TCPIPOCX

WILL

株式会社ウィル

白紙ページ

白紙ページ

目 次

白紙ページ

はじめに

白紙ページ

商品に含まれるもの

1．CD-ROM
- Willware.exe
- Cryptdll.exe
  （暗号 DLL 専用・実行環境用セットアップキット）
- readme.txt

2．フロッピーディスク
- レジストリファイル
- readme.txt

3．使用許諾契約書

4．マニュアル

動作環境について

■対応 OS

TCPIPOCX は、以下に示す OS で動作確認を行っております。

Microsoft Windows 95、Microsoft Windows 98、
Microsoft WindowsNT 4.0、Microsoft Windows 2000
Microsoft Windows XP、Microsoft Windows 2003

■開発に必要なソフトウェア

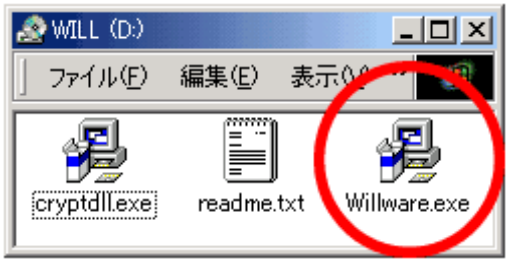
TCPIPOCX をご使用いただくには、以下のいずれかのソフトウェアが必要です。

Microsoft Visual Basic Ver 5.0
Microsoft Visual Basic Ver 6.0
Microsoft Office 2000 (Access、Excel)

TCPIPOCX は、Microsoft Visual C++ で作成しています。サンプルは、Microsoft Visual Basic Ver 5.0 及び Ver 4.0 で作成しています。
※本製品は日本語環境のみの対応となります。

インストール

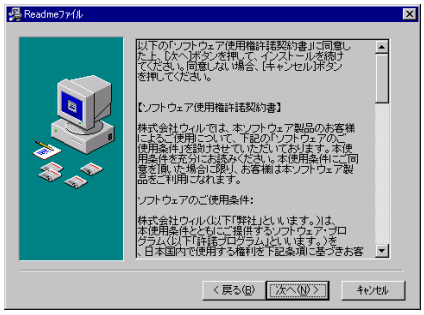
製品の CD-ROM に含まれているセットアップキット(Willware.exe)をダブルクリックします。



画面にしたがって、インストールを進めて下さい。

1. インストールを始めます。「次へ」をクリックして下さい。

2. 使用許諾契約書です。内容に同意される場合は「次へ」をクリックして下さい。



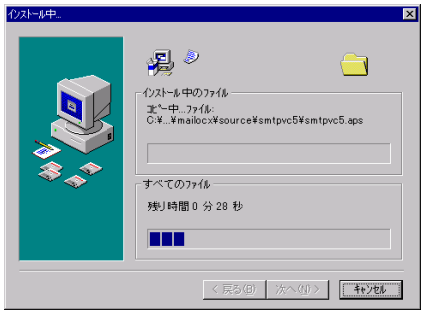
3. インストール先のフォルダを指定します。初期設定でよろしければ「次へ」をクリックして下さい。別のフォルダを指定したい場合は「参照」をクリックし、フォルダを指定して下さい。

4. インストール中に置換されるファイルのバックアップを作成できます。そのバックアップファイルの保存先フォルダを指定します。初期設定でよろしければ「次へ」をクリックして下さい。

5. WILLWARE Components を登録するスタートメニュー又はプログラムマネージャのグループフォルダを指定します。初期設定では、新規に「WILLWARE Components」の名前でフォルダを作成します。特に指定する必要がなければ、初期設定をお勧めします。

6. プログラムのコピーを開始します。「次へ」をクリックして下さい。

7. プログラムのコピーをしています。中断する場合は、「キャンセル」をクリックして下さい。

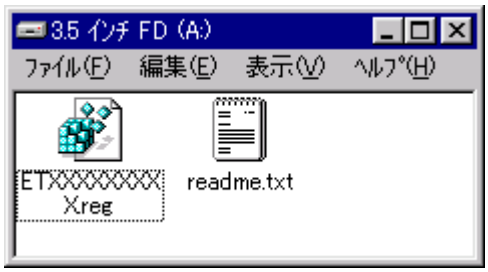


8. インストールが完了しました。「完了」をクリックし、インストールを終了して下さい

ライセンスの登録

■レジストリファイルから登録する

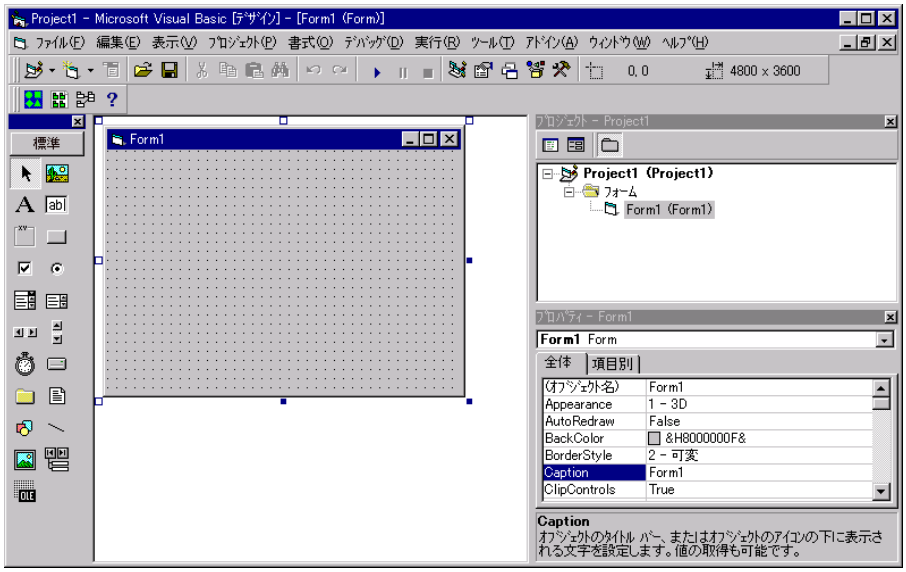
ライセンスを登録します。製品に含まれているフロッピーディスクのレジストリファイル（ETXXXXXXXXX.reg）をダブルクリックして下さい。（「XXXXXXXXX」は、任意の数字がファイル名として付けられています。）



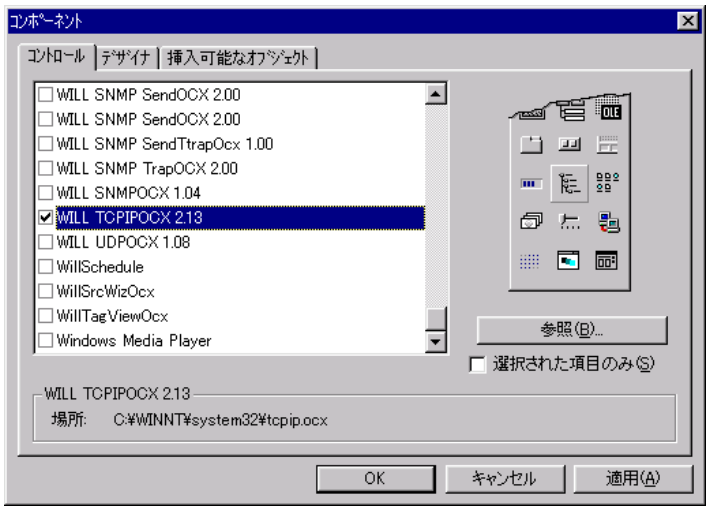
以下のメッセージボックスが表示され、ライセンスがレジストリに登録されます。

■手動で登録する

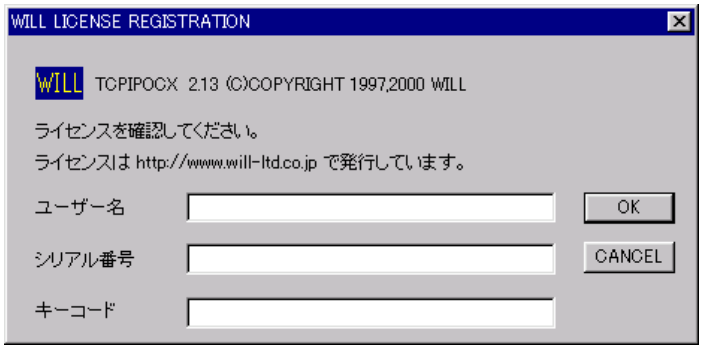
あらかじめ電子メールで通知しているライセンス情報を利用してライセンスを登録する等、レジストリファイルを利用しない場合は、VisualBasic 起動後に新規プロジェクトを選択し以下のデザイン画面を開きます。



ツールバーの「プロジェクト」から、「コンポーネント」を選択し、「コンポーネント」画面を開きます。次にコントロールタブの一覧から TCPIPOCX を選択して「OK」をクリックすると、TCPIPOCX がツールボックスに追加され、アイコンが表示されます。

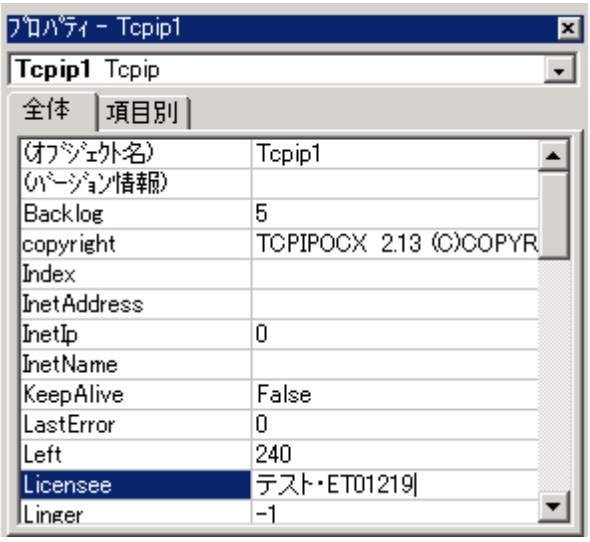


ツールボックスに追加された TCPIPOCX を選択し、フォームにアイコンを貼り付けると、以下の「WILL LICENSE REGISTRATION」画面が表示されます。ここで、ユーザー名、シリアル番号、キーコードをそれぞれ入力してライセンス登録を行います。

