IPCOCX マニュアル

WILL

株式会社ウィル

目　次

IPCOCX

Created 2000.03.21

【製品概要】

〈 特 徴 〉

IPCOCX は、同一マシンにおけるプロセス間通信(IPC:Inter Process Communication)をサポートする 32 ビット Active X コンポーネントです。
IPCOCX がサポートするプロセス間通信機能は、

1. プロセスの起動および起動したプロセスの終了の検出
2. プロセス間の排他制御(ミューテクス、セマフォー)
3. プロセス間の通知および待合せ制御(コンディション)
4. プロセス間のメモリ共有
5. ディレクトリ変更の検出

の5つです。
IPCOCX は、Microsoft Visual C++ Ver5.0 で作成しています。サンプルは、Microsoft Visual Basic Ver.5.0 で作成しています。

〈 動作環境 〉

● 対応OS
Windows 95, Windows 98, WidowsNT 4.0, Windows 2000, Windows XP, Windows 2003

● 開発に必要なソフトウェア
IPCOCX 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)

〈 販 売 〉

IPCOCX は1ヶ月間無料で試用することが出来ます。試用期間中はトライアルライセンスをご利用ください。試用期間終了後も引き続き使用される場合は、正規ライセンス申請を行ってください。詳しくは弊社ホームページの「お申し込み」のページをご覧ください。但し、教育用途でのライセンス料支払いを免除します。

〈 再 配 布 〉

IPCOCX を単体あるいはアプリケーションに組込んでの再配布を許可します。(ライセンスの再配布はできません。)

〈 著 作 権 〉

- IPCOCX およびこれに付随するマニュアル、サンプルプログラム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、サンプルプログラム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。
- 本ソフトウエアおよびサンプルプログラム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。

- IPCOCX を利用するアプリケーションは IPCOCX 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 IPCOCX 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、IPCOCX を使用していることを示してください。

【プログラミング概要】

ここでは、IPCOCX 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内容はそれぞれの項で説明していますので、適宜参照して下さい。

1. プロセスの起動および起動したプロセスの終了の検出

IPCOCX のプロセスの起動および起動したプロセスの終了監視機能を使うと以下の機能を実現できます。

(1) プログラムを順番に起動する(バッチ処理)
(2) 起動したプログラムの2重起動を禁止する(メニュー処理)
(3) プロセスの突然死を検出し再起動する(デーモン)

また、起動するプログラムの表示位置の指定や、非表示での起動ができるので、複数のプログラムをあたかも1つのプログラムであるかのように見せることもできます。

<プロセスの起動>

IPCOCX でプロセスの起動を行なうには以下の 2 つの方法があります。

1. ProcessRun メソッドを使用
2. ProcessExec メソッドを使用

どちらのメソッドもプロセスの起動をするとともに、プロセスの終了を検出するための専用スレッド(プロセス監視スレッド)を生成することができます。メソッドを実行しているスレッドと、プロセス監視スレッドは同時に平行して動作しますので、メソッドはブロックせずに次の処理を行なうことができます。プロセス監視スレッドが開始すると、ProcessOpened イベントが発生します。

<プロセス終了の検出>

プロセスの終了を検出するには、ProcessOpened イベント内またはそれ以降に ProcessWait メソッドを呼び出します。プロセスの終了の検出を行う直前に ProcessWaiting イベントを発生します。

プロセスの終了を検出すると、ProcessDone イベントが発生します。

プロセス監視スレッドを終了するには、ProcessDone イベントで ProcessClose メソッドを呼び出します。

※プロセスの起動だけを行い、終了の検出を行なわない場合は、ProcessOpened イベントで、ProcessClose メソッドを呼び出します。この呼び出しでプロセス監視スレッドは終了します。

<プロセス監視スレッドの待ち状態の中断>

プロセス監視スレッドの待ち状態(WAIT 状態)を中断する(IDLE 状態にする)場合は、ProcessAbort メソッドを呼び出します。このメソッドが成功すると、ProcessAborted イベントが発生し、Reason 変数に 2 が格納されます(Abort メソッドによる中断)。中断後再度終了の検出を行なうには、ProcessWait メソッドを再度呼び出します。

