CASIO®

WANGPRS ライブラリ
マニュアル

このマニュアルは、WANGPRS ライブラリの
仕様について記載します。

Ver. 1.07

ご注意

- このソフトウェアおよびマニュアルの、一部または全部を無断で使用、複製することはできません。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

変更履歴

バージョン	変更日付	ページ	内容
1.00	2009.09		新規作成
1.01	2010.07	9	RASDIALPARAMS 構造体の説明追加
1.02	2010.12	-	誤記訂正
1.03	2011.11	-	対象機種に IT-9000 を追加
	2012.01	42	WANSetRASSettingEx, WanSetRasSettingEx関数を追加
1.04	2012.09	7	wan_CONNECTSTATUS, RasConnStatusEx 構造体について、パラメータ値を追加
		37	WANGetCurrentOperator, WanGetCurrentOperator関数について、パラメータ値を追加
1.05	2015.02	-	対象機種に IT-G500 を追加
1.06	2016.02	44	WANRASConnect, WanRasConnect関数の pdwResult の解説を修正
1.07	2016.06	41	WANSetRASSetting, WanSetRasSetting関数の注意を追加

目次

1. 概要

WANGPRS ライブラリは、GPRS モジュールを用いたデータ通信の制御機能を提供します。

機能としては主に、以下があります。

- 電源制御
- SIM 制御
- GPRS 情報取得
- GPRS 通信情報取得
- GPRS 制御
- SMS 制御

2. 動作環境

WANGPRS ライブラリの動作環境を以下に示します。

対象機種

- DT-5300
- IT-9000
- IT-G500

対象 OS

- Microsoft Windows Mobile 6.5
- Microsoft Windows Embedded Handheld 6.5

開発環境

- Microsoft Visual Studio 2005 + SP1
- Microsoft Visual Studio 2008 + SP1

提供ファイル

- WANGPRSLib.h
- WANGPRSLib.lib
- WANGPRSLib.dll
- WANGPRSLibNet.dll (クラスライブラリ)

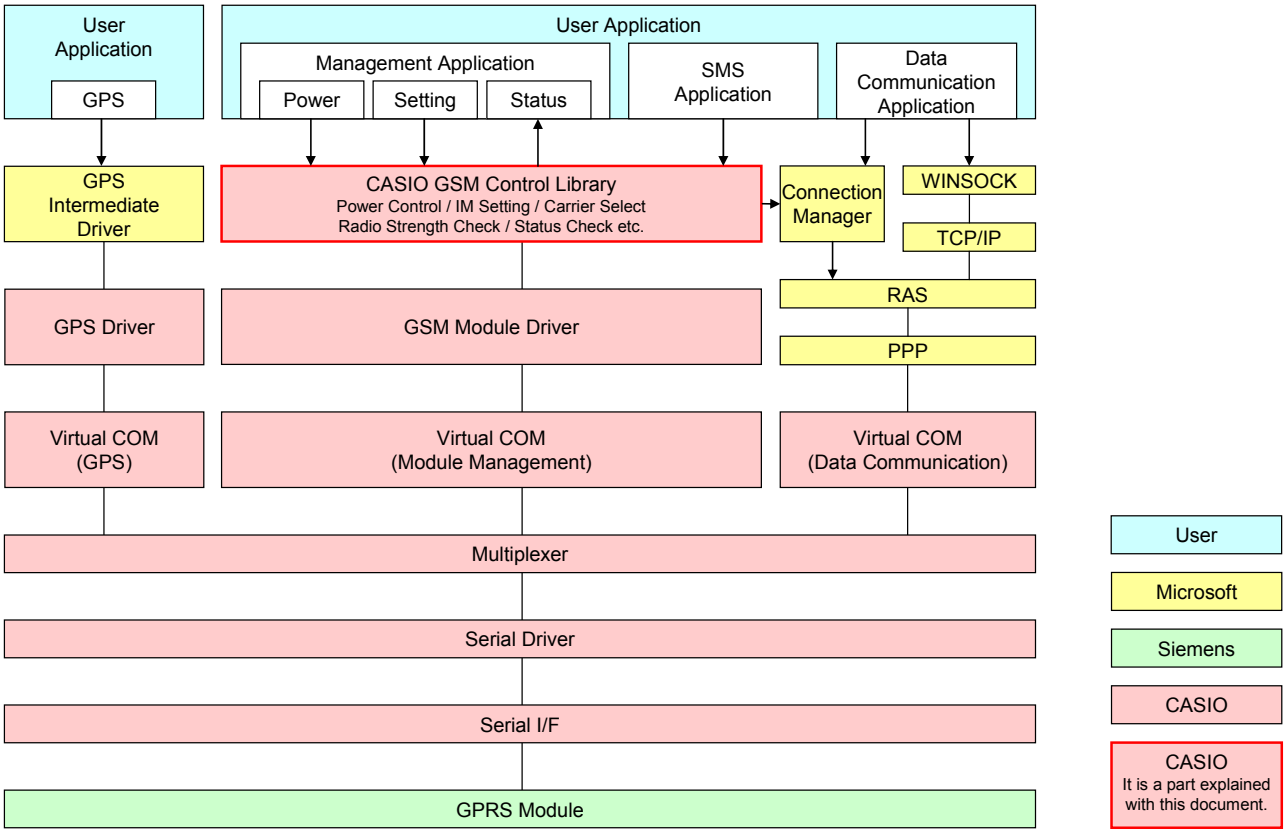
使用方法

- プログラムソース内に WANGPRSLib.h をインクルードし、WANGPRSLib.lib を使用するライブラリとして指定してください
- WANGPRSLib.dll は本体に内蔵されています。
- WANGPRSLibNet.dll を実行モジュールと同じフォルダにコピーしてください。

3. 機能一覧

3.1 ソフトウェア構成

下図のようなソフトウェア構成になっています。



3.2 構造体一覧

3.2.1 wan_ModuleInfo, WanModuleInfo 構造体

モジュール情報を格納します。

```
[C++]
struct wan_ModuleInfo {
  BYTE    szVendor[MODULEINFO_VENDOR_MAX_LENGTH];
  BYTE    szName[MODULEINFO_NAME_MAX_LENGTH];
  BYTE    szRevision[MODULEINFO_REVISION_MAX_LENGTH];
};
```

```
[Visual Basic]
Public Class WanModuleInfo
  Public  Vendor      As String
  Public  Name        As String
  Public  Revision    As String
End Class
```

```
[C#]
public class WanModuleInfo
{
  public  string  Vendor;
  public  string  Name;
  public  string  Revision;
};
```

パラメータ

szVendor
モジュールのメーカ名を格納します。

szName
モジュール名を格納します。

szRevision
モジュールのリビジョンを格納します。

3.2.2　wan_RSSI, WanRssi 構造体

GPRS の電波状態を格納します。

```
[C++]
struct wan_RSSI{
  long  nSignalStrengthMin;
  long  nSignalStrengthMax;
  long  nSignalStrength;
};
```

```
[Visual Basic]
Public Class WanRssi
  Public  SignalStrengthMin   As Long
  Public  SignalStrengthMax   As Long
  Public  SignalStrength      As Long
End Class
```

```
[C#]
public class WanRssi
{
  public  long  SignalStrengthMin;
  public  long  SignalStrengthMax;
  public  long  SignalStrength;
};
```

パラメータ

nSignalStrengthMin
RSSI がとる最小値[dBm]を格納します。

nSignalStrengthMax
RSSI がとる最大値[dBm]を格納します。

nSignalStrength
現在の RSSI[dBm]を格納します。

3.2.3　wan_OperatorInfo, WanOperatorInfo 構造体

オペレータ情報を格納します。

```
[C++]
Struct wan_OperatorInfo{
  BYTE  szLongName[MAXLENGTH_OPERATOR_LONG];
  BYTE  szShortName[MAXLENGTH_OPERATOR_SHORT];
  BYTE  szNumName[MAXLENGTH_OPERATOR_NUMERIC];
};
```

```
[Visual Basic]
Public Class WanOperatorInfo
  Public  LongName    As String
  Public  ShortName   As String
  Public  NumName     As String
End Class
```

```
[C#]
public class WanOperatorInfo
{
  public string LongName;
  public string ShortName;
  public string NumName;
};
```

パラメータ

szLongName
オペレータの名称を格納します(32 文字)。

szShortName
オペレータの名称を格納します(16 文字)。

szNumName
オペレータ ID を格納します。
国コード(3 文字)+ネットワークコード(2 文字)

3.2.4 wan_CONNECTSTATUS, RasConnStatusEx 構造体

GPRS の接続セッションステータスを格納します。

```
[C++]
Struct wan_CONNECTSTATUS{
  RASCONNSTATUS   rasconnstatus;
  DWORD           dwWanStatus;
} WAN_CONNECTSTATUS, *LPWAN_CONNECT_STATUS;
```

```
[Visual Basic]
Public Class RasConnStatusEx
  Public  Rasconnstatus   As RASCONNSTATUS
  Public  WanStatus       As Int32
End Class
```

```
[C#]
public class RasConnStatusEx
{
  public  RASCONNSTATUS   Rasconnstatus;
  public  Int32           WanStatus;
};
```

パラメータ

rasconnstatus

[C++]

RAS 接続ステータスを格納します。MSDN の RASAPI を参照してください。

[Visual Basic]、[C#]

RASCONNSTATUS 構造体を格納します。詳細は、RASCONNSTATUS 構造体を参照してください。

dwWanconnstatus

GPRS の接続ステータスを格納します。

WAN_STATUS_NOT_AVAILABLE	: GPRS/EGPRS 未対応
WAN_GPRS_AVAILABLE	: GPRS 対応
WAN_GPRS_ATTACHED	: GPRS 接続
WAN_EGPRS_AVAILABLE	: EGPRS 対応
WAN_EGPRS_ATTACHED	: EGPRS 接続
WAN_UMTS_AVAILABLE	: UMTS 対応
WAN_UMTS_ATTACHED	: UMTS 接続
WAN_HSDPA_AVAILABLE	: HSDPA 対応
WAN_HSDPA_ATTACHED	: HSDPA 接続
WAN_HSUPA_AVAILABLE	: HSUPA 対応
WAN_HSUPA_ATTACHED	: HSUPA 接続
WAN_HSPA_AVAILABLE	: HSDPA+HSUPA 対応
WAN_HSPA_ATTACHED	: HSDPA+HSUPA 接続
WAN_GSM_AVAILABLE	: GSM 対応
WAN_GSM_ATTACHED	: GSM 接続
WAN_STATUS_UNSUPPORT	: 未対応のネットワーク