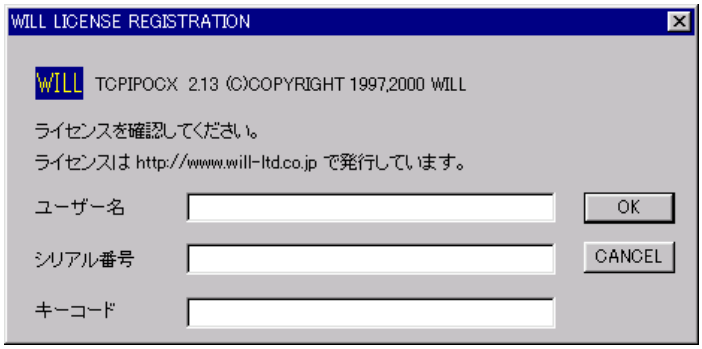


■トライアルライセンスから正規ライセンスへの移行

既にトライアルライセンスが登録されている場合には、デザイン画面にある TCPIPOCX のプロパティで「バージョン情報」をクリックして下さい。



「WILL LICENSE REGISTRATION」画面が表示されますので、ここで正規ライセンスを入力して下さい。



■ライセンス入力時のご注意

※ライセンスが入力できない！？

入力したライセンスにスペースが含まれていないか確認して下さい。(ライセンスに、スペースは使用していません。)

※登録したライセンスを認識しない！？

ライセンスを登録しても、オブジェクトが新規ライセンスを認識していない場合は、TCPIPOCX のアイコンを少し動かして下さい。この作業により、オブジェクトにライセンスが記憶されます。

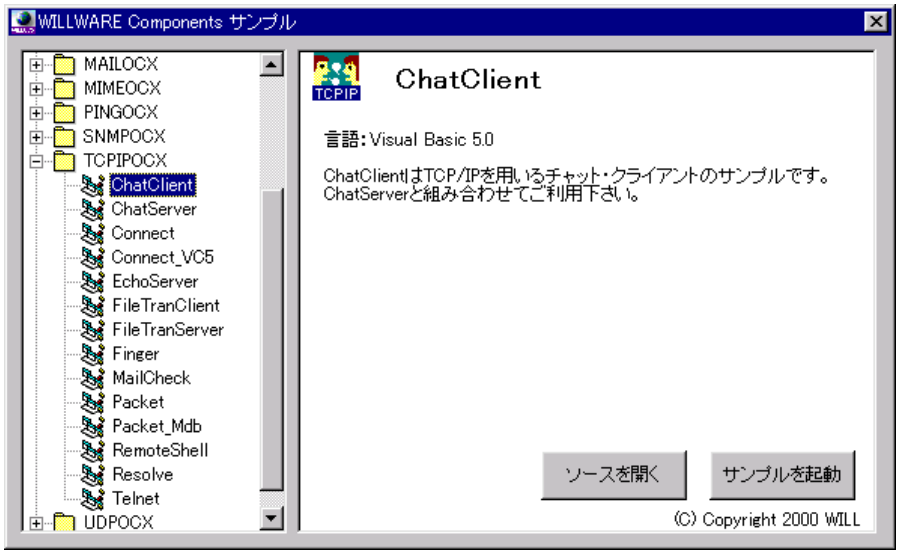
※トライアルライセンスで作成したアプリケーションはどうする！？

既にトライアルライセンスで作成したアプリケーションは、正規ライセンスを登録した後、再コンパイルする必要があります。

サンプルを見る

インストールが完了すると、スタートメニューに「WILLWARE Components」が追加されます。

「WILLWARE Components」の「サンプル」を起動すると「WILLWARE Components サンプル」画面が表示されます。サンプルの起動、またはそれぞれのソースを開くことができます。但し、ソースを開くにはライセンスが必要です。トライアルライセンス又は、正規ライセンスを登録してご利用下さい。(ライセンスの登録方法は前項の「ライセンスの登録」をご覧下さい。)



サポートについて（無償）

サポートは基本的に電子メールで受け付けております。サポートは無償でご利用いただけます。

■お問い合わせの前に

サポート作業を円滑に行うために、お問い合わせの際には以下の情報をご用意下さい。

1．製品名及びバージョン
2．開発環境（OS の種類及びバージョン、サービスパッケージの種類）
3．開発ツール及びバージョン
4．サーバーの種類
5．問題点
   (1) エラー内容又は、エラー状況のハードコピー
   (2) 問題点となる部分のサンプルソースコード

■FAQ

弊社ホームページの「サポート」のページで、キーワードを入力して FAQ を検索できます。休業日などサポートの対応が遅れる場合もありますので、まずはこちらをご確認下さい。

■お問合せ先

info@will-ltd.co.jp

バージョンアップについて（無償）

製品のバージョンアップは、すべて無償です。

■バージョンアップ情報の入手方法

バーションアップの情報は、弊社ホームページの新着情報で通知し、各商品のページの更新履歴で更新内容を掲示致します。

■最新バージョンの入手方法

最新バージョンのプログラムは、弊社ホームページ（http://www.will-ltd.co.jp/）のダウンロードのページよりダウンロードすることが出来ます。ダウンロードするファイルは、以下のバージョンアップの目的により異なりますのでご注意下さい。

● **WILLWARE Components（全製品用）セットアップキットを利用してバージョンアップ**
ファイル名　：「Willware.exe」

WILLWARE Components（全製品用）セットアップキットは全ての製品をインストールするためのものです。そのため本製品以外の製品及びサンプル、マニュアルも同時にバージョンアップされます。

● **各コンポーネント毎のセットアップキットを利用してバージョンアップ**
ファイル名　：「〇〇〇ocx.exe」

各コンポーネントのファイル(ocx、dll)及び、依存ファイルのみバージョンアップされます。サンプル及びマニュアルはバージョンアップされませんのでご注意下さい。

■バージョンアップをする前に

各セットアップキットを利用してバーションアップをする前に、以下のことにご注意下さい。

● **WILLWARE Components（全製品用）セットアップキット**を利用してバージョンアップする場合は、古いバージョンをアンインストールしてから、最新バージョンをインストールすることをお勧めいたします。
※ アンインストールの方法は、スタートメニューから「設定」→「コントロールパネル」→「アプリケーションの追加と削除」の画面で、「WILLWARE Components」を選択し、画面の指示に従って行って下さい。

● **各コンポーネント毎のセットアップキット**を利用してバージョンアップする場合は、最新バージョンをそのままインストールして下さい。古いファイルは上書きされます。
※ 弊社製品を複数ご利用いただいている場合、いずれか１つをバーションアップしても他の製品に影響はありません。

■バージョンアップの方法

セットアップキットをダブルクリックし、画面の指示に従ってインストールを進めて下さい。

再配布について

■作成したアプリケーションの配布時

TCPIPOCX を利用して作成したアプリケーションの配布時のランタイムライセンスはフリーです。但し、開発ライセンスの配布はできません。

■再配布時に必要な配布可能ファイル

TCPIPOCX を利用して作成したアプリケーションを配布する場合には、以下のファイルを添付する必要があります。( )内は推奨バージョンです。

- TCPIPOCX
- MFC42.DLL (Ver 4.21.7022)
- MFC42LOC.DLL (Ver 4.21.7022)
- MSVCRT.DLL (Ver 5.00.7022)
- OLEPRO32.DLL (Ver 5.0.4118)
- OLEAUT32.DLL (Ver 2.20.4118)

※ セットアップウィザードを使用する場合

TCPIPOCX をインストールすると、自動的に OCX の依存ファイルが以下のディレクトリにインストールされます。

C:¥Windows¥system (Windows95, Windows98 の場合)
C:¥WINNT¥system32 (Windows NT4.0, Windows2000, Windows2003 の場合)
C:¥Windows¥system32 (WindowsXP の場合)

セットアップウィザードを実行すると自動的にアプリケーション配布時に必要な OCX (内部で利用している OCX)と、DLL ファイルが Setup.lst ファイルに追加されます。

■著作権

- TCPIPOCX およびこれに付随するマニュアルの著作権は株式会社ウィル(横浜市保土ヶ谷区)にあります。
- 本ソフトウエアおよびマニュアルを運用した結果については、当社は一切責任を負いません。
- 本ソフトウエアの仕様またはマニュアルに記載されている事項は予告無く変更することがあります。
- マニュアルなどに記載されている会社名、製品名は、各社の商標および登録商標です。
- TCPIPOCX を利用するアプリケーションは TCPIPOCX の著作権表示を行わなければなりません。Copyright プロパティに TCPIPOCX の著作権を示す文字列があります。アプリケーションまたはドキュメントのいずれかにこの文字列を表示して、TCPIPOCX を使用していることを示してください。

白紙ページ

プログラミング概要

白紙ページ

ここではTCPIPOCXの使い方の概要を述べます。メソッド、プロパティ、イベントの具体的な説明は、それぞれの項で説明していますので適宜参照してください。

通信を開始する

通信を開始する方法は2つあります。1つは、接続要求を出す方法です。もう1つは接続要求を待って接続要求がきたらそれを受け入れる方法です。ちなみに、前者の方法で通信を開始するアプリケーションをクライアント、後者の方法で通信を開始するアプリケーションをサーバーと呼びます。

接続要求を出すには、Connect メソッドを用います。接続に成功すると、Connected イベントが発生します。失敗すると、WsError イベントが発生します。

接続要求を待つには、接続要求を受け付けるコントロール（1つ）と、接続を受け入れて通信を行うコントロール（受け入れている接続の数だけ必要）を用意します。そしてまず、受付用コントロールの Listen メソッドを用いて、受付を開始します。接続要求がきたら、受付用コントロールの Accepting イベントが発生します。接続要求を受け入れるには、Accepting イベントの引数である NewSocket を通信用コントロールの Accept メソッドに渡します。正しい NewSocket を Accept メソッドに渡すと、通信用コントロールの Connected イベントが発生します。

