE; PDA は TRUE

char *pszFriendlyName:デバイスを区別するために使用するわかりやすい名前

BOOL bOnline:FALSE は、実際のデバイスが無くても ScanServer にデバイスを追加できることを示す

戻り値: int: デバイスをコントロールするために使用するデバイス ID

5.4.2. WiFi デバイスをデバイスリストに追加

関数コール: int AddWIFIDevice(char *pszAddress, UINT nPort, BOOL bMulti = TRUE, char *pszFriendlyName = NULL, BOOL bOnline = TRUE);

パラメータ: char *pszAddress:WiFi デバイスの IP アドレス

UINT nPort:通信の Socket ポート

BOOL bMulti: WiFi デバイスの場合 TRUE

char *pszFriendlyName: デバイスを区別するために使用するわかりやすい名前

BOOL bOnline:FALSE は、実際のデバイスが無くても ScanServer にデバイスを追加できることを示す

戻り値: int: デバイスをコントロールするために使用するデバイス ID

5.4.3. 近くの Bluetooth デバイスを検索

関数コール: int SearchBluetoothDevice();

パラメータ: なし

戻り値: int: 見つかった Bluetooth デバイスの数

5.4.4. Bluetooth デバイス情報を検索

関数コール: void GetBluetoothDeviceInfo(int nIndex, LPWSTR pszName, LPWSTR pszAddr);

パラメータ: int nIndex:Bluetooth デバイスのインデックス

LPWSTR pszName:Bluetooth デバイス名を含む

LPWSTR pszAddr: Bluetooth デバイスの MAC アドレスを含む

戻り値: void

5.4.5. Bluetooth デバイスをデバイスリストに追加

関数コール: int AddBluetoothDevice(char *pszAddress, BOOL bMulti = TRUE, char *pszFriendlyName = NULL, BOOL bOnline = TRUE);

パラメータ: char *pszAddress: Bluetooth デバイスの MAC アドレス

BOOL bMulti: WiFi デバイスは TRUE

char *pszFriendlyName: デバイスを区別するために使用するわかりやすい名前

BOOL bOnline:FALSE は、実際のデバイスが無くても ScanServer にデバイスを追加できることを示す

戻り値: int: デバイスをコントロールするのに使用されたデバイス ID

5.4.6. デバイスの数を検索

関数コール: int GetDeviceCount();

パラメータ:

戻り値: int: ScanServer に追加されたデバイスの数

5.4.7. デバイスリストからデバイスを削除

関数コール: int DeleteDevice(UINT nDeviceID);

パラメータ: UINT nDeviceID: 削除されるデバイスのデバイス ID

戻り値: int: 削除されたデバイス ID

5.4.8. デバイスを停止

関数コール: BOOL StopDevice(UINT nDeviceID);

パラメータ: UINT nDeviceID: 停止するデバイスのデバイス ID

戻り値: BOOL:TRUE/FALSE

5.4.9. デバイスを開始

関数コール: BOOL StopDevice(UINT nDeviceID);

パラメータ: UINT nDeviceID: 開始するデバイスのデバイス ID

戻り値: BOOL:TRUE/FALSE

5.4.10. デバイスのステータスを検索

関数コール: STATUS GetDeviceStatus(UINT nDeviceID);

パラメータ: UINT nDeviceID: デバイスのデバイス ID

戻り値: STATUS : デバイスのステータス

5.4.11. デバイス情報を検索

関数コール: void GetDeviceInfo(LPDEVICE_INFO pstDeviceInfo, UINT nDeviceID);

パラメータ: LPDEVICE_INFO pstDeviceInfo: デバイス情報を含む

5.2.2 構造体定義を参照

UINT nDeviceID:The デバイスのデバイス ID

戻り値: void

5.4.12. デバイスに正しいメッセージを応答

WM_CHECKDATA メッセージを受け取ったときに、アプリケーションはデータが正しいかどうかを検証しなければなりません。データが正しい場合、アプリケーションはデバイスに応答するために SendCorrectMsg をコールし、そしてアプリケーションは WM_SAVEDATA メッセージを

受信します;

関数コール: void SendCorrectMsg(char *pszMsg, byte bSound, UINT nDeviceID);

パラメータ: char *pszMsg: デバイスに送られる文字列。SETTING_INFO の m_szCorrectRes を参照

byte bSound: ビープタイプ。 SETTING_INFO の **m_nOKSoundType** 参照

UINT nDeviceID:The デバイスのデバイス ID

戻り値: void

5.4.13. デバイスに正しくないメッセージを応答

WM_CHECKDATA メッセージを受け取ったときに、アプリケーションは、データが正しいかどうかを検証しなければなりません。データが正しくない場合、デバイスに応答するために SendIncorrectMsg をコールし、そしてアプリケーションは WM_SAVEDATA メッセージを受信しません。

関数コール: void SendIncorrectMsg(char *pszMsg, byte bSound, UINT nDeviceID);