3.2.5　RASCONNSTATUS 構造体

RAS 接続データを格納します。

```
[Visual Basic]
Public Class RASCONNSTATUS
  Public  Rasconnstate    As Int32
  Public  Error           As Int32
  Public  DeviceType()    As Char
  Public  DeviceName()    As Char
End Class
```

```
[C#]
public class RASCONNSTATUS
{
  public  Int32   Rasconnstate;
  public  Int32   Error;
  public  char[]  DeviceType;
  public  char[]  DeviceName;
};
```

パラメータ

Rasconnstate
接続状態を格納します。
RASCS_Connected　　：接続成功
RASCS_Disconnected　：接続失敗

Error
エラー番号を格納します。

DeviceType
モデムタイプを格納します。

DeviceName
モデム名を格納します。

3.2.6　RasDialParams 構造体

RAS 接続の呼び出しパラメータを格納します。

```
[C++]
C++で使用する RASDIALPARAMS 構造体については下記 MSDN の RASDIALPARAMS 構造体を参照してください。

http://msdn.microsoft.com/ja-jp/library/aa377238(v=VS.85).aspx
```

```
[Visual Basic]
Public Class RasDialParams
  Public  EntryName       As String
  Public  UserName        As String
  Public  Password        As String
  Public  Domain          As String
End Class
```

```
[C#]
public class RasDialParams
{
  public  string  EntryName;
  public  string  UserName;
  public  string  Password;
  public  string  Domain;
};
```

パラメータ

[Visual Basic], [C#]

EntryName

ダイアルアップ接続の接続名を格納します。

UserName

ダイアルアップ接続のユーザ名を格納します。

Password

ダイアルアップ接続のパスワードを格納します。

Domain

ダイアルアップ接続のドメイン名を格納します。

3.2.7　wan_SMS_ADDRESS, WanSmsAddress 構造体

ショートメッセージアドレスを格納します。

```
[C++]
struct wan_SMS_ADDRESS {
  WAN_SMS_ADDRESS_TYPE    AddressType;
  BYTE                    szAddress[WAN_MAX_ADDRESS_LENGTH];
}
```

```
[Visual Basic]
Public Class WanSmsAddress
  Public  AddressType As Int32
  Public  Address     As String
End Class
```

```
[C#]
public class WanSmsAddress
{
  public  Int32   AddressType;
  public  string  Address;
};
```

パラメータ

AddressType
SMS で使用するアドレスタイプを格納します。

szAddress[WAN_SMS_MAX_ADDRESS_LENGTH] , Address
SMS のアドレス実体（文字列）を格納します。

```
#define WAN_SMS_MAX_ADDRESS_LENGTH : 256 characters
```

※ 本構造体を使用する場合は、AddressType に WAN_SMSAT_UNKNOWN を指定してください。

3.3 関数一覧

3.3.1 WANSetPowerStatus, WanSetPowerStatus

WANGPRS モジュールの電源状態を設定します。

```
[C++]
DWORD WANSetPowerStatus(
  DWORD    dwPowerStatus
)
```

```
[Visual Basic]
Public Shared Function WanSetPowerStatus(_
  ByVal    PowerStatus As Def.PowerStatus_
) AS Int32
```

```
[C#]
public static Int32 WanSetPowerStatus(
  Def.PowerStatus PowerStatus
)
```

解説

本関数は、WANGPRS モジュールの電源状態を設定します。
モジュールの電源操作は、1～3 秒程度の時間がかかる場合があります。モジュールのセットアップ/シャットダウン処理が完了するまで、本関数はブロックされます。

パラメータ

dwPowerStatus
WANGPRS モジュールの電源状態を設定します。
WAN_MODULE_POWER_ON　　：電源 ON
WAN_MODULE_POWER_OFF　　：電源 OFF

戻り値

以下の値を返します。
WAN_ERROR_SUCCESS　　：正常終了
WAN_ERROR_FAIL　　：異常終了

3.3.2 WANGetPowerStatus, WanGetPowerStatus

WAN モジュールの電源状態を取得します。

```
[C++]
DWORD WANGetPowerStatus(
  DWORD   *pPowerStatus
)
```

```
[Visual Basic]
Public Shared Function WanGetPowerStatus(_
  ByRef   PowerStatus  As Def.PowerStatus_
)AS Int32
```

```
[C#]
public static Int32 WanGetPowerStatus(
  out Def.PowerStatus PowerStatus
)
```

解説

本関数は、WAN モジュールの電源状態を取得します。

パラメータ

pPowerStatus
WAN モジュールの電源状態を取得します。取得する値は、WANSetPowerStatus関数を参照してください。

戻り値

以下の値を返します。

WAN_ERROR_SUCCESS	: 正常終了
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.3 WANSIMInitialize, WanSimInitialize

SIM 機能の操作を行うためのハンドルを取得します。

```
[C++]
DWORD WANSIMInitialize(
  HANDLE  *phSIM
)
```

```
[Visual Basic]
Public Shared Function WanSimInitialize(_
  ByRef   SIMHandle  As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanSimInitialize(
  out IntPtr  SIMHandle
);
```

パラメータ

本関数は、SIM 機能の操作を行うためのハンドルを取得します。
SIM 機能の操作を行う場合は、必ず本関数を最初に実行してください。

パラメータ

phSIM
SIM 機能の操作を行うためのハンドルを取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCESS	:	正常終了
WAN_ERROR_BADPARAM	:	パラメータエラー
WAN_ERROR_FAIL	:	その他のエラー DeviceEmulator では発生しません
WAN_ERROR_BUSY	:	他プロセスでハンドルオープン済エラー

3.3.4 WANSIMDeInitialize, WanSimDeInitalize

取得したハンドルを解放します。

```
[C++]
DWORD WANSIMDeInitialize(
  HANDLE  hSIM
)
```

```
[Visual Basic]
Public Shared Function WanSimDeInitialize(_
  ByVal   SIMHandle  As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanSimDeInitialize(
  IntPtr  SIMHandle
)
```

パラメータ

本関数は、取得したハンドルを解放します。
SIM 操作が終了した場合は、必ず本関数を実行してください。

パラメータ

hSIM
取得したハンドルを指定します。

戻り値

WAN_ERROR_SUCCESS	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.5 WANSIMGetPhoneLockedState, WanSimGetPhoneLockedState

SIM ロック状態を取得します。

```
[C++]
DWORD WANSIMGetPhoneLockedState(
  HANDLE      hSIM,
  DWORD*      pdwLockState
)
```

```
[Visual Basic]
Public Shared Function WanSimGetPhoneLockedState(_
  ByVal       SIMHandle   As IntPtr,_
  ByRef       LockState   As Def.SimLockStatus _
) AS Int32
```

```
[C#]
public static Int32 WanSimGetPhoneLockedState(
  IntPtr                  SIMHandle,
  out Def.SimLockStatus   LockState
);
```

パラメータ

本関数は、SIM ロック状態を取得します。
Device Emulator では、SIM.ini ファイルで指定した SIM ロック状態を取得します。詳細は、SIM.iniを参照してください。

パラメータ

hSIM
取得したハンドルを指定します。

pdwLockState
SIM ロック状態を取得します。

WAN_SIM_LOCKSTATE_READY	: SIM ロック解除状態
WAN_SIM_LOCKSTATE_PIN	: PIN 入力待機状態
WAN_SIM_LOCKSTATE_PUK	: PUK 入力待機状態
WAN_SIM_LOCKSTATE_OTHER_LOCK	: その他のロック状態

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.6 WANSIMGetPINCounter, WanSimGetPINCounter

現在の状態の PIN カウンタを取得します。

```
[C++]
DWORD WANSIMGetPINCounter(
  HANDLE  hSIM,
  DWORD*  pdwCount
)
```

```
[Visual Basic]
Public Shared Function WanSimGetPINCounter(_
  ByVal       SIMHandle   As IntPtr, _
  ByRef       Count       As Int32 _
) AS Int32
```

```
[C#]
public static Int32 WanSimGetPINCounter(
  IntPtr      SIMHandle,
  out Int32       Count
);
```

解説

本関数は、現在の状態の PIN カウンタを取得します。
PIN カウンタは現在の状態での PIN 入力が行える回数を示しています。PIN カウンタが 0 になった場合は、PIN ロック状態に移行します。
Device Emulator では、SIM.ini ファイルで指定した PIN カウンタを取得します。詳細は、SIM.ini を参照してください。

パラメータ

hSIM
取得したハンドルを指定します。

pdwCount
PIN カウンタを取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー DeviceEmulator では発生しません
WAN_ERROR_FAIL	: その他のエラー

3.3.7 WANSIMUnLockPhone, WanSimUnlockPhone

SIM ロックを解除します。

```
[C++]
DWORD WANSIMUnLockPhone(
  HANDLE  hSIM,
  BYTE*   szPassword1,
  BYTE*   szPassword2
)
```

```
[Visual Basic]
Public Shared Function WanSimUnLockPhone(_
  ByVal   SIMHandle   As IntPtr, _
  ByVal   Password1   As String, _
  ByVal   Password2   As String _
) AS Int32
```

```
[C#]
public static Int32 WanSimUnLockPhone(
  IntPtr  SIMHandle,
  string  Password1,
  string  Password2
)
```

解説

本関数は、SIM ロックを解除します。
本関数は SIM のロック状態により、引数が異なります。そのため、本関数を実行する以前に、WANSIMGetPhoneLockedState関数を実行し、SIM のロック状態を確認してください。
SIM ロック状態が PIN 入力待機の場合は、SIM ロックを解除します。
また、SIM ロック状態が PUK 入力待機の場合は、PUK 入力待機状態を解除し、PIN コードを変更します。
Device Emulator では、SIM.ini ファイルで指定した PIN コードまたは PUK コードを使用します。詳細は、SIM.iniを参照してください。
Device Emulator では、PUK 時の PIN コード置き換えをサポートしません。