どちらの方法で通信を開始しても、connected イベントが発生した時点で送受信が可能になります。

データを受信する

データを受信すると、Received イベントが発生します。受信したデータは Data 変数に格納されています。送信されたデータは、1塊のデータが分断されて到着したり、別々に送信したデータが1つにまとまって到着したりすることがあります。ですから、ほしいデータが揃うまで待つことや、データの一部だけを切り出すなどの作業が必要になります。

なお受け取ったデータが ANSI 文字列であったとしても、ANSI 文字列から Unicode への変換は行われません。必要に応じて変換してください。

データを送信する

送信は、Sendable プロパティが True の場合に限り行うことができます。送信したいデータを文字列変数に代入して Send メソッドに渡します。Send メソッドを実行すると、Sendable プロパティはいったん False になります。そして、送信処理が完了して次の送信が可能になると、Sendable プロパティが True になり Sent イベントが発生します。

なお Unicode 文字列を送信するとき、Unicode から ANSI 文字列への変換は自動では行われません。必要に応じて変換してください。

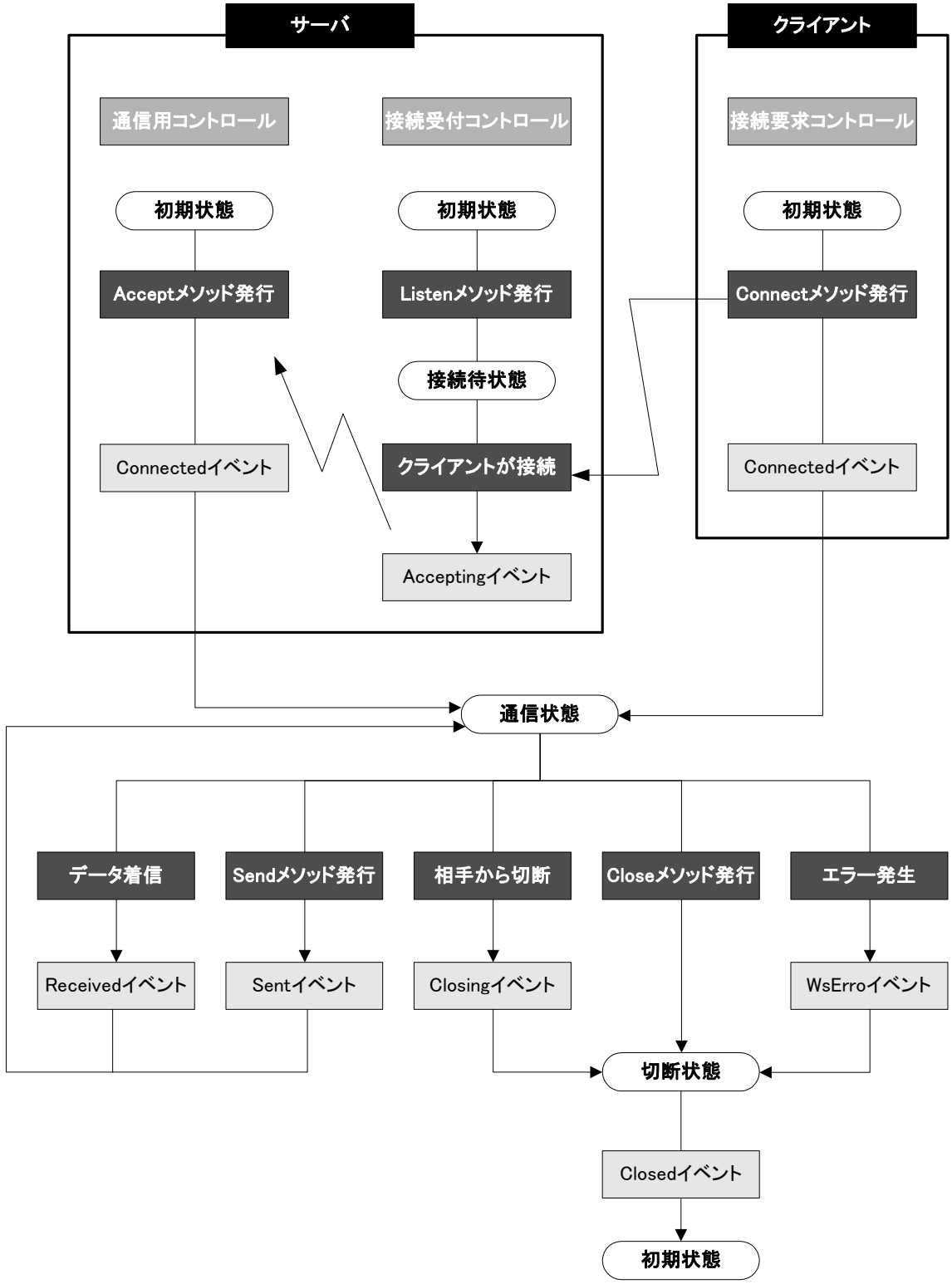
通信を終了する

通信を終了するには、close メソッドを用います。通信が終了すると、closed イベントが発生します。相手から先に通信を切断されると、closing イベントが発生します。closing イベントを抜けると、自動的に通信が終了して、closed イベントが発生します。

白紙ページ

状態遷移図

白紙ページ


サーバ
クライアント
通信用コントロール
接続受付コントロール
接続要求コントロール
初期状態
初期状態
初期状態
Acceptメソッド発行
Listenメソッド発行
Connectメソッド発行
接続待状態
Connectedイベント
クライアントが接続
Connectedイベント
Acceptingイベント
通信状態
データ着信
Sendメソッド発行
相手から切断
Closeメソッド発行
エラー発生
Receivedイベント
Sentイベント
Closingイベント
WsErroイベント
切断状態
Closedイベント
初期状態

白紙ページ

プロパティ

白紙ページ

Backlog プロパティ

■機　能

　Listen メソッドを発行する前に、接続待ちキューに入ります。未解決の接続の数を設定してください。これは 1 以上 5 以下でなければなりません。デフォルト値は、5 です。このプロパティは参照も可能です。

■構　文

Object.Backlog[=Value]
Backlog プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	未解決の接続の数を設定する整数式です。

■データ型

整数(Short)

Copyright プロパティ

■機　能

　TCPIPOCX のバージョンと著作権情報です。TCPIPOCX を用いるアプリケーションは、これを画面又はドキュメントに表示しなければなりません。この値は参照のみ可能です。

■構　文

Object.Copyright
Copyright プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

InetAddress プロパティ
InetIp プロパティ
InetName プロパティ

■機　能

　ホスト名を設定・参照することができます。なお、ホスト名の検索およびホスト名からの検索は非同期に行われるので、Resolved イベントが発生するまで有効ではありません。

■構　文

Object.InetAddress[=Value1]
Object.InetIp[=Value2]
Object.InetName[=Value3]
各プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value1	ドット区切りの IP アドレスを設定及び参照する文字列式です。
Value2	符号無し 32 ビットの IP アドレスを設定及び参照する整数式です。
Value3	ホスト名を設定する文字列式です。

■データ型

InetAddress　　文字列(String)
InetIp　　長整数(Long)
InetName　　文字列(String)

■解　説

　InetAddress プロパティ、InetIp プロパティ、InetName プロパティは連動していて、どれか一つのプロパティを変更すると、他の2つのプロパティが変化します。InetAddress プロパティは、ドット区切りの IP アドレスを、InetIp プロパティは符号無し 32 ビットの IP アドレスを、InetName プロパティはホスト名を設定及び参照することができます。

注意：通信中のオブジェクトにはこれらのプロパティをセットしないようにしてください。通信が止まることがあります。

KeepAlive プロパティ

■機　能

KeepAlive オプションの設定を行ないます。

■構　文

Object.KeepAlive[=Value]
KeepAlive プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	KeepAlive を有効にするかどうかを示すブール式です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
True	KeepAlive を有効にします。
False	KeepAlive を無効にします。

■データ型

ブール型(Boolean)

■解　説

受信待ち状態のとき、データが到着しないのは相手がデータを送信しないからなのか、或いは、通信経路にトラブルがあるのか、どちらによるものか区別が付きません。

TCP/IP には、通信経路が確保できているかどうかを、定期的に確認するための仕組みがあります。KeepAlive プロパティを True にすると、この仕組みが作動し、約 2 時間データを受信しない場合、通信相手に向けて生存確認のための TCP/IP パケットが送信されます。このパケットに相手が応答すると通信経路は正常であると判断され、接続は維持されます。このパケットに応答がない場合、通信経路に異常が発生したとして、接続を切ります。この機能は、主にサーバープロセスで利用されます。

■関連情報

パケット発生までの時間を変更するには、下記のレジストリエントリを修正します。レジストリエントリ変更後は、リブートが必要です。レジストリエントリの変更にはリスクが伴います。ご自身の責任において実行していただくようお願いします。

(Windows 95, Windows 98 の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥VxD¥MSTCP¥KeepAliveTime
生存確認までの時間(ミリ秒)。
初期値は、7,200,000 ミリ秒(2 時間)。

HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥VxD¥MSTCP¥KeepAliveInterval
生存確認に失敗したときの再確認までの時間(ミリ秒)。
初期値は、1,000 ミリ秒(1 秒)。

HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥VxD¥MSTCP¥MaxDataRetries
生存確認に失敗したときの再確認回数。
初期値は、5 回。

(Windows NT4.0, Windows2000, WindowsXP の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥Tcpip¥Parameters¥KeepAliveTime
生存確認までの時間(ミリ秒)。
初期値は、7,200,000 ミリ秒(2 時間)。

HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥Tcpip¥Parameters¥KeepAliveInterval
生存確認に失敗したときの再確認までの時間(ミリ秒)。
初期値は、1,000 ミリ秒(1 秒)。

HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥Tcpip¥Parameters¥TcpMaxDataRetransmissions
生存確認に失敗したときの再確認回数。
初期値は、5 回。

LastError プロパティ

■機　能

　Winsock で何らかのエラーが発生したとき、Winsock の返すエラーコードが格納されます。(エラーコードはエラーの種類の記述を参照して下さい。)このプロパティはエラーが発生すると上書きされます。

■構　文

Object.LastError
LastError プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

長整数(Long)