ProcessWait メソッドは Timeout プロパティを参照します。

Timeout が-1 に設定されている場合、プロセス監視スレッドがプロセスの終了を検出するまでイベントは発生しません。

-1 以外に設定されている場合は、指定された時間だけ監視を行ない、その時間内にプロセスの終了が検出されない場合は、ProcessAborted イベントを発生します。このとき、Reason 変数に 1 が格納されます(タイムアウトによる中断)。

プロセス監視スレッドは、コントロールが破棄される場合を除き、自動的には終了しません。終了するには、ProcessClose メソッドを呼び出します。ProcessClose メソッドを呼び出すと ProcessClosed イベントが発生します。ProcessClose メソッドは、いつでも呼び出すことができます。

2. プロセス間の排他制御(ミューテクス、セマフォー)

排他制御は、1 つ以上の資源を複数のプロセスで共有する際に、同時に利用できるプロセスの数を制限するための機能です。

IPCOCX は、ミューテクスまたはセマフォーと呼ばれる Windows の機構を用いて、排他制御を実現します。

ミューテクス:1 つのプロセスにだけ資源へのアクセス権を与える仕組み
セマフォー :同時に複数のプロセスに資源へのアクセス権を与える仕組み

排他制御は椅子取りゲームに似ています。1 つの椅子を取り合うときはミューテクス、複数の椅子を取り合うときはセマフォー、というように理解するとよいでしょう。

ミューテクスやセマフォーを用いても実際の資源を管理できるわけではありません。資源に名前をつけてその名前に対するアクセス権があるかどうかを調べる手段を提供するにすぎません。ですから、資源を使うすべてのプログラムが同じ方法で資源の排他制御を行なう必要があります。

ミューテクスやセマフォーを使わずに同様の排他制御を行なうには、例えば、ファイルのロック機構を使うとか、ソケットを用いて排他制御サーバーを作成するなどがありますが、ミューテクスやセマフォーを使う方がはるかに簡単で高速です。

〈ミューテクスを利用する〉

IPCOCX でミューテクスを利用するには、まず、MutexOpen メソッドでミューテクスの利用を宣言します。この時ミューテクスの状態監視のためのスレッド(ミューテクス状態監視スレッド)が生成され、それと同時に MutexOpened イベントが発生します。

ミューテクスのアクセス権を確保(LOCK)するには MutexLock メソッドを呼び出します。ミューテクス状態監視スレッドはアクセス権の確保を行なう直前に、MutexLocking イベントを発生します。

その後、アクセス権が確保(LOCK)されるまでミューテクス状態監視スレッドは停止しますが、MutexLock メソッドを呼び出したスレッドは実行を続けることができます。

この時はまだ、アクセス権は確保(LOCK)はされていないので、資源へのアクセスは控えてください。

アクセス権が確保(LOCK)されると、MutexLocked イベントが発生します。このイベントを合図に資源へのアクセスを開始して下さい。

資源へのアクセスが終わったら、MutexUnlock メソッドを呼び出してアクセス権の解放を行なってください。正常に解放されると、MutexUnlocked イベントが発生します。

アクセス権が確保(LOCK)されるまでミューテクス状態監視スレッドは停止していますが、この状態を MutexAbort メソッドを用いてキャンセルすることができます。キャンセルに成功すると、MutexAborted イベントが発生して、MutexOpened イベントが発生した直後の状態に戻ります。このとき、MutexAborted イベントの Reason 変数に 2 が格納されます(Abort メソッドによる中断)。

MutexLock メソッドは、Timeout プロパティを参照します。

Timeout が-1 に設定されている場合は、アクセス権が確保(LOCK)できるまでミューテクス状態監視スレッドは停止します。-1 以外に設定されている場合は、指定された時間だけ停止し、その時間内にアクセス権が確保(LOCK)できないない場合は、停止状態を解除して、MutexAborted イベントを発生させます。このとき、Reason 変数に 1 が格納されます(タイムアウトによる中断)。