パラメータ

hSIM
取得したハンドルを指定します。

szPassword1
SIM ロック状態が PIN 入力待機の場合は、PIN コードを指定します。
SIM ロック状態が PUK 入力待機の場合は、PUK コードを指定します。

szPassword2
SIM ロック状態が PIN 入力待機の場合は、本パラメータを無視します。
SIM ロック状態が PUK 入力待機の場合は、新しい PIN コードを指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	： 正常終了
WAN_ERROR_ILLIGAL_HANDLE	： 不正なハンドル
WAN_ERROR_NOSIM	： SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	： モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	： パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	： 操作未許可エラー
WAN_ERROR_INCORRECT_PASSWORD	： 不正なパスワード
WAN_ERROR_FAIL	： その他のエラー

注意

szPassword1 と szPassword2 は SIM ロック状態に依存して、入力する引数が異なります。

■ PIN コード入力待機状態の場合

SIM ロック状態を解除します。

szPassword1 ： PIN コード
szPassword2 ： 無視

■ PUK コード入力待機状態の場合

PUK 状態を解除し、PIN コードを新しい PIN コードに置き換えます。

szPassword1 ： PUK コード
szPassword2 ： 新しい PIN コード

3.3.8 WANSIMChangeLockingPassword, WanSimChangeLockingPassword

指定したファシリティのパスワードを変更します。

```
[C++]
DWORD WANSIMChangeLockingPassword(
  HANDLE  hSIM,
  DWORD   dwFacility,
  BYTE*   szOldPassword,
  BYTE*   szNewPassword
)
```

```
[Visual Basic]
Public Shared Function WanSimChangeLockingPassword(_
  ByVal   SIMHandle       As IntPtr, _
  ByVal   OldPassword     As String, _
  ByVal   NewPassword     As String _
) AS Int32
```

```
[C#]
public static Int32 WanSimChangeLockingPassword(
  IntPtr  SIMHandle,
  string  OldPassword,
  string  NewPassword
)
```

解説

本関数は、指定したファシリティのパスワードを変更します。
本関数実行前に、必ず WANSIMGetLockingStatus関数を実行し、SIM モジュールがロックが有効になっていることを確認してください。ロックが無効になっている場合、本関数は異常終了します。
Device Emulator では、何もせずに、常に WAN_ERROR_SUCCESS を返します。

パラメータ

hSIM
取得したハンドルを指定します。

dwFacility
ファシリティを指定します。
WAN_SIM_LOCK_FACILITY_PIN　　: PIN コード

szPassword1
既存のパスワードを指定します。

szPassword2
新しいパスワードを指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_INCORRECT_PASSWORD	: 不正なパスワード
WAN_ERROR_FAIL	: その他のエラー

3.3.9 WANSIMSetLockingStatus, WanSimSetLockingStatus

指定したファシリティのロック状態を変更します。

```
[C++]
DWORD WANSIMSetLockingStatus (
  HANDLE  hSIM,
  DWORD   dwFacility,
  BYTE*   szPassword,
  BOOL    bEnable
)
```

```
[Visual Basic]
Public Shared Function WanSimSetLockingStatus (_
  ByVal   SIMHandle   As IntPtr, _
  ByVal   Password    As String, _
  ByVal   Enable      As Boolean _
) AS Int32
```

```
[C#]
public static Int32 WanSimSetLockingStatus (
  IntPtr  SIMHandle,
  string  Password,
  bool    Enable
);
```

解説

本関数は、指定したファシリティのロック状態を変更します。
設定は、モジュールの次回起動時から有効となります。
Device Emulator では、SIM.ini ファイルで指定した PIN コードを使用します。詳細は、SIM.ini を参照してください。
Device Emulator では情報を保存しません。データ変更の有効期間は電源 ON 状態中のみです。

パラメータ

hSIM
取得したハンドルを指定します。

dwFacility
ファシリティを指定します。
WAN_SIM_LOCK_FACILITY_PIN　　: PIN コード

szPassword
ファシリティのパスワードを指定します。

bEnable
ファシリティのロックの有効/無効を指定します。
TRUE　　: 有効
FALSE　　: 無効

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_INCORRECT_PASSWORD	: 不正なパスワード
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.10 WANSIMGetLockingStatus, WanSimGetLockingStatus

指定したファシリティのロック状態を取得します。

```
[C++]
DWORD WANSIMGetLockingStatus (
  HANDLE  hSIM,
  DWORD   dwFacility,
  BYTE*   szPassword,
  BOOL*   pbEnable
)
```

```
[Visual Basic]
Public Shared Function WanSimGetLockingStatus (_
  ByVal   SIMHandle   As IntPtr, _
  ByVal   Password    As String, _
  ByRef   Enable      As Boolean _
) AS Int32
```

```
[C#]
public static Int32 WanSimGetLockingStatus (
  IntPtr      SIMHandle,
  string      Password,
  out bool    Enable
);
```

解説

本関数は、指定したファシリティのロック状態を取得します。
Device Emulator では、SIM.ini ファイルで指定した PIN コードを使用します。詳細は、SIM.ini を参照してください。

パラメータ

hSIM
取得したハンドルを指定します。

dwFacility
ファシリティを指定します。
WAN_SIM_LOCK_FACILITY_PIN　　: PIN コード

szPassword
ファシリティのパスワードを指定します。

bEnable
ファシリティのロック状態を取得します。取得する値は、WANSIMSetLockingStatus関数を参照してください。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.11 WANGetModuleInfo, WanGetModuleInfo

モジュール情報を取得します。

```
[C++]
DWORD WANGetModuleInfo(
  struct wan_ModuleInfo *pModuleInfo
);
```

```
[Visual Basic]
Public Shared Function WanGetModuleInfo(_
  ByRef pModuleInfo As WanModuleInfo _
) AS Int32
```

```
[C#]
public static Int32 WanGetModuleInfo(
  out WanModuleInfo pModuleInfo
)
```

解説

本関数は、モジュール情報を取得します。
Device Emulator では、Module.ini ファイルで指定した値を取得します。詳細は、Module.iniを参照してください。

パラメータ

pModuleInfo
wan_ModuleInfo, WanModuleInfo 構造体を指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.12 WANGetIMEI, WanGetImei

電話機の固有番号 IMEI(International Mobile Equipment Identity)を取得します。

```
[C++]
DWORD WANGetIMEI(
  BYTE*   szIMEI,
  DWORD*  pdwLength
);
```

```
[Visual Basic]
Public Shared Function WanGetImei(_
  ByRef IMEI As String _
) AS Int32
```

```
[C#]
public static Int32 WanGetImei(
  out string IMEI
);
```

解説

本関数は、電話機の固有番号 IMEI(International Mobile Equipment Identity)を取得します。
Device Emulator では、常に"1234567890"を返します。

パラメータ

szIMEI
電話機の固有番号 IMEI(International Mobile Equipment Identity)を取得します。
NULL を指定すると、WAN_ERROR_BUFFER_ERROR が返ります。

pdwLength
[in]
szIMEI のバッファの長さを指定します。
指定したバッファの長さが必要とするバッファの長さを下回った場合は、
WAN_ERROR_BUFFER_ERROR が返ります。
[out]
取得した IMEI の文字列長+1 を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_BUFFER_ERROR	: バッファエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.13 WANGetIMSI, WanGetImsi

加入者識別子 IMSI(International Mobile Subscriber Identity)を取得します。

```
[C++]
DWORD WANGetIMSI(
  BYTE*   szIMSI,
  DWORD*  pdwLength
);
```

```
[Visual Basic]
Public Shared Function WanGetImsi(_
  ByRef IMSI As String _
) AS Int32
```

```
[C#]
public static Int32 WanGetImsi(
  out string IMSI
)
```

解説

本関数は、加入者識別子 IMSI(International Mobile Subscriber Identity)を取得します。
IMSI は、SIM カードに記録してあります。取得するには、以下の条件が必要です。
- SIM カードを挿入済
- PIN ロックを解除済

Device Emulator では、SIM.ini ファイルで指定した IMSI を取得します。詳細は、SIM.iniを参照してください。

パラメータ

szIMSI
加入者識別子 IMSI(International Mobile Subscriber Identity)を取得します。
NULL を指定すると、WAN_ERROR_BUFFER_ERROR が返ります。

pdwLength
[in]
szIMSI のバッファの長さを指定します。
指定したバッファの長さが必要とするバッファの長さを下回った場合は、
WAN_ERROR_BUFFER_ERROR が返ります。
[out]
取得した IMSI の文字列長+1 を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー

WAN_ERROR_BUFFER_ERROR	: バッファエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.14 WANGetTelNum, WanGetTelephoneNumber

電話番号 MSISDN (Mobile Subscriber ISDN Number)を取得します。

```
[C++]
DWORD WANGetTelNum(
  BYTE*   szTelNum,
  DWORD*  pdwLength
);
```

```
[Visual Basic]
Public Shared Function WanGetTelNum(_
  ByRef TelNum As String _
) AS Int32
```

```
[C#]
public static Int32 WanGetTelNum(
  out string TelNum
);
```

解説

本関数は、電話番号 MSISDN(Mobile Subscriber ISDN Number)を取得します。
MSISDN は、SIM カードに記録してあります。取得するには、以下の条件が必要です。
- SIM カードを挿入済
- PIN ロックを解除済

Device Emulator では、SIM.ini ファイルで指定した MSISDN を取得します。詳細は、SIM.iniを参照してください。

パラメータ

szTelNum
加入者識別子 IMSI(International Mobile Subscriber Identity)を取得します。
NULL を指定すると、WAN_ERROR_BUFFER_ERROR が返ります。

pdwLength
[in]
szTelNum のバッファの長さを指定します。
指定したバッファの長さが必要とするバッファの長さを下回った場合は、
WAN_ERROR_BUFFER_ERROR が返ります。
[out]
取得した MSISDN の文字列長+1 を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー

WAN_ERROR_BUFFER_ERROR	: バッファエラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.15 WANGetRSSI, WanGetRssi

受信電波強度 RSSI(Received Signal Strength Indicator)を取得します。

```
[C++]
DWORD WANGetRSSI(
  struct wan_RSSI *pRSSI
);
```

```
[Visual Basic]
Public Shared Function WanGetRssi(_
  ByRef pRSSI As WanRssi _
) AS Int32
```

```
[C#]
static Int32 WanGetRssi(
  out WanRssi pRSSI
);
```