Linger プロパティ

■機　能

回線を閉じるタイミングを設定します。

■構　文

Object.Linger[=Value]
Linger プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	回線を閉じるタイミングを指定する長整数式です。次の「設定値」を参照してください。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
-1	送信待機中のデータがある場合、すべて送信したら回線を閉じます。(初期値)
0	送信待機中のデータを破棄して、回線を閉じます。 相手に WSACONRESET エラーが通知されます。
>0	指示した時間を経過したらデータを破棄して回線を閉じます。(秒)

■データ型

長整数(Long)

■解　説

通常、TCPIPOCX は、グレースフルクローズを行ないますが、あえて、ハードクローズを行ないたい場合は、Linger プロパティを 0 に設定してください。
Close メソッドでこのプロパティを参照しています。
通常、設定値は変更しないで下さい。

LocalHost プロパティ

■機　能

実行中のマシンのホスト名が格納されています。参照のみ可能です。

■構　文

Object.LocalHost
LocalHost プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

LocalIp プロパティ

■機　能

　接続が確立した時点での、ローカルホストの IP アドレスが格納されます。参照のみ可能です。

■構　文

Object.LocalIp

LocalIp プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

長整数(Long)

LocalPort プロパティ

■機　能

　接続が確立した時点での、ローカルホストのポート番号が格納されます。参照のみ可能です。

■構　文

Object.LocalPort

LocalPort プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

整数(Short)

NODELAY プロパティ

■機　能

これを True にすると、Nagle アルゴリズムを無効にします。デフォルトは False です。

■構　文

Object.NODELAY[=Value]
NODELAY プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	Nagle　アルゴリズムを有効にするか無効にするかを示すブール値です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
True	Nagle アルゴリズムを有効にします。
False	Nagle アルゴリズムを無効にします。

■データ型

ブール型(Boolean)

■単語解説

Nagle アルゴリズム

Nagle アルゴリズムは、応答確認待ちのデータがある場合に、小さいセグメント（TCP のデータ転送単位）の数を制限する方式を提案しています。

OOBINLINE プロパティ

■機　能

　このプロパティを True にすると OOB(OutOfBand)データを通常のデータとして受信します。デフォルトは False です。

■構　文

Object.OOBINLINE[=Value]
OOBINLINE プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	受信するデータの種類を指定するブール式です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
True	OOB データを通常のデータとして受信します。
False	OOB データを ReceivedOob イベントで通知します（初期値）

■データ型

ブール型(Boolean)

■単語解説

OOB データ

OOB データとは、緊急データのことです。通常のデータ受信は、Recieved イベントで受信できます。OOB は、RecievedOOB イベントで受信します。OOBINLINE を True にすると、OOB データが到着しても、RecievedOOB イベントは発生しないで、Recieved イベントが発生します。

Pause プロパティ

■機　能

イベントの抑止機能を設定します。

■構　文

Object.Pause[=Value]
Pause プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	イベントの発生を許可するブール式です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
True	イベントの発生を抑止します。
False	イベントの発生を許可します。(初期値)

■データ型

ブール型(Boolean)

■解　説

イベントの発生を一時的に停止するのに使用します。停止できるイベントは、Accepting イベント、Connected イベント、Sent イベント、Recieved イベント、Closing イベント、WsError イベントです。実行時のみ設定可能(設計時に設定したものは無視されます)。Close メソッドで False に設定されます。

RCVBUF プロパティ

■機　能

受信バッファの大きさを設定します。デフォルト値は 0 です。(システムのデフォルト値)

■構　文

Object.RCVBUF[=Value]
RCVBUF プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	受信バッファの大きさを指定する長整数式です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

(値)	(説　明)
0	システムのデフォルト値を使用します。(デフォルト値)
1 以上	受信バッファを指定の大きさに設定します。

■データ型

長整数(Long)

RemoteIp プロパティ

■機　能

　接続が確立した時点での、接続先ホストの IP アドレスを返します。正確には、Connected イベントまたは、Accepting イベント発生直前に返します。参照のみ可能です。

■構　文

Object.RemoteIp
RemoteIp プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

長整数(Long)

RemotePort プロパティ

■機　能

　接続が確立した時点での、接続先ホストのポート番号を返します。正確には、Connected イベントまたは、Accepting イベント発生直前に返します。参照のみ可能です。

■構　文

Object.RemotePort
RemotePort プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	TCPIPOCX オブジェクトです。

■データ型

整数(Short)

ReUseAddr プロパティ

■機　能

このプロパティが True のときは、2MSL 待ち状態になっているポートを即座に再利用できるようにします。False のときには、2MSL 待ち状態になっているソケット・ペアを割り当てようとすると、エラーになります。このプロパティは Listen メソッドを使用する前にセットしてください。デフォルト値は False です。Listen メソッドを使用しないクライアントアプリケーションでは、参照されません。

■構　文

Object.ReUseAddr[=Value]
ReUseAddr プロパティの構文の指定項目は次のとおりです。

（指定項目）	（内　容）
Object	TCPIPOCX オブジェクトです。
Value	ポートの利用を制御するブール値です。次の「設定値」を参照して下さい。

■設定値

Value の設定値は次のとおりです。

（値）	（説　明）
True	アドレスを即座に再利用できるようにします。
False	アドレスは 2MSL 時間待ってから再利用できます。

■データ型

ブール型(Boolean)

■単語解説

MSL

MSL とは最大セグメント寿命を示します。TCP 状態遷移において、CLOSED 状態に移行する直前の TIME_WAIT 状態になったときに、TCP コネクションが存在できる最大の時間量です。MSL は RFC によって、2 分間と定義されています。

ソケット・ペア

ソケット・ペアとは通信を行うための、「クライアントIP アドレス、クライアント・ポート番号、サーバーIP アドレス、サーバ・ポート番号の組」のことを指します。一度、接続したソケットペアは、2MSL 時間が過ぎるまで再利用できないのですが、このプロパティを True に設定することで、この制限を回避することが出来ます。

Sendable プロパティ

■機　能

　TCPIPOCX が送信可能状態かどうかを示します。送信可能状態であれば「True」、そうでなければ「False」が格納されます。参照のみ可能です。

■構　文

Object.Sendable
Sendable プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■参照値

参照値は次のとおりです。

(値)	(説　明)
True	通信可能です。
False	通信不可能です。

■データ型

ブール型(Boolean)

■解　説

　送信処理が完了するには、時間がかかります。Send メソッドを用いてデータを送信すると、Sendable は「False」となり、送信が完了して Sent イベントが発生する直前に、Sendable は「True」になります。

SendQueueBytes プロパティ

■機　能

送信バッファにあるデータサイズを取得します。参照のみ可能です。

■構　文

Object.SendQueueBytes
SendQueueBytes プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

長整数(Long)

SNDBUF プロパティ

■機　能

送信バッファの大きさを指定します。デフォルトは 0 です。(システムのデフォルト値)

■構　文

Object.SNDBUF[=Value]
SNDBUF プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	送信バッファの大きさを指定する整数式です。システムにより値の制限があります。

■データ型

長整数(Long)

Socket プロパティ

■機　能

　ソケットハンドルを格納します。-1 のときはソケットが無効状態で、通信が行われていません。参照のみ可能です。

■構　文

Object.Socket

Socket プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

長整数(Long)

■単語解説

ソケット

　ソケットとはプロセス間通信によるデータ送受信を行うアプリケーションを作成するためのプログラミング・インターフェース（API）です。

State プロパティ

■機　能

TCPIPOCX の状態を示します。参照のみ可能です。

■構　文

Object.State
State プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■参照値

参照値は次の通りです。

(値)		(説　明)
STATE_CLOSED	0	接続していません
STATE_OPEN	1	開いています
STATE_LISTENING	2	接続聴取中です
STATE_CONNECTION_PENDING	3	接続待機中です
STATE_RESOLVING_HOST	4	ホスト名解決中です
STATE_HOST_RESOLVED	5	ホスト名が解決しました
STATE_CONNECTIONING	6	接続処理中です
STATE_CONNECTED	7	接続しました
STATE_CLOSING	8	切断処理中です
STATE_ERROR	9	エラーです
STATE_SHUTDOWN_SEND	10	送信口が閉じられています
STATE_SHUTDOWN_RECV	11	受信口が閉じられています
STATE_SHUTDOWN_BOTH	12	送受信口が閉じられています

■データ型

整数(Short)

UserData1～5 プロパティ

■機　能

　アプリケーションが自由に利用できるデータ領域です。バイナリデータは格納できません。

■構　文

Object.UserData[=Value]
UserData プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	データ領域を指定する文字列式です。

■データ型

文字列(String)

UserFlag プロパティ

■機　能

アプリケーションが自由に利用できるデータ領域。バイナリデータは格納できません。

■構　文

Object.UserFlag[=Value]
UserFlag プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Value	データ領域を指定する文字列式です。

■データ型

文字列(String)

WinsockVer プロパティ

■機　能

TCPIPOCX が使用している Winsock のバージョンを格納しています。このプロパティは、デザイン時には使用できません。実行時の値の参照のみ可能です。

■構　文

Object.WinsockVer

WinsockVer プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

WinsockDesc プロパティ

■機　能

TCPIPOCX が使用している Winsock の情報を格納しています。

■構　文

Object.WinsockDesc
WinsockDesc プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

WinsockSysStat プロパティ

■機　能

TCPIPOCX が使用している Winsock のシステム状況を格納しています。

■構　文

Object.WinsockSysStat
WinsockSysStat プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

WinsockMaxSockets プロパティ

■機　能

TCPIPOCX が使用している Winsock が生成できるソケットの最大数を格納しています。

■構　文

Object.WinsockMaxSockets
WinsockMaxSockets プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

WinsockMaxDatagram プロパティ

■機　能

TCPIPOCX が使用している Winsock が生成できる datagram の最大パケット数を格納しています。

■構　文

Object.WinsockMaxDatagram

WinsockMaxDatagram プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■データ型

文字列(String)