ミューテクス状態監視スレッドはコントロールが破棄される場合を除き、自動的には終了しません。終了するには、MutexClose メソッドを呼び出します。MutexClose メソッドを呼び出すと MutexClosed イベントが発生します。MutexClose メソッドは、いつでも呼び出すことができます。

〈セマフォーを利用する〉

IPCOCX でセマフォーを利用するには、まず、SemaphoreOpen メソッドでセマフォーの利用を宣言します。この時セマフォーの状態監視のためのスレッド（セマフォー状態監視スレッド）が生成され、それと同時に SemaphoreOpened イベントが発生します。

セマフォーのアクセス権を得るには SemaphoreCapture メソッドを呼び出します。セマフォー状態監視スレッドがアクセス権の確保(CAPTURE)を行なう直前に、SemaphoreCapturing イベントを発生します。

その後、アクセス権が確保(CAPTURE)されるまでセマフォー状態監視スレッドは停止しますが、SemaphoreCapture メソッドを呼び出したスレッドは実行を続けることができます。

この時はまだ、アクセス権は確保(CAPTURE)はされていないので、資源へのアクセスは控えてください。

アクセス権が確保(CAPTURE)されると、SemaphoreCaptured イベントが発生します。このイベントを合図に資源へのアクセスを開始して下さい。

資源へのアクセスが終わったなら、SemaphoreRelease メソッドを呼び出してアクセス権の解放を行なってください。正常に解放されると、SemaphoreReleased イベントが発生します。

アクセス権が確保(CAPTURE)されるまでセマフォー状態監視スレッドは停止していますが、この状態を SemaphoreAbort メソッドを用いてキャンセルすることができます。

キャンセルに成功すると、SemaphoreAborted イベントが発生して、SemaphoreOpened イベントが発生した直後の状態に戻ります。このとき、SemaphoreAborted イベントの Reason 変数に 2 が格納されます(Abort メソッドによる中断)。

SemaphoreCapture メソッドは、Timeout プロパティを参照します。

Timeout が-1 に設定されている場合は、アクセス権が確保(CAPTURE)できるまでセマフォー状態監視スレッドは停止します。-1 以外に設定されている場合は、指定された時間だけ停止し、その時間内にアクセス権が確保(CAPTURE)できないない場合は、停止状態を解除して、SemaphoreAborted イベントを発生させます。このとき、Reason 変数に 1 が格納されます(タイムアウトによる中断)。

セマフォー状態監視スレッドはコントロールが破棄される場合を除き、自動的には終了しません。終了するには、SemaphoreClose メソッドを呼び出します。SemaphoreClose メソッドを呼び出すとSemaphoreClosedイベントが発生します。SemaphoreCloseメソッドは、いつでも呼び出すことができます。

上記のように、確保したアクセス権をそのオブジェクトが必ず解放する方法を同期的使用方法と呼びます。

同期的使用方法を行なうには、SemaphoreSyncMode プロパティを True にします。

一方、SemaphoreSyncMode プロパティを False にすると、非同期的使用方法になります。

非同期的使用方法にすると、アクセス権を確保するオブジェクトとアクセス権を提供するオブジェクトが分離され、アクセス権を確保するオブジェクトは SemaphoreRelease メソッドを使うことができなくなります。

アクセス権を提供するオブジェクトは、SemaphoreIncrease メソッドを使って、アクセス権を与えるプロセスの数を任意に増加させることができます。

3. プロセス間の通知および待合せ制御(コンディション)

複数のプロセスで協調して処理を行なうには、処理の開始通知や完了通知などの通知制御と、相手の完了を待って処理を開始する待ち合わせ制御が必要になります。

コンディションを用いることでこれらを簡単に実現できます。

(セマフォーの非同期的使用法を用いても実現可能です)

〈コンディションについて〉

IPCOCX は、Event と呼ばれる Windows の機構を用いて、コンディションを実現します(VB にはイベントという言葉が既に用いられているため、Event に対する機能をコンディションと呼ぶことにしました)。