解説

本関数は、受信電波強度 RSSI(Received Signal Strength Indicator)を取得します。
本関数を続けて使用する場合は、必ず 3 秒以上間隔を空けてください。
受信電波を検出できない場合は、wan_RSSI, WanRssi 構造体の nSignalStrength に WAN_SIGNALSTRENGTH_UNKNOWN を格納します。
Device Emulator では、Operator.ini ファイルで指定した受信電波強度を取得します。詳細は、Operator.iniを参照してください。

パラメータ

pRSSI
wan_RSSI, WanRssi 構造体を指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.16 WANGetOperatorList, WanGetOperatorList

オペレータリストを取得します。

```
[C++]
DWORD WANGetOperatorList(
  struct wan_OperatorInfo*    pOperatorList,
  DWORD                       nOperator,
  DWORD*                      pdwNumOperator
)
```

```
[Visual Basic]
Public Shared Function WanGetOperatorList(_
  ByRef   pOperatorList() As WanOperatorInfo, _
  ByVal   Operator        As Int32 _
) AS Int32
```

```
[C#]
public static Int32 WanGetOperatorList(
  out WanOperatorInfo[]   pOperatorList,
  Int32                   Operator
)
```

解説

本関数は、オペレータリストを取得します。
基地局をスキャンし、現在接続可能な基地局のリストを返します。スキャンした基地局が指定した要素数より多い場合は、WAN_ERROR_MORE_OPERATOR を返します。その場合は、指定した要素数まで、基地局情報を格納し、pdwNumOperator にスキャンした基地局数を格納します。
また、本関数はスキャンが終了するまで制御が返りません。
Device Emulator では、OperatorList[n].ini ファイル数だけスキャンを行い、ini ファイルで指定した情報を取得します。詳細は、OperatorList[n].iniを参照してください。

パラメータ

pOperatorList
wan_OperatorInfo, WanOperatorInfo 構造体の配列を指定します。

nOperator
取得する要素数を指定します。

pdwNumOperator
スキャンした要素数を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー

WAN_ERROR_OPERATION_NOT_ALLOW	：操作未許可エラー
WAN_ERROR_MORE_OPERATOR	：基地局要素数エラー
WAN_ERROR_FAIL	：その他のエラー DeviceEmulator では発生しません

3.3.17 WANSetOperatorConnectionMode, WanSetOperatorConnectionMode

オペレータへの接続方法を設定します。

```
[C++]
DWORD WANSetOperatorConnectionMode(
  DWORD                    nMode,
  Struct wan_OperatorInfo *pOperator
)
```

```
[Visual Basic]
Public Shared Function WanSetOperatorConnectionMode(_
  ByVal   Mode        As Def.ConnectionMode, _
  ByVal   pOperator   As WanOperatorInfo_
) AS Int32
```

```
[C#]
public static Int32 WanSetOperatorConnectionMode(
  Def.ConnectionMode  Mode,
  WanOperatorInfo     pOperator
);
```

解説

本関数は、オペレータへの接続方法を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
WANGetOperatorConnectionMode関数を実行することにより、設定値を確認することができます。

パラメータ

nMode
オペレータへの接続方法を指定します。

WAN_CONNECTION_MODE_AUTO	: 自動接続
WAN_CONNECTION_MODE_AUTO_RESCAN	: 自動接続(再探索付)
WAN_CONNECTION_MODE_MANUAL	: 手動接続
WAN_CONNECTION_MODE_ADAPT	: 適応モード

pOperator
wan_OperatorInfo, WanOperatorInfo 構造体を指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備(電源 OFF)
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

補足

自動接続の場合は、SIM カードの情報に基づき、接続を行います。現在、自動接続に設定している場合は、そのまま維持します。（pOperator（Operator）を無視します）
自動接続（再探索付）の場合は、現在の設定にかかわらず、本関数を実行したタイミングで基地局の探索を行います。探索後は、自動接続になります。（pOperator（Operator）を無視します）
手動接続は、指定した基地局に接続します。（pOperator（Operator）の szNumName（NumName）に接続したいオペレータ ID を指定してください）
適応モードは、手動接続試行後に、自動接続に移行します。（pOperator（Operator）の szNumName（NumName）に接続したいオペレータ ID を指定してください）

3.3.18 WANGetOperatorConnectionMode, WanGetOperatorConnectionMode

オペレータへの接続方法を取得します。

```
[C++]
DWORD WANGetOperatorConnectionMode(
  DWORD *pnMode
)
```

```
[Visual Basic]
Public Shared Function WanGetOperatorConnectionMode(_
  ByRef Mode As Def.ConnectionMode _
) AS Int32
```

```
[C#]
public static Int32 WanGetOperatorConnectionMode(
  out Def.ConnectionMode Mode
);
```

解説

本関数は、オペレータへの接続方法を取得します。

パラメータ

pnMode

オペレータへの接続方法を取得します。取得する値は、WANSetOperatorConnectionMode関数を参照してください。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.19 WANGetCurrentOperator, WanGetCurrentOperator

接続中のオペレータの情報を取得します。

```
[C++]
DWORD WANGetCurrentOperator(
  struct wan_OperatorInfo*    pOperatorInfo,
  DWORD*                      pGPRS_Status
);
```

```
[Visual Basic]
Public Shared Function WanGetCurrentOperator(_
  ByRef   pOperatorInfo   As WanOperatorInfo, _
  ByRef   GPRS_Status     As Def.GprsStatus _
) AS Int32
```

```
[C#]
public static Int32 WanGetCurrentOperator(
  out WanOperatorInfo     pOperatorInfo,
  out Def.GprsStatus      GPRS_Status
);
```

解説

本関数は、接続中のオペレータの情報を取得します。
Device Emulator では、パラメータチェックを行い、常に WAN_GPRS_AVAILABLE を取得します。

パラメータ

pOperatorInfo
wan_OperatorInfo, WanOperatorInfo 構造体を指定します。

pGPRS_Status
接続中のオペレータの情報を取得します。

WAN_STATUS_NOT_AVAILABLE	: GPRS/EGPRS 未対応
WAN_GPRS_AVAILABLE	: GPRS 対応
WAN_GPRS_ATTACHED	: GPRS 接続
WAN_EGPRS_AVAILABLE	: EGPRS 対応
WAN_EGPRS_ATTACHED	: EGPRS 接続
WAN_UMTS_AVAILABLE	: UMTS 対応
WAN_UMTS_ATTACHED	: UMTS 接続
WAN_HSDPA_AVAILABLE	: HSDPA 対応
WAN_HSDPA_ATTACHED	: HSDPA 接続
WAN_HSUPA_AVAILABLE	: HSUPA 対応
WAN_HSUPA_ATTACHED	: HSUPA 接続
WAN_HSPA_AVAILABLE	: HSDPA+HSUPA 対応
WAN_HSPA_ATTACHED	: HSDPA+HSUPA 接続
WAN_GSM_AVAILABLE	: GSM 対応
WAN_GSM_ATTACHED	: GSM 接続
WAN_STATUS_UNSUPPORT	: 未対応のネットワーク

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_NO_OPERATOR	: オペレータ非存在 DeviceEmulator では発生しません
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

3.3.20 WANGetCurrentOperatorID, WanGetCurrentOperatorID

接続中のオペレータの ID 情報を取得します。

```
[C++]
DWORD WANGetCurrentOperator(
  struct wan_OperatorInfo*    pOperatorInfo,
  DWORD*                      pGPRS_Status
);
```

```
[Visual Basic]
Public Shared Function WanGetCurrentOperator(_
  ByVal   pOperatorInfo   As WanOperatorInfo, _
  ByRef   GPRS_Status     As Def.GprsStatus _
) AS Int32
```

```
[C#]
public static Int32 WanGetCurrentOperator(
  out WanOperatorInfo     pOperatorInfo,
  out Def.GprsStatus      GPRS_Status
);
```

解説

本関数は、接続中のオペレータの ID 情報のみを取得します。
WANGetCurrentOperator関数に比べて、短時間での実行が可能です。
wan_OperatorInfo, WanOperatorInfo 構造体の szNumName（NumName）のみを取得し、その他のメンバには NULL を格納します。
Device Emulator では、パラメータチェックを行い、常に WAN_GPRS_AVAILABLE を取得します。

パラメータ

pOperatorInfo
wan_OperatorInfo, WanOperatorInfo 構造体を指定します。

pGPRS_Status
接続中のオペレータの情報を取得します。取得する値は、WANGetCurrentOperator関数を参照してください。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	： 正常終了
WAN_ERROR_NOSIM	： SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	： モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	： パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	： 操作未許可エラー
WAN_ERROR_NO_OPERATOR	： オペレータ非存在 DeviceEmulator では発生しません
WAN_ERROR_FAIL	： その他のエラー DeviceEmulator では発生しません

3.3.21 WANSetRASSetting, WanSetRasSetting

GPRS 接続を行うための RAS エントリを設定します。

```
[C++]
DWORD WANSetRASSetting(
  BYTE*    szEntryName,
  BYTE*    szNumber,
  BYTE*    szServiceProvidor
)
```

```
[Visual Basic]
Public Shared Function WanSetRASSetting(_
  ByVal    EntryName        As String, _
  ByVal    Number           As String, _
  ByVal    AccessPointName As String _
) AS Int32
```

```
[C#]
public static Int32 WanSetRASSetting(
  string   EntryName,
  string   Number,
  string   AccessPointName
);
```

解説

本関数は、RAS エントリを設定します。

ConnectionManager 機能を用いて、接続制御を行います。
DeviceConfigurationAPI を用いて、GPRS 通信を行うための接続を設定します。詳細は下記を参照してください。

http://msdn.microsoft.com/en-us/library/ms852998.aspx
http://msdn.microsoft.com/en-us/library/aa455854.aspx

Device Emulator では、パラメータチェックのみを行い、WAN_ERROR_SUCCESS を返します。

パラメータ

szEntryName
RAS 接続エントリ名を指定します。

szNumber
電話番号を指定します。
NULL を指定した場合は、既定の設定"*99***1#"を書き込みます。

szServiceProvidor
GPRS 通信プロバイダ名を指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCESS　　：正常終了
WAN_ERROR_BADPARAM　　：パラメータエラー
WAN_ERROR_FAIL　　：その他のエラー