白紙ページ

メソッド

白紙ページ

Accept メソッド

■機　能

　サーバーアプリケーションのみ使用できます。接続要求を受け入れる場合に使用します。正しい NewSocket を指定した場合、Connected イベントが発生します。以後の通信は、NewSocket を設定したコントロールを通じて行なわれます。

■構　文

Object.Accept (NewSocket As Long)
Accept メソッドの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
NewSocket	通信に使用する socket ハンドルの長整数式です。

■戻り値

なし。

■単語解説

ソケット
　ソケットとはプロセス間通信によるデータ送受信を行うアプリケーションを作成するためのプログラミング・インターフェース（API）です。

ClearSendQueue メソッド

■機　能

送信バッファーをクリアします。

■構　文

Object.ClearSendQueue()
ClearSendQueue メソッドの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■戻り値

なし。

■解　説

TCPIPOCX の送信バッファーをクリアします。
Winsock 内の送信バッファをクリアするわけではありません。

Close メソッド

■機　能

　通信を終了します。（接続要求を受け付けているソケットを閉じます。）クライアントアプリケーションでもサーバーアプリケーションでも使用できます。

■構　文

Object.Close()

Close メソッドの構文の指定項目は次のとおりです。

（指定項目）	（内　容）
Object	TCPIPOCX オブジェクトです。

■戻り値

なし。

■解　説

　このメソッドを呼び出すと、送信待ちのデータはすべて破棄されます。最終データの送信後通信を終了する場合は、送信完了の Sent イベントを受取ってから Close メソッドを使用してください。Sent イベントを受け取らず Close メソッドを使用するとデータが送信されずに通信を終了することがあります。

Connect メソッド

■機　能

引数で指定されたリモートコンピュータのポート番号で通信を開始します。

■構　文

Object.Connect (IP As String, Port As String, LocalPort As String)
Connect メソッドの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
IP	接続先のホスト名、または IP アドレスを指定する文字列式です。
Port	接続先のポート番号、またはサービス名を指定する文字列式です。
LocalPort	ローカルのポート番号、またはサービス名を指定する文字列式です。

■戻り値

なし。

■解　説

LocalPort には通常 "0" を指定します。特権ポート番号が必要なものには"-1"を指定します。これらの値が指定されると、自動的に最適なポート番号を割り振ります。

"0"を指定したときと、"-1"を指定したときの割り振りの違いについては、「ポート番号について」を参照してください。

■関連情報

Connect メソッドを発行し、名前解決が完了すると、相手に Syn パケット送信して通信を開始します。

Syn パケットに対する Ack パケットを受け取ると接続が完了し Connected イベントが発生し通信可能になります。

Syn パケットに対する Ack 応答がない場合、一定時間置いた後再度 Syn パケットを送信します。

これを何度か繰り返しても Ack 応答が得られないときはじめてエラーになり WsError イベントが発生します。

Syn パケットを再送するまでの時間は送信するたびに 2 倍にします。最初の待ち時間は 3 秒です。初期の設定では 3 回まで再送します（合計 4 回送信します）ので、

3+6+12+24=45

の計算により 45 秒後にエラーになります。

エラー発生までの時間を変更するには、下記のレジストリエントリを修正します。
レジストリエントリ変更後は、リブートが必要です。レジストリエントリの変更にはリスクが伴います。ご自身の責任において実行していただくようお願いします。

(Windows95, Windows 98 の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥VxD¥MSTCP¥MaxConnectRetries
接続に失敗したときの再確認回数。
初期値は、3 回。

(Windows NT4.0, Windows2000, WindowsXP の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥Tcpip¥Parameters¥TcpMaxConnectRetransmissions
接続に失敗したときの再確認回数。
初期値は、3 回。

htonl メソッド

■機　能

　ホストバイトオーダーのバイナリ値をネットワークバイトオーダーに変換します。戻り値 net は String 型ですが、先頭から 4 バイト分に有効な値が入っています。

■構　文

net = Object.htonl(hostlong As Long)
htonl メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
hostlong	ホストバイトオーダーの Long(4 バイト)値です。
net	ネットワークバイトオーダーの値です。型は　String　ですが、実際にはバイナリ値が格納されています。

■戻り値

文字列（String）

■解　説

　Send　メソッドを使用してバイナリ値を送る場合、バイトオーダーが重要なデータの場合、このメソッドで変換してください。net は必ず変数を用いてください。変数以外を指定するとエラーになります。

htons メソッド

■機　能

　ホストバイトオーダーのバイナリ値をネットワークバイトオーダーに変換します。戻り値 net は String 型ですが、先頭から 2 バイト分に有効な値が入っています。

■構　文

net = Object.htons(hostshort As Integer)
htons メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
hostshort	ホストバイトオーダーの Short(2 バイト)値です。
net	ネットワークバイトオーダーの値です。型は String ですが、実際にはバイナリ値が格納されています。

■戻り値

文字列(String)

■解　説

　Send メソッドを使用してバイナリ値を送る場合、バイトオーダーが重要なデータの場合、このメソッドで変換してください。net は必ず変数を用いてください。変数以外を指定するとエラーになります。

Listen メソッド

■機　能

引数で指定されたポートで接続要求を待ちます。

■構　文

Object.Listen(LocalPort As String)
Listen メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
LocalPort	接続要求を受け付けるローカルポート番号です。

■戻り値

なし。

■解　説

　接続要求がくると、Accepting イベントが発生します。接続の受け入れには、Accept メソッドを使用しなくてはなりません。

ntohl メソッド

■機　能

　ネットワークバイトオーダーのバイナリ値をホストバイトオーダーに変換します。引数 net は String 型ですが、このメソッドは先頭から 4 バイトだけを変換対象とします。

■構　文

host = Object.ntohl(netlong As String)
ntohl メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
host	ホストバイトオーダーの Long(4 バイト)値です。
netlong	ネットワークバイトオーダーの値です。型は　String　ですが、実際にはバイナリ値を指します。

■戻り値

長整数(Long)

■解　説

Received イベントの引数 Data にバイナリ値が含まれていてバイトオーダーが重要なデータのときは、このメソッドで変換してください。

ntohs メソッド

■機　能

　ネットワークバイトオーダーのバイナリ値をホストバイトオーダーに変換します。引数 net は String 型ですが、このメソッドは先頭から 2 バイトだけを変換対象とします。

■構　文

host = Object.ntohs(netshort As String)
ntohs メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
host	ホストバイトオーダーの Short(2 バイト)値です。
netshort	ネットワークバイトオーダーの値です。型は　String　ですが、実際にはバイナリ値を指します。

■戻り値

整数(Short)

■解　説

Received イベントの引数 Data にバイナリ値が含まれていてバイトオーダーが重要なデータのときは、このメソッドで変換してください。

Send メソッド

■機　能

データを送信します。

■構　文

Object.Send(data As String)
Send メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
data	送信したいデータを指定します。文字列でもバイナリデータを指定しても構いません。

■戻り値

なし。

■解　説

Unicode 文字列の場合、ANSI 文字列への変換は自動的には行なわれません。StrConv を用いて事前に変換してください。data のところには必ず変数を用いてください。

■関連情報

Send メソッドを発行し、データを送信したとき、送信パケットに対する Ack 応答がない場合、一定時間置いた後再度データを送信します。
これを何度か繰り返しても Ack 応答が得られないときはじめてエラーになり WsError イベントが発生します。
データを再送するまでの時間は送信するたびに 2 倍にします。最初の待ち時間は 3 秒です。初期の設定では 5 回まで再送します（合計 6 回送信します）ので、

3+6+12+24+48+96=189

の計算により 189 秒後にエラーになります。
エラー発生までの時間を変更するには、下記のレジストリエントリを修正します。レジストリエントリ変更後は、リブートが必要です。レジストリエントリの変更にはリスクが伴います。ご自身の責任において実行していただくようお願いします。

(Windows95, Windows98 の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services

¥VxD¥MSTCP¥MaxDataRetries
送信に失敗したときの再確認回数。
初期値は、5 回。

(Windows NT4.0, Windows2000, WindowsXP の場合)
HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Services
¥Tcpip¥Parameters¥TcpMaxDataRetransmissions
送信に失敗したときの再確認回数。
初期値は、5 回。

SendOob メソッド

■機　能

　data を OOB データとして送信します。OOB として通信する以外は Send メソッドと同じです。

■構　文

Object.SendOob(data As String)
SendOob メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
data	送信したいデータを指定します。文字列でもバイナリデータを指定しても構いません。

■戻り値

なし。

Shutdown メソッド

■機　能

送信口、受信口を閉じます。

■構　文

Object.Shutdown(how As Integer)
Shutdown メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
how	送受信口の状態を指定します。次の「設定値」を参照して下さい。

■戻り値

なし。

■設定値

設定値は次のとおりです。

(値)	(説　明)
0	受信口を閉じる。
1	送信口を閉じる。
2	送受信口を閉じる。

■解　説

通常 shutdown は送信が終了したことを相手に伝えるために、how を 1 にて使用します。こちらの送信口を閉じても相手からのデータは到着するので、相手からのデータをすべて受け取り、さらに相手の送信終了の合図をもって接続を終了するようにするのがもっともきれいな終了方法です。

StopRequest メソッド

■機　能

ホスト名やサービス名の非同期検索を中止します。

■構　文

Object.StopRequest()
StopRequest メソッドの構文の指定項目は次の通りです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■戻り値

なし。

■解　説

ホスト名やサービス名の非同期検索は close メソッドでは中止できません。

白紙ページ

イベント

白紙ページ

Accepting イベント

■機　能

接続の要求がきたときに発生します。

■構　文

Private Sub Object_Accepting(ByVal NewSocket As Long, ByVal RemoteIp As Long, ByVal RemotePort As Integer, CancelAccept As Boolean)
Accepting イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
NewSocket	受け入れるべき相手を識別する ID です。
RemoteIp	接続を要求してきた相手の 4 バイト IP 値です。
RemotePort	接続を要求してきた相手のポート番号です。
CancelAccept	受入れを拒否するときに、True にします。