コンディションを理解するには、門(ゲート)を想像してください。

門には、閉じている状態と開いている状態があります。門が開いていると通行でき、門が閉じていると門が開くまで待たなければなりません。

コンディションでは、門が開いている状態を On、閉じている状態を Off と呼びます。On になると、(VB の)イベントが発生するので、門の状態が変化したことを知ることができます。

コンディションでは、On にすること、Off にすること、そして、一旦 On にして、すぐに Off にすることができます。

〈コンディションを利用する〉

IPCOCX でコンディションを利用するには、まず、ConditionOpen メソッドでコンディションの利用を宣言します。この時コンディションの状態監視のためのスレッド(コンディション状態監視スレッド)が生成され、それと同時に ConditionOpened イベントが発生します。

コンディションが On の状態になるのを検出するために ConditionFind メソッドを呼び出します。検出を行なう直前に、ConditionWaiting イベントが発生します。

コンディションが On の状態を検出すると、ConditionFound イベントが発生します。

コンディション状態監視スレッドの監視状態(WAIT 状態)を中断する(IDLE 状態にする)には ConditionAbort メソッドを呼び出します。中断に成功すると、ConditionAborted イベントが発生して、ConditionOpened イベントが発生した直後の状態に戻ります。このとき、ConditionAborted イベントの Reason 変数に 2 が格納されます(Abort メソッドによる中断)。

中断後、監視を再開するには、ConditionFind メソッドを再度呼び出します。

ConditionFind メソッドは、Timeout プロパティを参照します。

Timeout が-1 に設定されている場合は、コンディションが On である状態が検出されるまで監視を続けます。

-1 以外に設定されている場合は、指定された時間だけ監視し、その時間内にコンディションが On である状態が検出されない場合、監視を中断して、ConditionAborted イベントを発生させます。このとき、Reason 変数に 1 が格納されます(タイムアウトによる中断)。

コンディション状態監視スレッドは、コントロールが破棄される場合を除き、自動的には終了しません。終了するには、ConditionClose メソッドを呼び出します。ConditionClose メソッドを呼び出すと ConditionClosed イベントが発生します。ConditionClose メソッドは、いつでも呼び出すことができます。

コンディションの状態を変化させるには、ConditionOn メソッド、ConditionOff メソッド、ConditionOnAndOff メソッドを使用します。ConditionOn メソッドはコンディションの状態を On にします。ConditionOff メソッドはコンディションの状態を Off にします。

ConditionOnAndOff メソッドはコンディションを一瞬 On にした後すぐに Off にします。

4. プロセス間のメモリ共有

IPCOCX のメモリ共有は非常に便利でかつ安全です。

メモリへの書き込みおよび読み込みは、内部で Mutex を用いて排他的に行われます。

メモリへの書き込みおよび読み出しは、VB の変数をそのまま渡すことができるので、ローカルメモリにアクセスする手軽さです。

IPCOCX ではメモリ共有を利用するために 4 つのメソッドが用意されています。

(1)共有メモリをオープンする

IPCObject.SharedMemoryOpen 共有メモリ名,初期値[,確保バイト数]

(2)共有メモリにデータを保存する

IPCObject.SharedMemoryWrite 変数

(3)共有メモリからデータを読み出す

変数=IPCObject.SharedMemoryRead

(4)共有メモリをクローズする

IPCObject.SharedMemoryClose

初期値は、共有メモリが作成されたときの値となります。

すでに共有メモリが作成されている場合、初期値は無視されます。

IPCOCX では同じ名称の共有メモリには、初期値と同じ型のデータしか保存できません。

SharedMemoryWrite メソッドは与えれられた変数を、初期値と同じ型に変換して保存します。変換できない場合はエラーになります。

SharedMemoryRead メソッドは初期値と同じ型で値を返します。

IPCOCX の共有メモリに保存できるのは、文字列の配列およびバリアントの配列、オブジェクト、Empty 値以外のすべての型およびその 1 次元配列です。

〈整数を共有メモリに格納して取り出す例〉

```
Dim X As Integer,Y As Interger

IPC1.SharedMemoryOpen "NAME1", 1
X = IPC1.SharedMemoryRead
IPC1.SharedMemoryWrite 2
Y = IPC1.SharedMemoryRead
IPC1.SharedMemoryClose
Debug.Print X,Y
```