RAS 設定

EntryName	szEntryName
Modem:	
Phone number	*99***1#
Force Local	Checked
Device Proptaties	
Port Setting	
Baud Rate	115200[bps] mul driver fix-rate 460k[bps]
Data Bits	8[bits]
Parity	None
Stop Bits	1
Flow Control	Hardware
Call Option	
ExtraSetting	+cgdcont=1,ip,szServiceProvider[APN]
TCP/IP Setting	
IP addresss assign	Auto
NameServer assign	Auto
Software compression	Enable
IP Header compression	Enable
Security Setting	
Use Data Encryption	NO
Use EAP	NO

注意

Win32 API の RAS 関数を使用しないでください

WAN 通信は、OS の Connection Manager が接続情報および接続動作の管理を行っているため、Connection Manager の管理下にある RAS 関数（Ras から始まる Win32 API）を、アプリケーションから直接使用することはできません。接続管理は、WANGPRS ライブラリ機能、または、Internet Explorer の起動時に自動起動する Connection Manager 機能をご使用ください。

もし、アプリケーションから直接 RAS 関数を使用した場合、以下のような状態となり、正しく動作しない可能性があります。

- Connection Manager に設定されている情報が、必ずしも常に RAS 側に反映されているとは限らず、想定した動作とは異なる設定で動作してしまう
- Connection Manager が RAS のリソースを使用中に、アプリケーションから RAS 関数を使用した場合、正常動作しない
- アプリケーションが RAS 関数を使用している時には、Connection Manager からのアクセスが失敗してしまい、正しく動作しない

3.3.22 WANSetRASSettingEx, WanSetRasSettingEx

GPRS 接続を行うための RAS エントリを設定します。

```
[C++]
DWORD WANSetRASSettingEx(
  BYTE*    szEntryName,
  BYTE*    szNumber,
  BYTE*    szServiceProvidor,
  BYTE*    szLoginName,
  BYTE*    szPassword,
  DWORD    dwGPRSAuthType
)
```

```
[Visual Basic]
Public Shared Function WanSetRASSettingEx(_
  ByVal    EntryName        As String, _
  ByVal    Number           As String, _
  ByVal    AccessPointName  As String, _
  ByVal    LoginName        As String, _
  ByVal    Password         As String, _
  ByVal    GPRSAuthType     As Int32 _
) AS Int32
```

```
[C#]
public static Int32 WanSetRASSettingEx(
  string   EntryName,
  string   Number,
  string   AccessPointName
  string   LoginName,
  string   Password,
  Int32    GPRSAuthType
);
```

解説

本関数は、RAS エントリを設定します。

ConnectionManager 機能を用いて、接続制御を行います。
DeviceConfigurationAPI を用いて、GPRS 通信を行うための接続を設定します。詳細は下記を参照してください。

http://msdn.microsoft.com/en-us/library/ms852998.aspx
http://msdn.microsoft.com/en-us/library/aa455854.aspx

Device Emulator では、パラメータチェックのみを行い、WAN_ERROR_SUCCESS を返します。

パラメータ

szEntryName
RAS 接続エントリ名を指定します。

szNumber
電話番号を指定します。
NULL を指定した場合は、既定の設定"*99***1#"を書き込みます。

szServiceProvidor
GPRS 通信プロバイダ名を指定します。

szLoginName
GPRS 通信プロバイダへのログイン名を指定します。
NULL を指定した場合は、無指定となります。

szServiceProvidor
GPRS 通信プロバイダへのパスワードを指定します。
NULL を指定した場合は、無指定となります。

dwGPRSAuthType
GPRS 通信プロバイダの認証方式を指定します。

WAN_RAS_AUTH_DEFAULT	: 自動
WAN_RAS_AUTH_PAP	: PAP 方式
WAN_RAS_AUTH_CHAP	: CHAP 方式

戻り値

以下の値を返します。

WAN_ERROR_SUCCESS	: 正常終了
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_FAIL	: その他のエラー

3.3.23 WANRASConnect, WanRasConnect

RAS エントリで GPRS 接続をします。

```
[C++]
DWORD WANRASConnect(
  LPDWORD              pdwResult,
  LPRASDIALPARAMS      prasDialParam,
  DWORD                dwNotifierType,
  LPVOID               lpvNotifier,
  LPHRASCONN           pRasConn
)
```

```
[Visual Basic]
Public Shared Function WanRasConnect(_
  ByRef   Result          As Int32, _
  ByVal   RasDialParam    As RasDialParams, _
  ByVal   NotifierType    As Def.RasDialNotifierType, _
  ByVal   Notifier        As IntPtr , _
  ByRef   RasConn         As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanRasConnect(
  out Int32               Result,
  RasDialParams           RasDialParam,
  Def.RasDialNotifierType NotifierType,
  IntPtr                  Notifier,
  out IntPtr              RasConn
)
```

解説

本関数は、RAS エントリで GPRS 接続をします。

ConnectionManagerAPI を用いて、GPRS 接続を行います。正常終了した場合は、HRASCONN のハンドルを返します。
ConnectionManager が RAS の管理を行っているため、RAS の通信状態を変更する関数を使用しないでください。

Device Emulator では、パラメータチェックのみを行い、何も動作しません。

パラメータ

pdwResult
本関数が正常終了時に 0 を、異常終了時に 0 以外の値を返します。

prasDialParam
補足を参照してください。

dwNotifierType
本パラメータは無視します。

lpvNotifier
NULL を指定してください。

pRasConn
RAS 接続ハンドルを取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_NETWORK_NOT_AVAILABLE	: ネットワーク利用不可 DeviceEmulator では発生しません
WAN_ERROR_FAIL	: その他のエラー DeviceEmulator では発生しません

補足

RasDialParams 構造体の各メンバを以下のように設定してください。

dwSize
構造体のサイズを指定してください。

szEntryName
WANSetRASSetting関数で指定したエントリ名を指定してください。

szPhoneNumber
本パラメータは無視します。

szCallbackNumber
本パラメータは無視します。

szUserName
接続に認証が必要な場合はユーザ名を指定してください。

szPassword
接続に認証が必要な場合はパスワードを指定してください。

szDomain
本パラメータは無視します。

3.3.24 WANRASDisConnect, WanRasDisconnect

GPRS 接続を切断します。

```
[C++]
DWORD WANRASDisConnect(
  LPDWORD      pdwResult,
  HRASCONN     hCon
)
```

```
[Visual Basic]
Public Shared Function WanRasDisConnect(_
  ByRef        Result      As Int32, _
  ByVal        RASConn     As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanRasDisConnect(
  out Int32    Result,
  IntPtr       RASConn
);
```

解説

本関数は、RashangUp 関数を使用して、GPRS 接続を切断します。
詳細は下記 URL を参照してください。
http://msdn.microsoft.com/en-us/library/aa377567(VS.85).aspx
Device Emulator では、パラメータチェックのみを行い、何も動作しません。

パラメータ

pdwResult
RashangUp 関数の戻り値を取得します。

hCon
RAS 接続ハンドルを指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	:	正常終了
WAN_ERROR_BADPARAM	:	パラメータエラー
WAN_ERROR_FAIL	:	その他のエラー DeviceEmulator では発生しません

3.3.25 WANRASGetConnectStatus, WanRasGetConnectStatus

GPRS 接続セッションのステータスを取得します。

```
[C++]
DWORD WANRASGetConnectStatus(
  HRASCONN                    hCon,
  struct wan_CONNECTSTATUS*   pconnstatus
)
```

```
[Visual Basic]
Public Shared Function WanRasGetConnectStatus(_
  ByRef   pconnstatus As RasConnStatusEx, _
  ByVal   hCon        As IntPtr
) AS Int32
```

```
[C#]
public static Int32 WanRasGetConnectStatus(
  out RasConnStatusEx pconnstatus,
  IntPtr              hCon
);
```

解説

本関数は、RasGetConnectStatus 関数を使用して、GPRS 接続セッションのステータスを取得します。
詳細は下記 URL を参照してください。

http://msdn.microsoft.com/en-us/library/aa920162.aspx

Device Emulator では、パラメータチェックを行ない正しければ WAN_ERROR_SUCCESS を返します。
ステータスは固定値を返します。

パラメータ

hCon
RAS 接続ハンドルを指定します。

pconnstatus
接続ステータスを取得します。取得する値は、WANGetCurrentOperator関数を参照してください。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	： 正常終了
WAN_ERROR_BADPARAM	： パラメータエラー
WAN_ERROR_FAIL	： その他のエラー DeviceEmulator では発生しません

3.3.26 WANSmsOpen, WanSmsOpen

SMS 操作を行うために初期化を行います。

```
[C++]
DWORD WANSmsOpen(
  HANDLE*      hSMS,
  DWORD        dwFlag,
  HANDLE*      phMessageAvailableEvent
);
```

```
[Visual Basic]
Public Shared Function WanSmsOpen(_
  ByRef        SMSHandle              As IntPtr, _
  ByVal        Flag                   As Def.SmsModes, _
  ByRef        MessageAvailableEvent  As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanSmsOpen(
  out IntPtr       SMSHandle,
  Def.SmsModes     Flag,
  out IntPtr       MessageAvailableEvent
);
```

解説

本関数は、SMS 操作を行うために初期化を行います。
SMS 操作を行う場合は、必ず最初に本関数を実行してください。
Device Emulator では、パラメータチェック後、WAN_ERROR_SUCCESS を返します。

パラメータ

hSMS
SMS ハンドルを取得します。

dwFLAG
SMS をオープンするモードを指定します。OR 指定で複数指定可能です。
WAN_SMS_MODE_RECEIVE　：受信操作
WAN_SMS_MODE_SEND　：送信操作

phMessageAvailableEvent
メッセージ受信時（WAN_SMS_MODE_RECEIVE 指定時）に、シグナル状態になるイベントを取得します。
イベントがシグナル時に、メッセージを受信しています。イベントはイベント状態の調査のみ使用できます。イベントの変更および開放を行わないでください。

戻り値

以下の値を返します。
WAN_ERROR_SUCCES　：正常終了

WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_BUSY	: 他プロセスでハンドルオープン済エラー
WAN_ERROR_FAIL	: その他のエラー