パラメータ: char *pszMsg: デバイスに送信される文字列。SETTING_INFO の m_szIncorrectRes を参照

byte bSound:Beep type. SETTING_INFO の **m_nErrorSoundType** を参照

UINT nDeviceID:デバイスのデバイス ID

戻り値: void

5.4.14. インデックスによってデバイス ID を検索

関数コール: UINT GetDeviceIDByIndex(UINT nIndex);

パラメータ: UINT nIndex: デバイスのインデックス

戻り値: UINT: デバイスのデバイス ID

5.4.15. ID によってデバイスインデックスを検索

関数コール: int GetIndexByDeviceID(UINT nDeviceID);

パラメータ: UINT nDeviceID: デバイスのデバイス ID

戻り値: int: デバイスのインデックス

5.4.16. RS232 デバイスをデバイスリストに追加

関数コール: int AddRS232Device1(int nPort, LPRS232_PARAMETER pszParameter, BOOL bMulti = TRUE, char *pszFriendlyName = NULL, BOOL bOnline = TRUE, MACHINE_TYPE nMachineType = READER);

パラメータ: int nPort: COM ポート番号

LPRS232_PARAMETER pszParameter: RS232 の転送速度と他のパラメータ

BOOL bMulti: スキャナは FALSE; PDA は TRUE

char *pszFriendlyName: デバイスを区別するために使用するわかりやすい名前

BOOL bOnline: FALSE は、実際にデバイスが無くても ScanServer にデバイスを追加できることを示します。

MACHINE_TYPE nMachineType: マシンタイプは SCANNER/PDA/READER。

戻り値: int: デバイスをコントロールするために使用するデバイス ID

6. Dot Net Framework 用の ScanServerDllCF

ScanServerDllCF は、ネイティブの ScanServerDll の.NET ラッパーです。そして、メソッド、構造体、および ScanServerDllCF の開発プロセスは、ネイティブの DLL と同じです。
サンプルプログラムは、の使用方法のデモを含んでいます。 C++、C# と Vb.net バージョンは、SETUP フォルダにあります。

7. Web ページでデータ受信するための PcScannerActiveX コントロールの使い方

"Translate to ActiveX control" がチェックされた場合、ユーザは Web ページでデータを受け取るようにすることができます。データを受信するために PcScannerActiveX.ocx に 4 つのチャンネルがありますが、現在のところ最初のチャンネルのみが使用できます。
"ScannerCtl_Channel.HTM"を参照して下さい。

7.1 レジスターコントロール

レジスターコントロールをするには、 "Regsvr32.exe PcScannerActiveX.ocx" を実行します。

7.2. Html に埋め込む

```
<OBJECT ID="Scanner"
CLASSID="CLSID:B8FA0E16-3DBB-41C6-BFA1-35AA5DD6CE49"
WIDTH=0    HEIGHT=0>
</OBJECT>
```

7.3. Channel1 を有効/無効

```
<SCRIPT LANGUAGE="Javascript">
function OnStart1(){Scanner.Channel1=1;ShowChannel();}
function OnStop1(){Scanner.Channel1=0;ShowChannel();}
```

```
</SCRIPT>

<INPUT NAME="START1" TYPE="BUTTON" VALUE="Start" onClick="OnStart1()" >
<INPUT NAME="STOP1" TYPE="BUTTON" VALUE="Close" DISABLED ="true"
onClick="OnStop1()" 〉
```

7.4. DataReady イベント処理

```
<SCRIPT LANGUAGE="Javascript" FOR="Scanner"
EVENT="DataReady(channel,barcode,length)">
if(channel==1)
    BARCODE1.value=barcode
else
    alert("Unknown Source");
</script>
```

8. FAQ

8.1 "Translate to ActiveX Control"をチェックしても Web ページでデータを受け取れないのはどうしてですか

ActiveX コントロールに登録するために以下のステップを行って下さい:

1. WinXP では、 regsvr32 D:¥PcScannerActiveX.ocx を実行します。

2. Vista と Win7 ユーザは、PCScannerActiveX.ocx を登録する前に resvr32 を実行する前に UAC(ユーザアカウント制御)をオフにしなければなりません。

1)UAC をオフにするステップ

″Windows スタートメニュー -> コントロールパネル-> ユーザアカウントと家族のための安全設定 -> ユーザアカウント -> ユーザアカウント制御設定の変更をクリックして、-> スライダーを「通知しない」の位置に移動すると UAC はオフになります。

2) regsvr32 D:¥PcScannerActiveX.ocx を実行します。

3) UAC をオンにしないで下さい。

ここで、バーコードデータを受信するために ScannerCtl_Channel.HTM を実行することが出来ます。

4) UAC をオンにしたい場合、ActiveX コントロールに登録後に行います。

データ受信のために scannerCtl_Channel.HTM を実行する前に Internet Explorer のセキュリティモードを無効にしなければなりません。 Run Internet Explorer->ツール->インターネットオプション-> セキュリティタブ-> 「保護モードを有効にする」のチェックを外し、Internet Explorer を再起動します。

ここで、バーコードデータを受信するために ScannerCtl_Channel.HTM を実行することが出来ます。