〈倍精度実数の配列を共有メモリに格納して取り出す例〉

共有メモリに格納した配列を取り出すときはバリアント変数で受けてください。

```
Dim D() As Double, v As Variant

ReDim D(1)
D(0) = 1
D(1) = 2
IPC1.SharedMemoryOpen "NAME2", D
v = IPC1.SharedMemoryRead
IPC1.SharedMemoryClose

Debug.Print v(0), v(1)
```

バリアントで受けることに関しては、データが期待する型かどうか不明であることに不安を感じるかもしれませんが、SharedMemoryOpenメソッドで型と大きさのチェックを行なっていますので、型が合わないという心配はありません。

〈文字列を共有メモリに格納して取り出す例〉

文字列など可変のデータを取り扱う場合、SharedMemoryOpen メソッドで確保バイト数を指定してください。これを指定しないと初期値を格納するのに必要な大きさの共有メモリしか確保しません。

```
Dim X As String,Y As String

IPC1.SharedMemoryOpen "NAME3", "",100
X = "TEST"
IPC1.SharedMemoryWrite X
Y = IPC1.SharedMemoryRead
IPC1.SharedMemoryClose
Debug.Print X,Y
```

SharedMemoryOpen メソッドを用いて、既存の共有メモリを使用する際、確保バイト数が一致しないとエラーとなります。共有を行なうすべてのプロセスで SharedMemoryOpen メソッドの指定は同じになるようにしてください。

5. ディレクトリ変更の検出

ディレクトリ変更の検出は、文字どおりディレクトリおよびディレクトリに含まれるファイルに変更があった場合に、変更を検出するための機能です。

<ディレクトリ変更の検出機能を利用する>

IPCOCX でディレクトリ変更の検出機能を利用するには、まず、NotificationOpen メソッドでディレクトリ変更検出の利用を宣言します。この時ディレクトリ変更の監視のためのスレッド(ディレクトリ変更監視スレッド)が生成され、それと同時に NotificationOpened イベントが発生します。

ディレクトリ変更の検出を行うために NotificationFind メソッドを呼び出します。ディレクトリ変更の検出を行う前に、NotificationWaiting イベントが発生します。

監視しているディレクトリに変化があると、NotificationFound イベントが発生します。

<ディレクトリ変更監視スレッドの待ち状態を中断する>

ディレクトリ変更監視スレッドの待ち状態(WAIT 状態)を中断する(IDLE 状態にする)には NotificationAbort メソッドを呼び出します。

中断に成功すると、NotificationAborted イベントが発生して、NotificationOpened イベントが発生した直後の状態に戻ります。このとき、NotificationAborted イベントの Reason 変数に 2 が格納されます(Abort メソッドによる中断)。

中断後、監視を再開するには、NotificationFind を再度呼び出します。

NotificationFind メソッドは、Timeout プロパティを参照します。

Timeout が-1 に設定されている場合は、ディレクトリに変化があるまで監視を続けます。

-1 以外に設定されている場合は、指定された時間だけ監視し、その時間内にディレクトリに変化がなければ、監視を中断して、NotificationAborted イベントを発生させます。このとき、Reason 変数に 1 が格納されます(タイムアウトによる中断)。

ディレクトリ変更監視スレッドはコントロールが破棄される場合を除き、自動的には終了しません。終了するには、NotificationClose メソッドを呼び出します。NotificationClose メソッドを呼び出すと NotificationClosed イベントが発生します。NotificationClose メソッドは、いつでも呼び出すことができます。