3.3.27 WANSmsClose, WanSmsClose

SMS ハンドルを開放します。

```
[C++]
DWORD WANSmsClose(
  HANDLE  hSMS
);
```

```
[Visual Basic]
Public Shared Function WanSmsClose(_
  ByVal   SMSHandle   As IntPtr _
) AS Int32
```

```
[C#]
public static Int32 WanSmsClose(
  IntPtr  SMSHandle
);
```

解説

本関数は、SMS ハンドルを開放します。
SMS 操作を終了する場合は、必ず本関数を実行してください。
Device Emulator では、何もせずに、WAN_ERROR_SUCCESS を返します。

パラメータ

hSMS
SMS ハンドルを指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCESS	: 正常終了
WAN_ERROR_ILLIGAL_HANDLE	: 不正なハンドル
WAN_ERROR_FAIL	: その他のエラー

3.3.28 WANSmsSetSMSC, WanSmsSetSmsCenterAddress

SMS センタのアドレスを設定します。

```
[C++]
DWORD WANSmsSetSMSC(
  HANDLE              hSMS,
  WAN_SMS_ADDRESS*    pSMSC
);
```

```
[Visual Basic]
Public Shared Function WanSmsSetSmsCenterAddress (_
  ByVal   SMSHandle   As IntPtr, _
  ByVal   pSMSC       As WanSmsAddress _
) AS Int32
```

```
[C#]
public static Int32 WanSmsSetSmsCenterAddress (
  IntPtr              SMSHandle,
  WanSmsAddress       pSMSC
);
```

解説

本関数は、SMS センタのアドレスを設定します。
Device Emulator では、何もせずに、WAN_ERROR_SUCCESS を返します。

パラメータ

pSMSC
SMS センタを指定します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_ILLEGAL_HANDLE	: 不正なハンドル
WAN_ERROR_FAIL	: その他のエラー

3.3.29 WANSmsGetSMSC, WanSmsGetSmsCenterAddress

SMS センタのアドレスを取得します。

```
[C++]
DWORD WANSmsGetSMSC(
  HANDLE              hSMS,
  WAN_SMS_ADDRESS*    pSMSC
);
```

```
[Visual Basic]
Public Shared Function WanSmsGetSmsCenterAddress (_
  ByVal   SMSHandle   As IntPtr, _
  ByRef   pSMSC       As WanSmsAddress _
) AS Int32
```

```
[C#]
public static Int32 WanSmsGetSmsCenterAddress (
  IntPtr              SMSHandle,
  out WanSmsAddress   pSMSC
);
```

解説

本関数は、SMS センタのアドレスを取得します。
Device Emulator では、何もせずに、WAN_ERROR_SUCCESS を返します。

パラメータ

hSMS
SMS ハンドルを指定します。

pSMSC
SMS センタのアドレスを取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_ILLEGAL_HANDLE	: 不正なハンドル
WAN_ERROR_FAIL	: その他のエラー

3.3.30 WANSmsRecvMessage, WanSmsReceiveMessage

SMS メッセージを受信します。

```
[C++]
DWORD WANSmsRecvMessage(
  HANDLE              hSMS,
  WAN_SMS_ADDRESS*    psmsaSMSCAddress,
  WAN_SMS_ADDRESS*    psmsaSourceAddress,
  SYSTEMTIME*         pstReceiveTime,
  BYTE*               pbBuffer,
  DWORD               dwBufferSize,
  BYTE*               pbProviderSpecificBuffer,
  DWORD               dwProviderSpecificDataBuffer,
  DWORD*              pdwBytesRead
);
```

```
[Visual Basic]
Public Shared Function WanSmsReceiveMessage(_
  ByVal hSMS                  As IntPtr, _
  ByRef psmsaSourceAddress    As WanSmsAddress, _
  ByRef pstReceiveTime        As DateTime, _
  ByRef pbBuffer              As String, _
) AS Int32
```

```
[C#]
public static Int32 WanSmsReceiveMessage(
  IntPtr              hSMS,
  out WanSmsAddress   psmsaSourceAddress,
  out DateTime        pstReceiveTime,
  out string          pbBuffer
)
```

解説

本関数は、SMS メッセージを受信します。
受信メッセージが指定したサイズより大きい場合は、メッセージサイズのみを取得し、WAN_ERROR_BUFFER_ERROR を返します。その場合は、メッセージサイズをもとにバッファサイズを変更し、再度受信してください。
Device Emulator では、何もせずに、WAN_ERROR_NOMAIL を返します。

パラメータ

hSMS
SMS ハンドルを指定します。

psmsaSMSCAddress
wan_SMS_ADDRESS, WanSmsAddress 構造体を指定します。本パラメータは無視します。

psmsaSourceAddress
wan_SMS_ADDRESS, WanSmsAddress 構造体を指定します。（送信元 SMS を取得します）

pstReceiveTime
受信時刻を取得します。

pbBuffer
受信メッセージを格納します。

dwBufferSize
受信するサイズを指定します。

pbProviderSpecificBuffer
プロバイダ固有のデータを取得します。本パラメータは無視します。

dwProviderSpecificDataBuffer
プロバイダ固有のデータのサイズを指定します。本パラメータは無視します。

pdwBytesRead
受信バイト数を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備（電源 OFF）
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_ILLEGAL_HANDLE	: 不正なハンドル
WAN_ERROR_NOMAIL	: メール未受信
WAN_ERROR_BUFFER_ERROR	: バッファエラー
WAN_ERROR_FAIL	: その他のエラー

3.3.31 WANSmsSendMessage, WanSmsSendMessage

メッセージを送信します。

```
[C++]
DWORD WANSmsSendMessage()
  HANDLE                  hSMS,
  WAN_SMS_ADDRESS*        pSMSCAddress,
  WAN_SMS_ADDRESS*        pDestinationAddress,
  BYTE                    bValidityPeriod,
  BYTE *                  const pbData,
  DWORD                   dwDataSize,
  BYTE *                  pProviderSpecificData,
  DWORD                   dwProviderSpecificDataSize,
  WAN_SMS_DATA_ENCODING   DataEncoding,
  DWORD                   dwOptions,
  WAN_SMS_MESSAGE_ID*     pMessageID
);
```

```
[Visual Basic]
Public Shared Function WanSmsSendMessage(_
  ByVal hSMS                  As IntPtr, _
  ByVal pDestinationAddress   As WanSmsAddress, _
  ByVal bValidityPeriod       As Byte, _
  ByVal pbData                As String, _
  ByRef pMessageID            As Int32
) AS Int32
```

```
[C#]
public static Int32 WanSmsSendMessage()
  IntPtr          hSMS,
  WanSmsAddress   pDestinationAddress,
  byte            bValidityPeriod,
  string          pbData,
  out Int32       pMessageID
)
```

解説

本関数は、SMS メッセージを送信します。
送信可能な文字は、GSM 03.38 文字セットの main character です。(詳細は補足を参照してください)
Device Emulator では、何もせずに、WAN_ERROR_SUCCESS を返します。

パラメータ

hSMS
SMS ハンドルを指定します。

pSMSCAddress
wan_SMS_ADDRESS, WanSmsAddress 構造体を指定し、経由 SMSC を指定します。

NULL を指定すると、既定の SMSC を使用します。(NULL 推奨)

pDestinationAddress
wan_SMS_ADDRESS, WanSmsAddress 構造体を指定し、送信先アドレスを指定します。

bValidityPeriod
メッセージの有効期間を指定します。デフォルトは 1 日です。

0 to 143	: (TP-VP + 1) x 5 minutes (i.e. 5 minutes intervals up to 12 hours)
144 to 167	: 12 hours + ((TP-VP -143) x 30 minutes)
168 to 196	: (TP-VP - 166) x 1 day
197 to 255	: (TP-VP - 192) x 1 week

pbData
送信するメッセージを指定します。

dwDataSize
送信メッセージのサイズを指定します。

pProviderSpecificData
プロバイダ固有のデータを指定します。本パラメータは無視します。

dwProviderSpecificDataSize
プロバイダ固有のデータのサイズを指定します。本パラメータは無視します。

DataEncoding
文字セットを指定します。

WAN_SMS_ENCODED_GSM : GSM 文字列

dwOptions
予約領域です。

pMessageID
メッセージ ID を取得します。

戻り値

以下の値を返します。

WAN_ERROR_SUCCES	: 正常終了
WAN_ERROR_NOSIM	: SIM 未挿入
WAN_ERROR_MODULE_NOT_READY	: モジュール未準備(電源 OFF)
WAN_ERROR_BADPARAM	: パラメータエラー
WAN_ERROR_OPERATION_NOT_ALLOW	: 操作未許可エラー
WAN_ERROR_ILLEGAL_HANDLE	: 不正なハンドル
WAN_ERROR_FAIL	: その他のエラー