■設定値

CancelAccept の設定値は次の通りです。

(値)	(説　明)
True	通信受付け不可能状態です。
False	通信受付け可能状態です。

■解　説

Listen メソッドを使って接続待ちになっているコントロール(受付コントロール)に対し、接続要求がきたときにこのイベントが発生します。通常は、受付コントロールとは別の通信用コントロールを新たに用意して、通信用コントロールの Accept メソッドを用いて接続要求を受け入れます。受付コントロールは close するまで新たな接続を受け付続けます。通信用コントロールでは、Accept メソッドが成功することにより Connected イベントが発生します。通信用コントロールは、接続要求が受付けられる度に新たに必要となります。接続要求が多すぎる場合は、CancelAccept を True にして、Accepting イベントを終了することにより拒否されます。

Closed イベント

■機　能

　通信が終了したとき発生します。通信が終了する原因としては、「Close メソッドを発行した」、「WsError が発生した」、「Closing イベントが発生して CancelClose を True にしなかった」などがあります。

■構　文

Private Sub Object_Closed()
Closed イベントの構文の指定項目は次のとおりです。

（指定項目）	（内 容）
Object	TCPIPOCX オブジェクトです。

Closing イベント

■機　能

相手が送信口を閉じたときに発生します。

■構　文

Private Sub Object_Closing(CancelClose As Boolean)
Closing イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
CancelClose	イベント後に Close 処理をするかどうかを示します。次の「設定値」を参照してください。

■設定値

CancelClose 設定値は次の通りです。

(値)	(説　明)
True	Close 処理を行わない。
False	Close 処理を行う。

■解　説

Closing イベントは、相手が close メソッドを発行した場合、または shutdown により明示的に送信口を閉じた場合に発生します。このイベント処理後、自動的に通信は終了され、closed イベントが発生します。送信を継続するために、自動的に通信が終了するのを拒否するには、CancelClose を True にして Closing イベントを終了してください。

Connected イベント

■機　能

相手と接続したときに発生します。

■構　文

Private Sub Object_Connected()
Connected イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■解　説

このイベントが発生したときは、sendable プロパティが True で、
LocalIP/LocalPort/RemoteIP/RemotePort に正しい値が入っていることを保証します。

Resolved イベント

■機　能

InetName、InetAddress、InetIp のいずれかのプロパティにデータをセットしたときに、データの検索が終わると発生します。引数で渡されるデータは、すべて対応するプロパティで確認することが出来ます。

■構　文

Private Sub Object_Resolved(ByVal Ip As Long, ByVal Name As String, ByVal Address As String)

Resolved イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
Ip	４バイトの IP アドレス値が渡されます。
Name	ホスト名が渡されます。
Address	ドット区切りの IP アドレスが渡されます。

Received イベント

■機　能

　通信相手からデータが到着したときに発生します。渡される引数に受け取ったデータが格納されています。

■構　文

Private Sub Object_Received(data As String)
Received イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
data	socket から読み込んだデータが渡されます。

■解　説

　渡されたデータを Visualbasic の文字列処理関数で処理するには、Unicode に変換しなくてはなりません。

ReceivedOob イベント

■機　能

　OOB データを受信したときに発生します。data は MSG_OOB により受信したデータです。

■構　文

Private Sub Object_ReceivedOob(data As String)
ReceivedOob イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。
data	socket から読み込んだデータが渡されます。

Sent イベント

■機　能

Sendメソッドが実行された後、再び送信可能状態になったとき発生します。このとき、Sendable は True となっています。

■構　文

Private Sub Object_Sent()

Sent イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

Shutdowned イベント

■機　能

送信口を閉じたことを知らせます。

■構　文

Private Sub Object_Shutdowned()
Shutdowned イベントの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	TCPIPOCX オブジェクトです。

■解　説

Shutdown メソッドで送信口を閉じる指示を行なうと、すべてのデータを送信したあとに実際の送信口を閉じる作業を行ないます。(それまでの間送信口は閉じられません)送信口を閉じたタイミングをこのイベントを使ってアプリケーションに通知します。

■新機能

TCPIPOCX2.12 から Send メソッド実行直後に Shutdown メソッドを実行することができるようになりました。(いままでは、Sent イベント内で Shutdown しなければなりませんでした)これにより正しく終了する方法はより一層簡単になりました。つまり、終了したくなった場合は、どのタイミングでも Shutdown 1 を実行すれば正しく終了できます。

WsError イベント

■機　能

何らかのエラーが発生したときに発生します。この後、Closed イベントが発生します。なお、CancelClose は TCPIPOCX の内部デバッグ用途にのみ使用しますので、設定しないでください。

■構　文

Private Sub Object_WsError(ByVal Ecode As Long, ByVal Description As String, ByVal Where As String, CancelClose As Boolean)
WsError イベントの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	TCPIPOCX オブジェクトです。
Ecode	エラーコードです。
Description	エラー内容です。
Where	エラーが発生した関数名です。
CancelClose	このイベント後に Close 処理をするかどうか指定します。次の「設定値」を参照してください。

■設定値

CancelClose 設定値は次の通りです。

(値)	(説　明)
True	Close 処理を行わない。
False	Close 処理を行う。(デフォルト)

ポート番号について

白紙ページ

通常、Connect メソッドの LocalPort 引数には"0"を指定します。 TCPIPOCX は Winsock に自動的にポート番号を割り当てるように要求します。

Rsh プログラムのように、特別なポート(特権ポートと呼びます)を使用するときには、Connect メソッドの LocalPort 引数に"-1"を指定します。TCPIPOCX 自身が特権ポートの中から最適なポート番号を探し出して、それを使用します。具体的に割り当てられるポート番号は次のとおりです。

ポート番号の指定	範囲
0	1024-5000
-1	512-1023

なお、1 から 1023 までのポートは「予約済みポート」と呼ばれます。また、非特権サーバを作成するときには、5001 以降のポート番号を 指定します。なぜなら、5001 以降はシステム(Winsock と TCPIPOCX)によって自動的に割り当てられることはないからです。

白紙ページ

エラーについて

白紙ページ

エラーの種類

番 号	内 容
29901	書き込み不可能な状態
29902	CONNECT していません
29903	Winsock でエラーが発生しています (番号には下記記述の Winsock エラーコードが格納されます。)
29904	プロパティの値が不正です
29905	このプロパティは読み出せません
29906	読み出し(recv)エラーです
29907	すでに CONNECT しています
29908	メモリー不足です
Winsock エラー	以下の Winsock エラーコードを参照してください。

Wiinsoc エラーコード

10004	システムコールの割り込みが発生
10009	無効なソケット番号を指定した
10013	アクセスが拒否された
10014	間違ったアドレスを指定した
10022	無効な引数
10024	開いているファイル数が多すぎる
10035	ブロッキングモードなら操作がブロックした
10036	ブロッキング処理中に別の API を呼び出した
10037	非ブロッキング処理中に同じ API を呼び出した
10038	ソケット以外に対してソケット操作が行われた
10039	宛先のアドレスが必要
10040	メッセージが長すぎる
10041	ソケットに対するプロトコルの種類が不正
10042	プロトコルが使用できない
10043	プロトコルがサポートされていない
10044	ソケットの種類がサポートされていない
10045	操作がソケットでサポートされていない
10046	プロトコルファミリがサポートされていない
10047	アドレスファミリがサポートされていない
10048	アドレスがすでに使用中
10049	要求されたアドレスを割り当てられない
10050	ネットワークが落ちている
10051	ネットワークが到達できない
10052	リセットによりネットワーク接続が落とされた
10053	ホスト内部で接続を中断した
10054	相手が接続を強制的に切った
10055	利用可能なバッファスペースが存在しない
10056	ソケットがすでに接続されている
10057	ソケットが接続されていない
10058	ソケットの遮断後であるため送信できない
10059	参照数が過大：結合不可能
10060	時間切れで connect または send に失敗した
10061	接続が拒否された
10062	シンボリックリンクが多すぎる
10063	ファイル名が長すぎる
10064	ホストが落ちている
10065	ホストへの到達経路が存在しない
10091	ネットワークサブシステムが使用不能
10092	Winsock がサポートしていないバージョンを指示した
10093	Winsock が初期化されていない
10101	接続が切れている
11001	ホストが見つからない
11002	やり直し可能なエラー
11003	回復不可能なエラー
11004	データがない

サンプル

白紙ページ

FINGER クライアント

（FINGER Ver1.0）

●概　要

FINGER は RFC742 で規定されたプロトコルを用いて FINGER サーバーと通信を行うことにより、ユーザー情報を表示します。FINGER は、ポート 79 で受信している FINGER サーバーに接続し、ユーザー名に CRLF を付加して送信します。FINGER サーバーがユーザー名に対する情報を送信してきますので、FINGER は、受信したデータを画面に表示します。FINGER サーバーはすべてのデータを送信すると、通信を切断しますので、FINGER は、相手が切断したことを検知して、通信を閉じます。

●プログラミング概要

FINGER プログラムは下記の手順で通信を行います。

① マンドボタンが押されたとき Connect メソッドを用いて通信を開始します。
② Connected イベントでユーザー名を送信します。
③ Recieved イベントで受信したデータを表示します。