【共通の注意点】

(1) ミューテクス、セマフォー、共有メモリの資源名称に関する注意

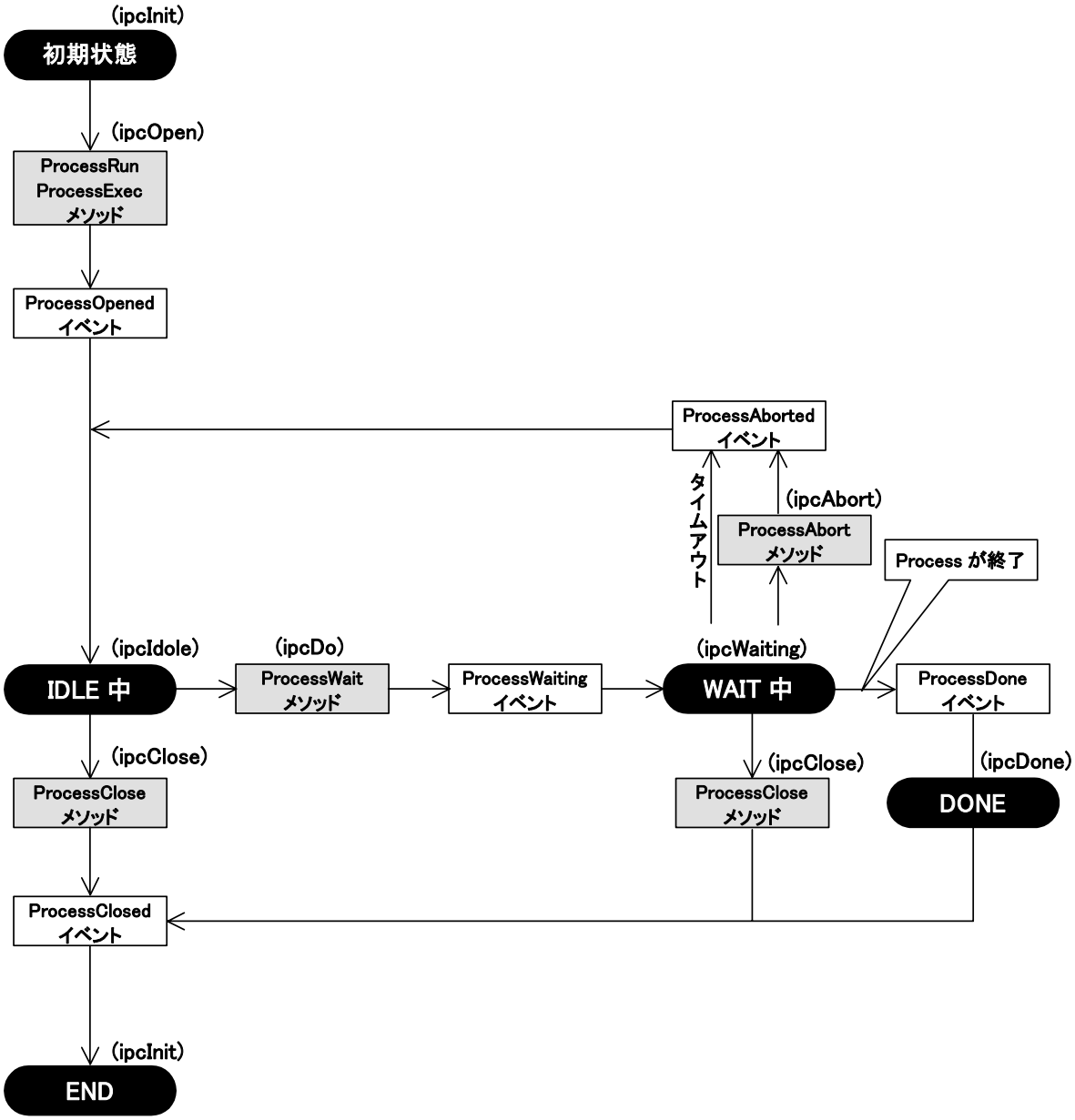
資源名称のための名前空間を共有しています。
資源名称には¥(円マーク)は含めることができません。
大文字小文字は区別されます。
名前の長さは、250 バイト以内にしてください。
(260 バイト中 10 バイトは IPCOCX が予約しています)

(2) 1 つの IPCOCX オブジェクト(インスタンス)では下記のメソッドは、呼び出した機能が完了するまでほかの機能は呼び出せません。複数の機能を同時に使いたい場合は必要な数の IPCOCX オブジェクトを用意してください。