補足

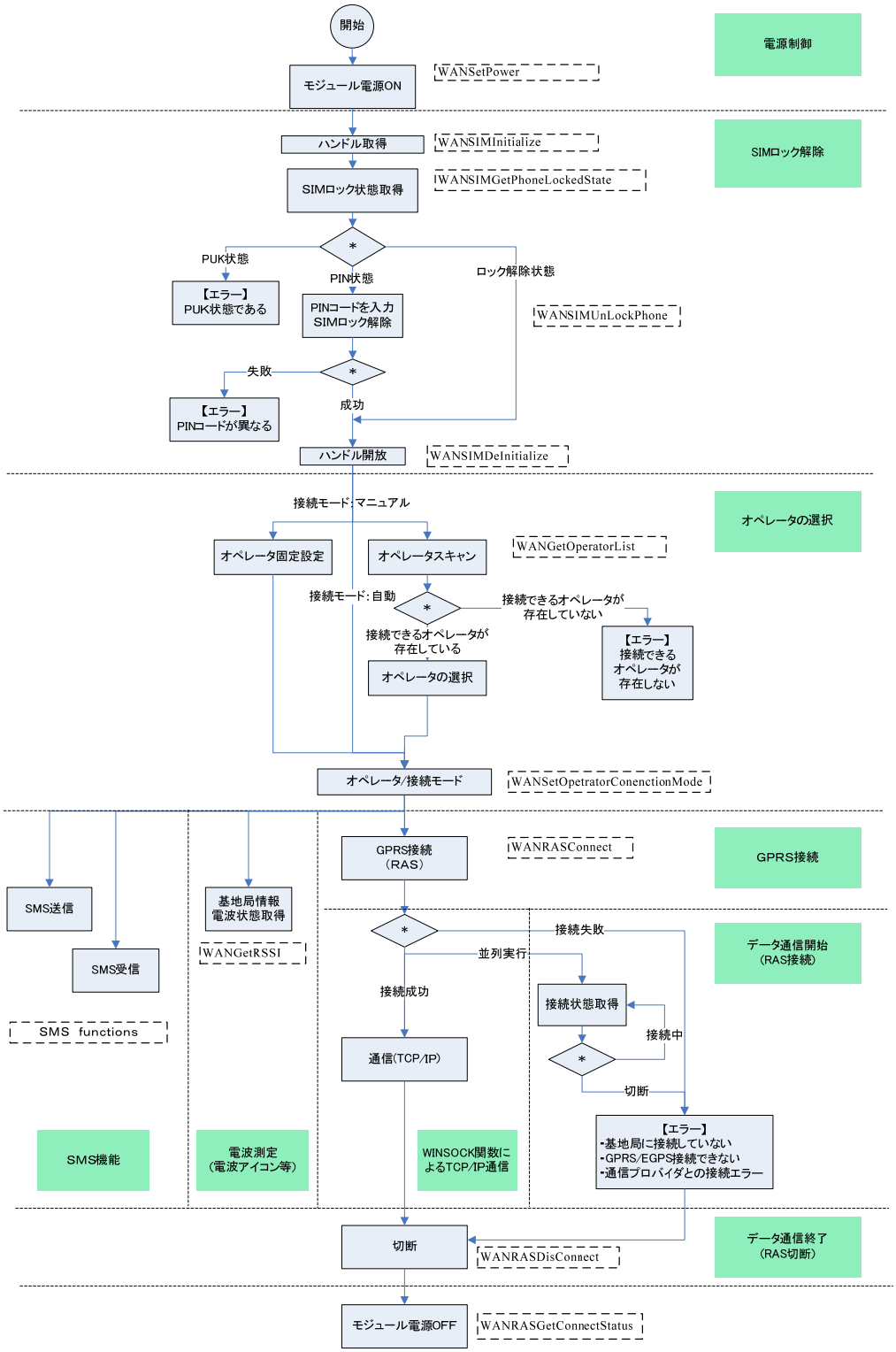
GSM 03.38 文字セット一覧表を以下に示します。
ASCII コードで表示できる文字が使用可能です(網掛け部分は使用できません)。

@ 0040		SP 0020	0 0030		P 0050		p 0070
	_ 005F	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
$ 0024		" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
¥ 00A5		# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
			4 0034	D 0044	T 0054	d 0064	t 0074
		% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
		& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
		' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
		(0028	8 0038	H 0048	X 0058	h 0068	x 0078
		) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF [LF]		* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
		+ 002B	; 003B	K 004B		k 006B	
		, 002C	< 003C	L 004C		l 006C	
CR [CR]		- 002D	= 003D	M 004D		m 006D	
		. 002E	>003E	N 004E		n 006E	
		/ 002F	? 003F	O 004F		o 006F	

4. プログラミング上の注意

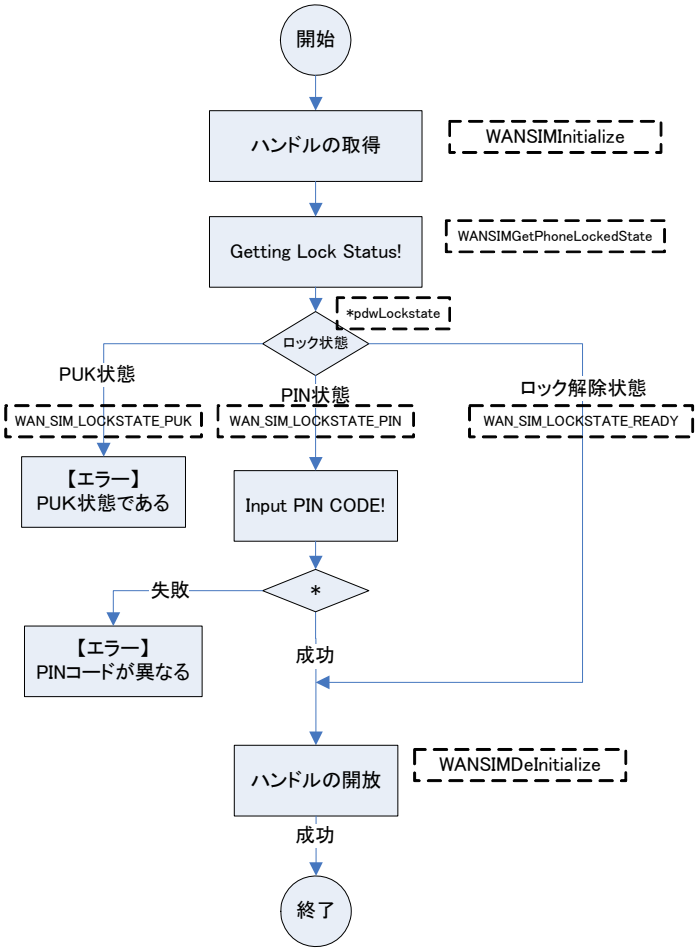
4.1 GPRS 制御手順

GPRS の制御手順は以下の通りです。



4.2 SIM 制御

以下のような手順で SIM セキュリティの解除を行ってください。



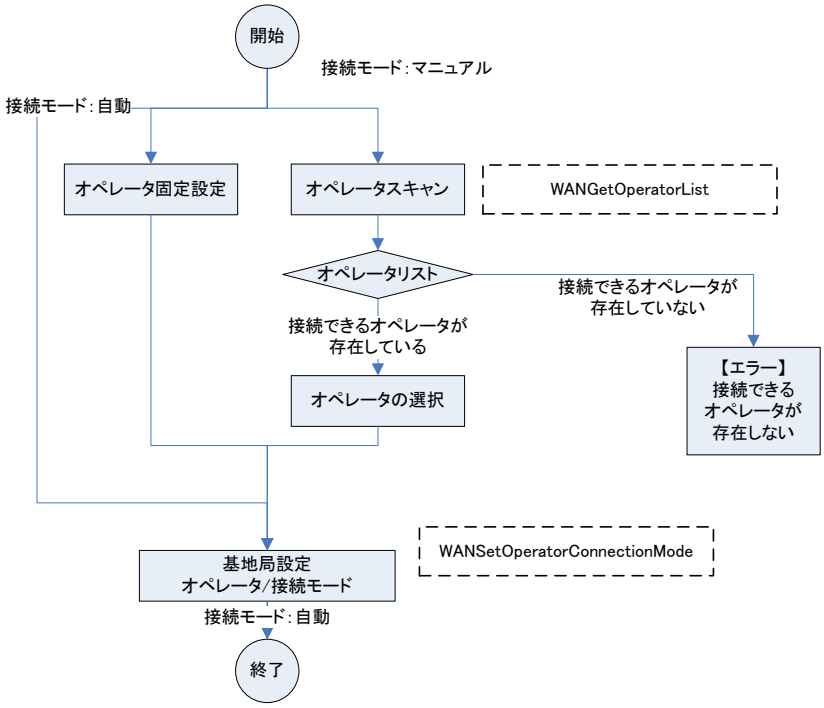
◆ロック状態の取得
ロック状態を必ず取得してください。PIN コードの入力回数には制限があり、試行回数を超えると次の状態に移行します。通常 PIN コードの入力は 3 回まで試行する事ができます※。この回数をオーバーするとPUK コード入力待機状態に移行します。
端末の電源 OFF やリセット、または、GPRS モジュールの電源 OFF 後は、GPRS 使用前に必ずロック状態を取得し、必要な処理を行ってください。

◆PIN コードを入力
ロックを解除するために、PIN コードを入力します。上記のロック状態を取得し、状態に対応した、PIN コードを入力する必要があります。

※ オペレータより供給される SIM カードにより異なります。

4.3 GPRS 制御

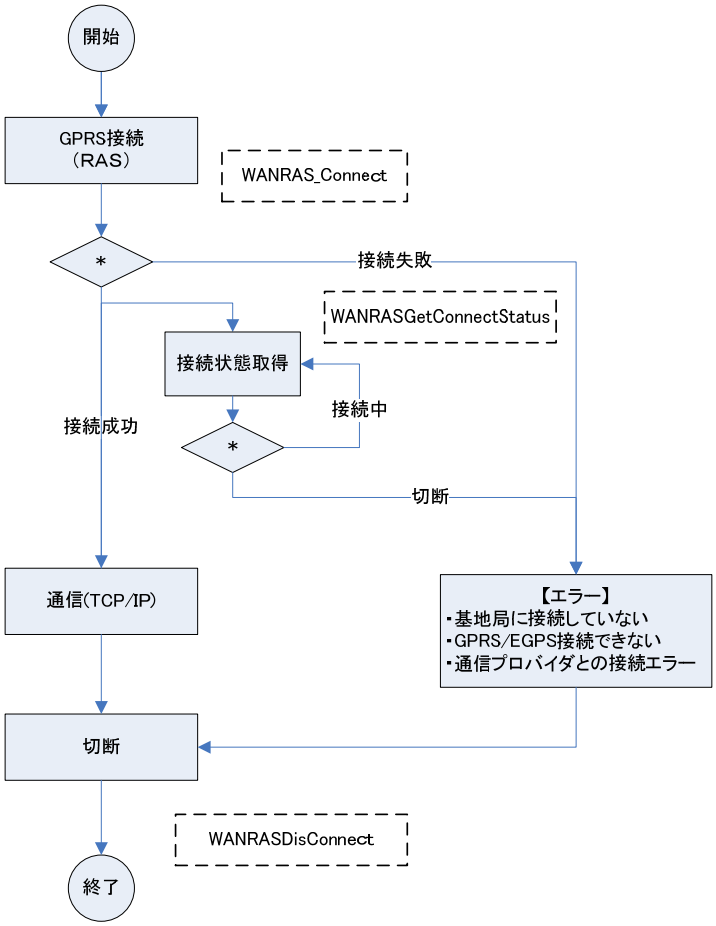
4.3.1 オペレータ選択手順



接続モードを決定します

- 接続モードが自動の場合
  オペレータ名を設定する必要はありません。
  SIM カードに書かれているオペレータに自動接続します。
- 接続モードがマニュアルの場合
  オペレータのスキャンを行います。
  - 存在するオペレータから接続するオペレータを選択します。
  オペレータ固定
  - 固定のオペレータを設定します。
- 接続モードが適応モードの場合
  マニュアル動作を試みます。
  マニュアル動作で接続できない場合は自動モードに移行します。