ECHO サーバー

（ECHOSRV Ver1.6）

●概　要

ECHO SERVER は、ポート 7 で接続を待ちます。接続要求が来ると、サーバーは接続を許可し、その後送信されてくる文字列をそのまま送信します。

●プログラミング概要

ECHO SERVER プログラムは下記の手順で通信を行います。

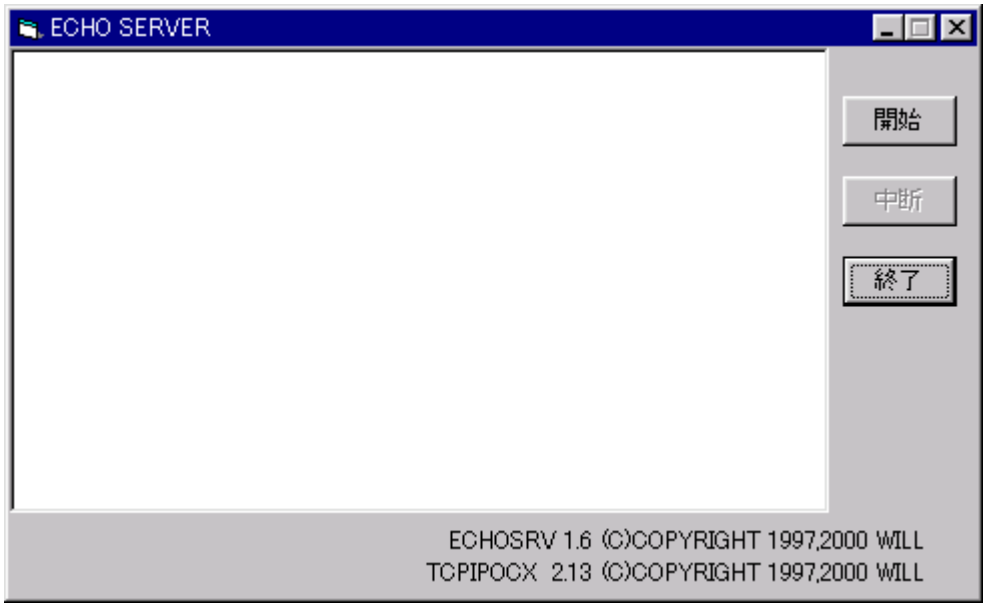
①2つの TCPIP コントロールを用意します。1つを受付用に、もう一つを通信用にします。通信用コントロールは、コントロール配列にします。

②コマンドボタンが押されたとき 受付用コントロールの Listen メソッドを用いて接続受付を開始します。

③接続要求が来たら Accepting イベントが発生します。通信用コントロールを新しく用意して、その Accept メソッドを用いて接続を受け入れます。

④通信用コントロールの Recieved イベントで受信したデータをそのまま送信します。

⑤通信用コントロールの Closed イベントで対象となる通信用コントロールを破棄します。



メールチェック

（MAILCHK Ver1.1）

●概　要

MAILCHK は POP プロトコルを利用して、メールが到着しているかどうかを調べるプログラムです。このプログラムを応用すると、独自のメールプログラムを作成することができるようになります（なるかもしれません）。

●プログラミング概要

変数 Action に何をしているのかを記憶しておきます。

- ″INIT″ これから Connect メソッドを発行します。
- ″USER″ Send メソッドを用いて USER コマンドを発行しました。
- ″PASS″ Send メソッドを用いて PASS コマンドを発行しました。
- ″QUIT″ Send メソッドを用いて QUIT コマンドを発行しました。
- ″CLOSE″ Close メソッドを発行しました。

発行したメソッド・コマンドと、それに対するサーバーからのメッセージ(Reply)に応じて処理を切り替えます。

CONNECT クライアント

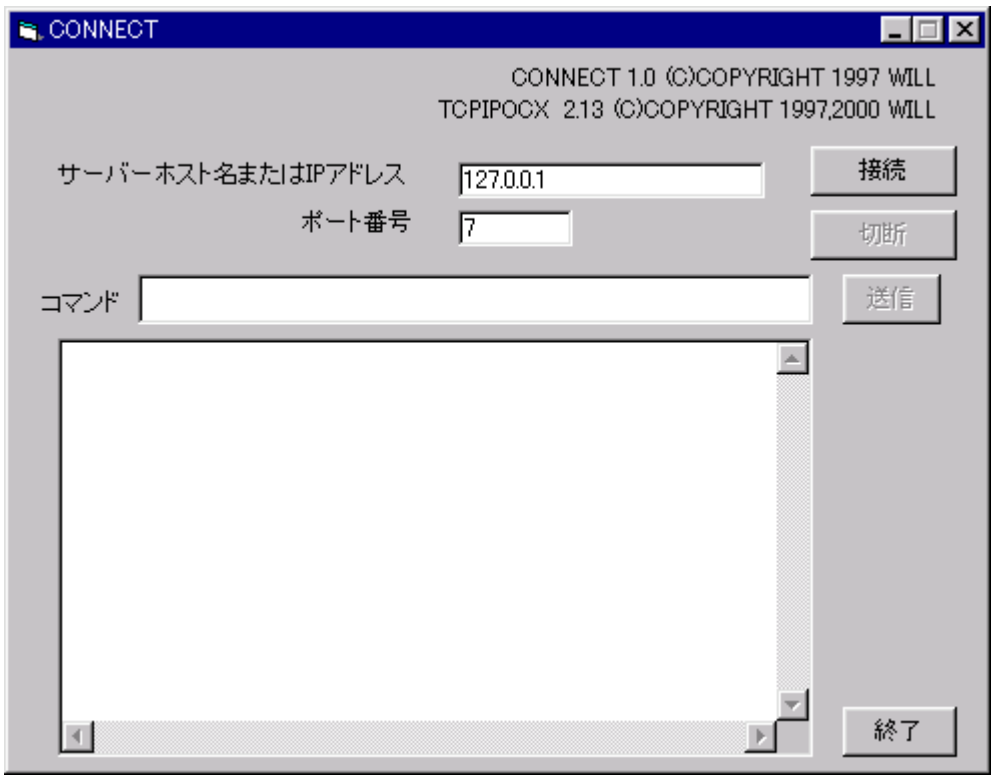
（CONNECT Ver1.0）

●概 要

CONNECT は汎用のインターネットクライアントです。これを用いると、FINGER サーバー、メールサーバー、WHOIS サーバーなど、各種のインターネットサーバーに対してコマンドを送ることができます。 CONNECT を利用するには、サーバーの IP アドレスとポート番号が判っていることと、さらにサーバーの要求するプロトコルの知識が必要です。

例えば、メールサーバーのアドレスが分かっている場合は、サーバーの IP アドレスをいれ、ポート番号を 25 に設定して接続コマンドを押してみてください。うまく接続できたなら、画面にサーバーからのメッセージが表示されます。

コマンドとして、HELP を入力して、送信ボタンを押すと、画面にヘルプが表示されます。QUIT を入力して、送信ボタンを押すと、接続がきれます。



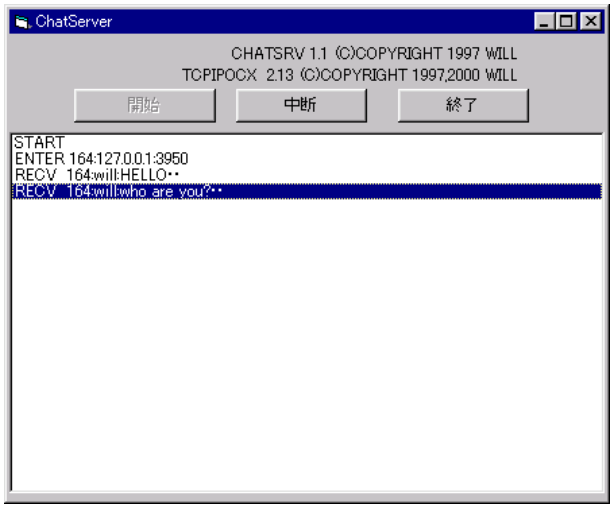
チャット

（CHATSRV Ver1.1 , CHATCLT VER1.1）

●サンプルプログラム利用方法

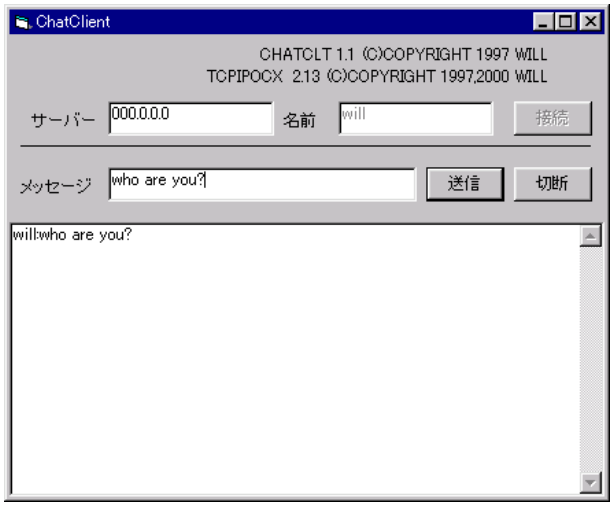
<サーバー>

①サーバーを起動して、開始ボタンを押してください。
　中断ボタンを押すと新規の接続ができなくなりますが、接続中のものはそのまま継続して通信をおこなうことができます。
②終了するには、終了ボタンを押した上でウィンドウを閉じます。



<クライアント>

①クライアントを起動して、サーバーのIPアドレスあるいはドメインをサーバーエリアに入力し、接続を押してください。
②送りたい文字をメッセージエリアに入力して、エンターキーを押せば、接続中のサーバー全員にメッセージが送信されます。また、#who# というメッセージを送信すると、接続中のユーザーのリストが出ます。
③終了するには、切断ボタンを押した上でウィンドウを閉じます。



ファイル転送

（File Tran Client Ver1.0、File Tran Server Ver1.0）

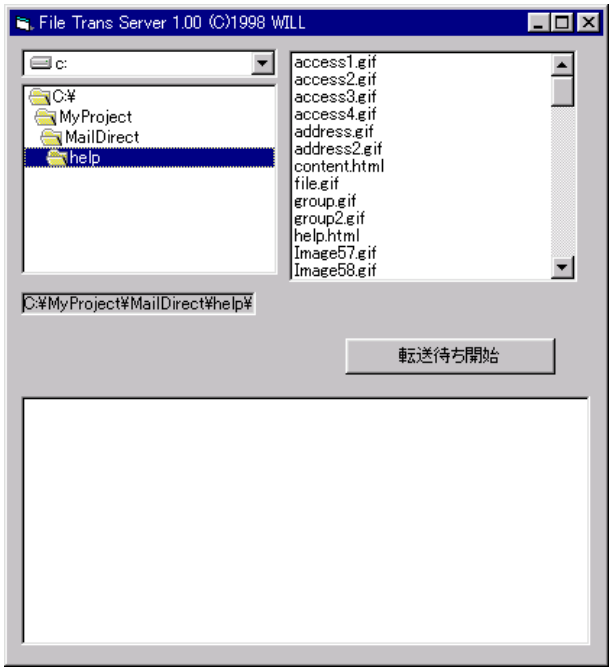
●概　要

FileTranServer は、TCP/IP を用いるファイルを転送するためのプログラムです。クライアント、サーバを組み合わせて利用して下さい。

●サンプルプログラム利用方法

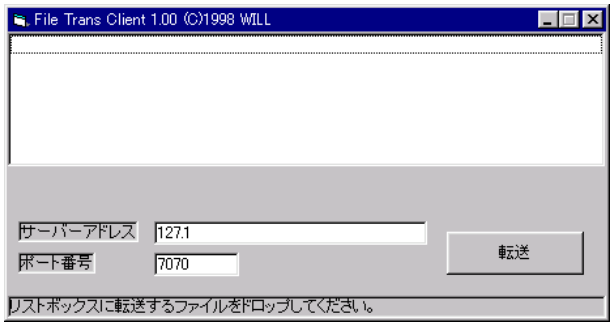
<サーバー>

① サーバーを起動して、「転送待ち開始」ボタンを押してください。
　ファイル転送先のフォルダを指定してください。



<クライアント>

① クライアントを起動して、サーバーの IP アドレスと、ポート番号を指定してください。
② 転送するファイルをリストボックスにドロップしてください。
③ 「転送」を押すとファイルがサーバに転送されます。