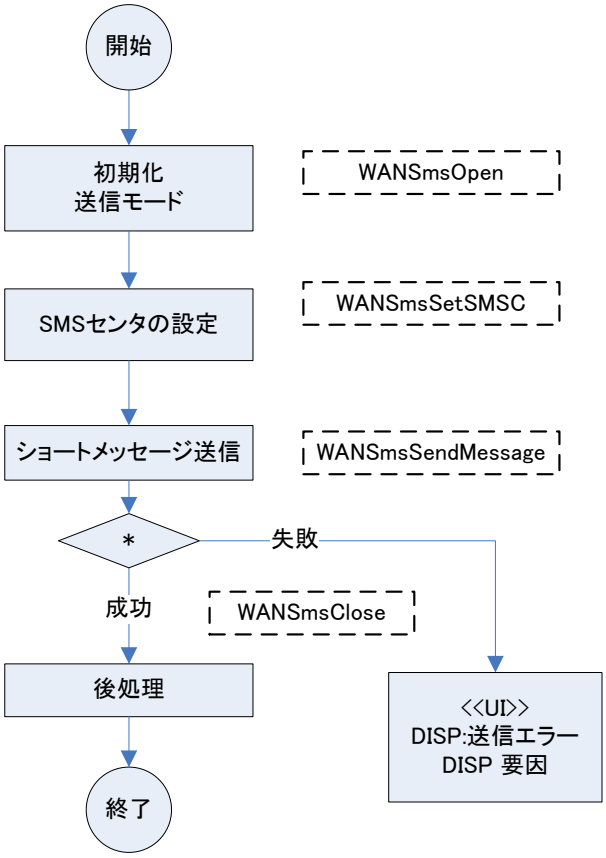
4.3.2　GPRS 接続手順



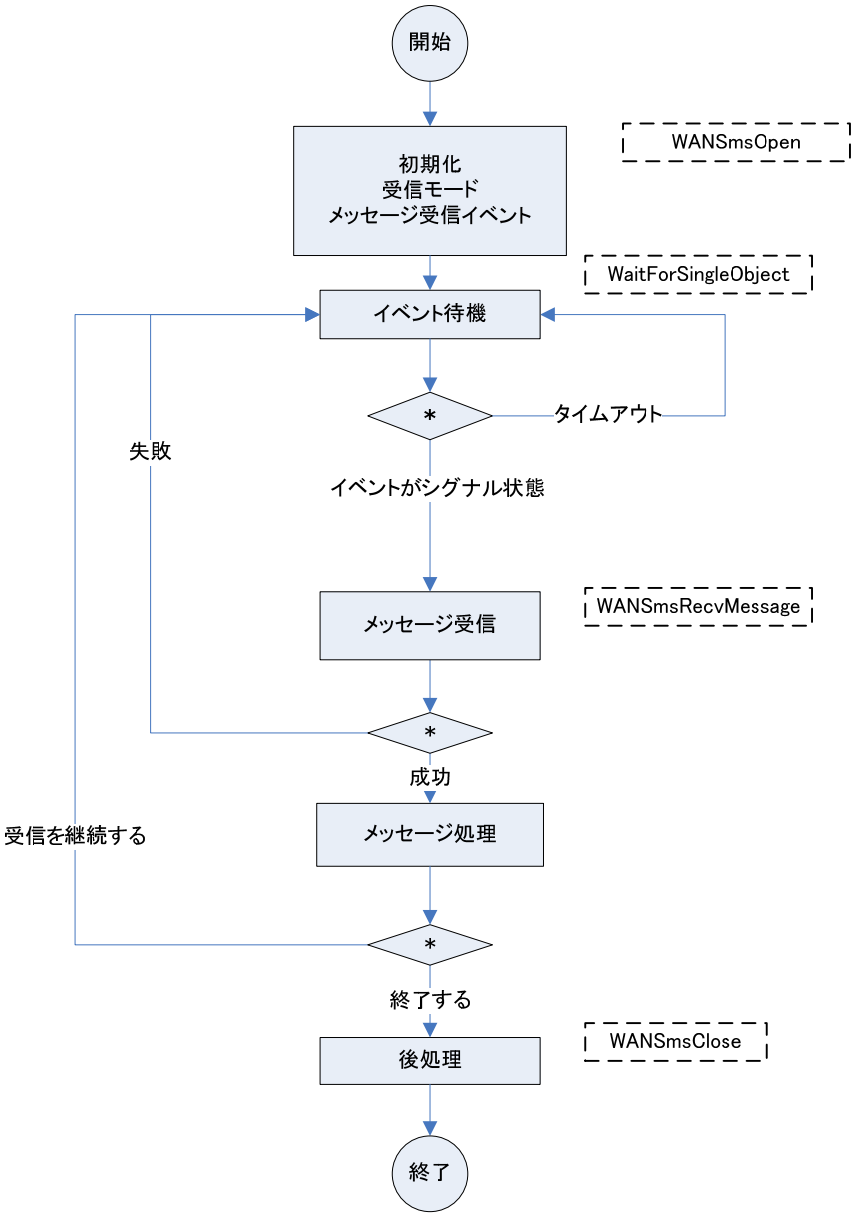
RAS 接続関数にて、GPRS 接続を行います。
RAS エントリを作成する関数 WAN_SetRASSetting を用意しています。
この関数を使用して作成したエントリで RAS 接続を行います。
接続中は、接続ステータスを取得する関数で、通信ステータスを取得してください。ステータスが切断状態になった場合は、切断処理を行ってください。

4.4 SMS 制御

4.4.1 SMS 送信



4.4.2　SMS 受信


開始
初期化
受信モード
メッセージ受信イベント
WANSmsOpen
WaitForSingleObject
イベント待機
*
タイムアウト
失敗
イベントがシグナル状態
メッセージ受信
WANSmsRecvMessage
*
成功
受信を継続する
メッセージ処理
*
終了する
後処理
WANSmsClose
終了

5. DeviceEmulator

Device Emulator 上で本ライブラリを正しく動作させるためには、以下のファイルが必要です。

SIM.ini
Module.ini
Operator.ini
OperatorList[n].ini ([n]:0-9)

ファイルはあらかじめ以下のフォルダにインストールされています。

¥Storage Card¥WAN

5.1 SIM.ini

SIM カード情報を保存するファイルです。
SIM.ini のサンプルを以下に示します。

```
[SIM]
INSERTED=1
LOCKED_STATE=1
LOCKED_ENABLE=1
PIN_CODE="1111"
PUK_CODE="2222"
PIN_COUNTER=3
PIN_COUNTER_MAX=3
PUK_COUNTER_MAX=10
IMSI="0987654321"
TELNUM="0426395155"
```

[SIM]　SIM カード情報セクション	
INSERTED	SIM カードの未挿入/挿入済を設定します。 0 :未挿入 1 :挿入済（既定値）
LOCKED_STATE	SIM ロック状態を設定します。 0 :UNLOCK 1 :PIN state（既定値） 2 :PUK state
LOCKED_ENABLE	SIM ロックの有効/無効を設定します。 0 :無効 1 :有効（既定値）
PIN_CODE	PIN コードを文字列で設定します。（既定値:"123456789"）
PUK_CODE	PUK コードを文字列で設定します。（既定値:"111111111"）
PIN_COUNTER	PIN/PUK コードの入力可能回数を数値で設定します。（既定値:3）
PIN_COUNTER_MAX	PIN コードの入力試行回数の初期値を数値で設定します。 （既定値:3）
PUK_COUNTER_MAX	PUK コードの入力試行回数の初期値を数値で設定します。 （既定値:10）
IMSI	識別番号を文字列で設定します。（既定値:"0987654321"）
TELNUM	電話番号を文字列で設定します。（既定値:"0426395155"）

※ 既定値は、ini ファイルが存在しない場合に適用する値です。

5.2 Module.ini

WANGPRS モジュールの情報を保存するファイルです。
Module.ini のサンプルを以下に示します。

```
[MODULE]
VENDOR="CASIO"
NAME="XT75-EMULATE"
REVISION="0.0.0.0"
```

[MODULE] モジュール情報セクション		
	VENDOR	製造メーカ名を設定します。(既定値:"CASIO??")
	NAME	デバイス名を設定します。(既定値:"XT75-EMULATE")
	REVISION	リビジョンを設定します。(既定値:"0.0.0")

※ 既定値は、ini ファイルが存在しない場合に適用する値です。

5.3 Operator.ini

基地局の受信電波強度を保存するファイルです。
Operator.ini のサンプルを以下に示します。

```
[OPERATOR]
RSSI_MIN=-113
RSSI_MAX=-53
RSSI=-70
```

[MODULE] オペレータ情報セクション		
	RSSI_MIN	受信電波感度の最小値[dBm]を数値で設定します。(既定値:-113)
	RSSI_MAX	受信電波感度の最大値[dBm]を数値で設定します。(既定値:-53)
	RSSI	受信電波感度[dBm]を数値で設定します。(既定値:-70)

※ 既定値は、ini ファイルが存在しない場合に適用する値です。

5.4 OperatorList[n].ini

モジュールがオペレータをスキャンする場合に使用する情報を保存するファイルです。
OperatorList[n].ini のサンプルを以下に示します。

```
[OPERATORINFO]
LONGNAME="o2 - de"
SHORTNAME="o2 - de"
ID="26207"
```

[OPERATORINFO]オペレータ情報セクション		
	LONGNAME	基地局名を 32 文字以内の文字列で設定します。
	SHORTNAME	基地局名を 16 文字以内の文字列で設定します。
	ID	基地局の ID を 16 文字以内の文字列で設定します。

※ 既定値はありません。ini ファイルが存在しない場合は、基地局が存在しない状態となります。

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

http://casio.jp/support/ht/

カシオ計算機株式会社

〒151-8543　東京都渋谷区本町 1-6-2

TEL 03-5334-4638(代)