アドレス解決

（RESOLVE Ver1.0）

●概 要

ホスト名、ドット区切りの IP アドレスと符号無し 32 ビットの IP アドレスをそれぞれ参照するプログラムです。

TCPIPOCX は、ホスト名を InetName に、ドット区切りの IP アドレスを InetAddress に、符号無し 32 ビットの IP アドレスを InetIp に格納しています。三つのプロパティは連動していて、設定・参照することができます。どれか一つのプロパティを変更すると、ホスト名やサービス名の非同期検索が行い、他の2つのプロパティが変化します。検索が終わると Resolved イベントが発生し、三つのプロパティは参照できるようになります。

パケット　クライアント

（PACKET Ver1.0）

●概　要

PACKET は、固定長ヘッダーを持つデータ転送方式のサンプルです。固定長ヘッダーにデータ長、データ型などの情報を含めることにより、処理を階層化できるようになります。

PACKET は、ECHO サーバーを利用して、送信を行う部分と受信を行う部分の説明を1つのプログラムで行っています。

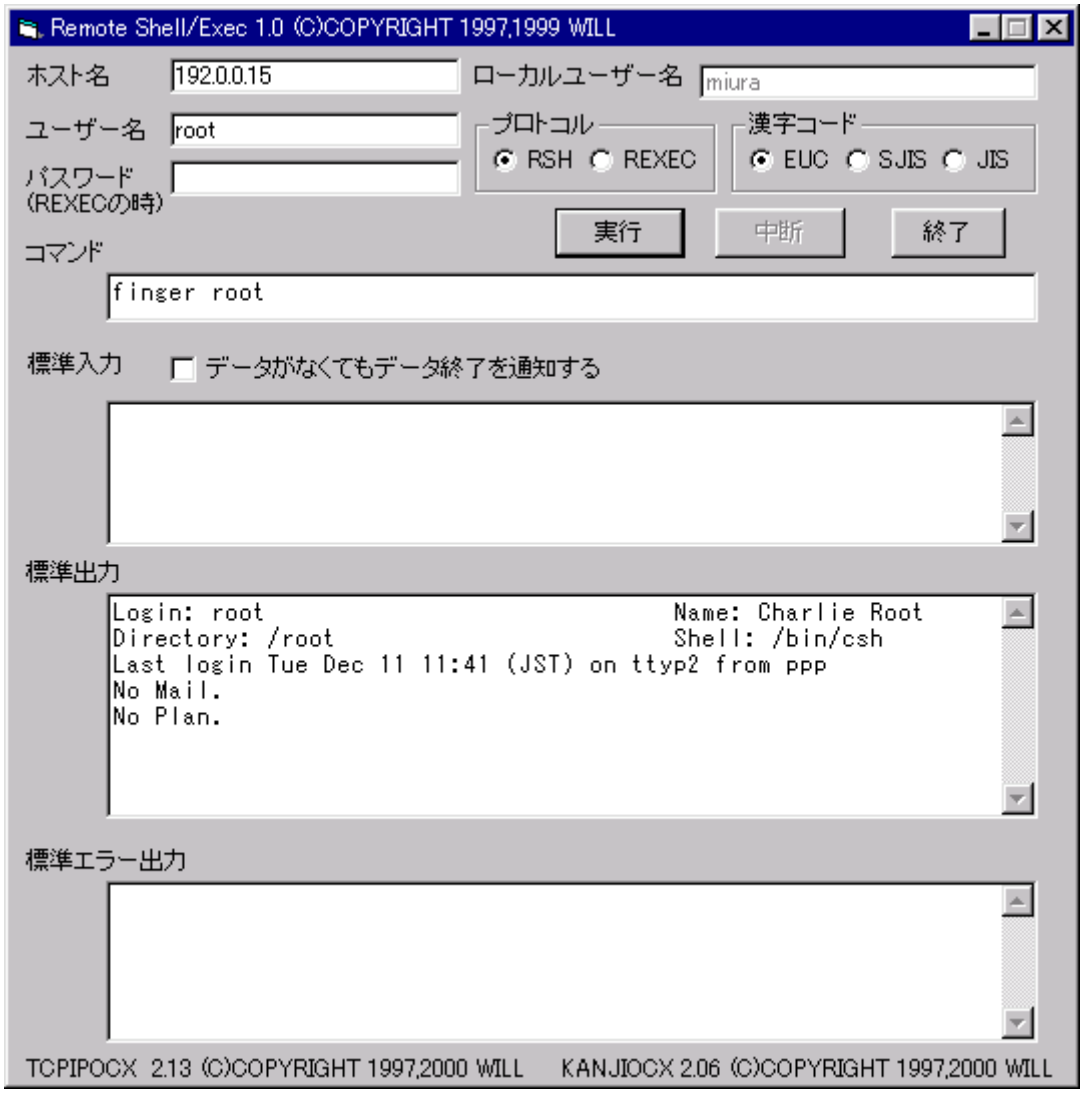
また、PACKET は、整数を送受信するテクニックを紹介しています。データ長に 32 ビット整数、データ型に 16 ビット整数を用いて送受信しています。

リモートシェル／実行

（Remote Sell/Exec Ver1.0）

●概 要

Remote Sell/Exec.EXE は BSD 系 UNIX の rexecd と通信を行い、サーバー上の UNIX コマンドを実行し、その結果を画面に表示します。サーバーは、UNIX コマンドを実行する前に、パスワードの確認を行います。パスワードは、ネットワーク上を暗号化されていない文字でながれます。



WILL TELNET

(WILLNET Ver1.0)

●概 要

WILL TELNET は TELNET 通信ソフトです。送信時と受信時で別々に指定の漢字コード(SJIS/JIS/EUC)に変換可能です。(ご利用の際には、TCPIPOCX 及び KANJIOCX が必要です。)

●サンプルプログラム利用方法

① リモート コンピュータに接続するには

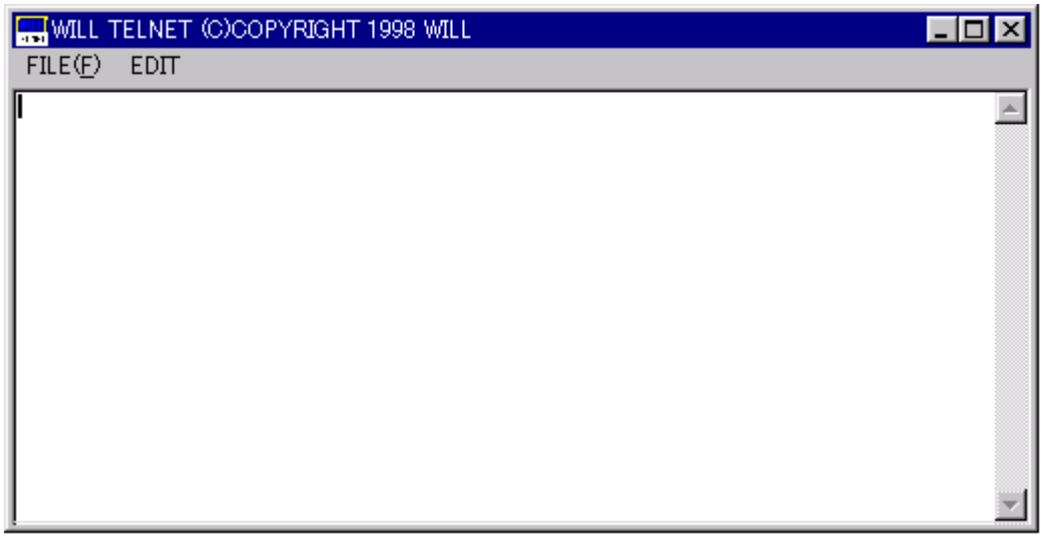
1. [FILE] メニューの [CONNECT] をクリックします。
2. [サーバー] ボックスに、接続するリモート システムの IP アドレスを入力します。
3. [ポート] ボックスで、使用するポート番号を指定します。
4. 端末の種類を指定するには、[ターミナルタイプ] ボックスで、使用する文字列を入力する。(ディフォルトは"dumb"となっています。)
5. [CONNECT]ボタンをクリックします。

② 送信文字コードの指定

1. [EDIT] メニューの [SEND CODE] を選択します。
2. 指定する文字コードをクリックします。

③ 受信文字コードの指定

1. [EDIT] メニューの [RECV CODE] を選択します。
2. 指定する文字コードをクリックします。



索 引

<W>

白紙ページ

TCPIPOCX マニュアル

1998 年 1 月 31 日 初版第 1 版
1999 年 6 月 1 日 第 2 版
1999 年 7 月 1 日 第 3 版
2001 年 6 月 15 日 第 4 版
2002 年 6 月 1 日 第 5 版
発行所 株式会社ウィル
〒240－0022
神奈川県横浜市保土ヶ谷区西久保町 15 グランディシンヤ 302
TEL：045－338－3525
FAX：045－338－3526
Mail-Address：info@will-ltd.co.jp
URL：http://www.will-ltd.co.jp /
発行者 小川 史彦