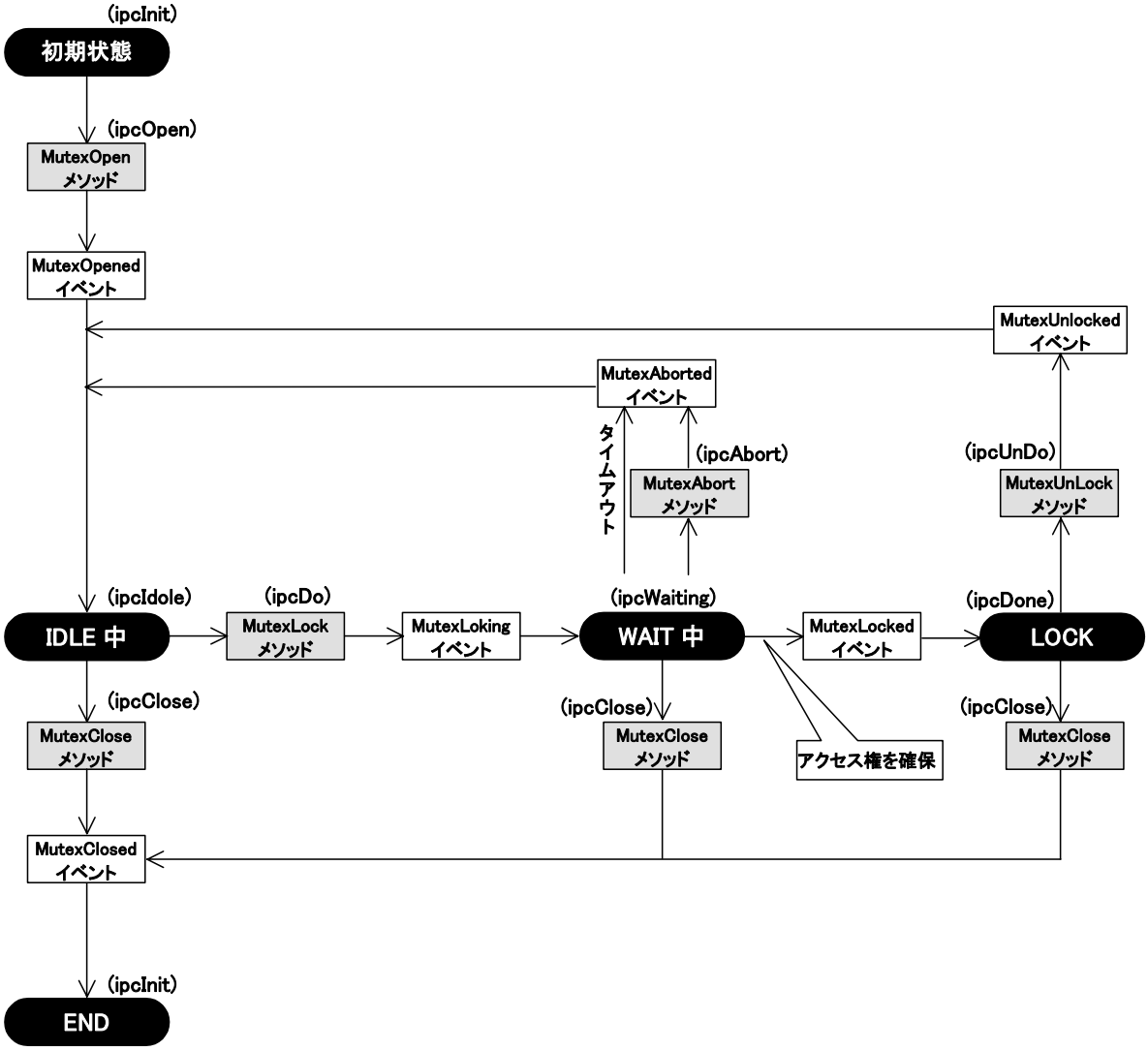
ProcessRun
ProcessExec
MutexOpen
SemaphoreOpen
Notification
ConditionOpen

【IPCOCX 状態遷移図】

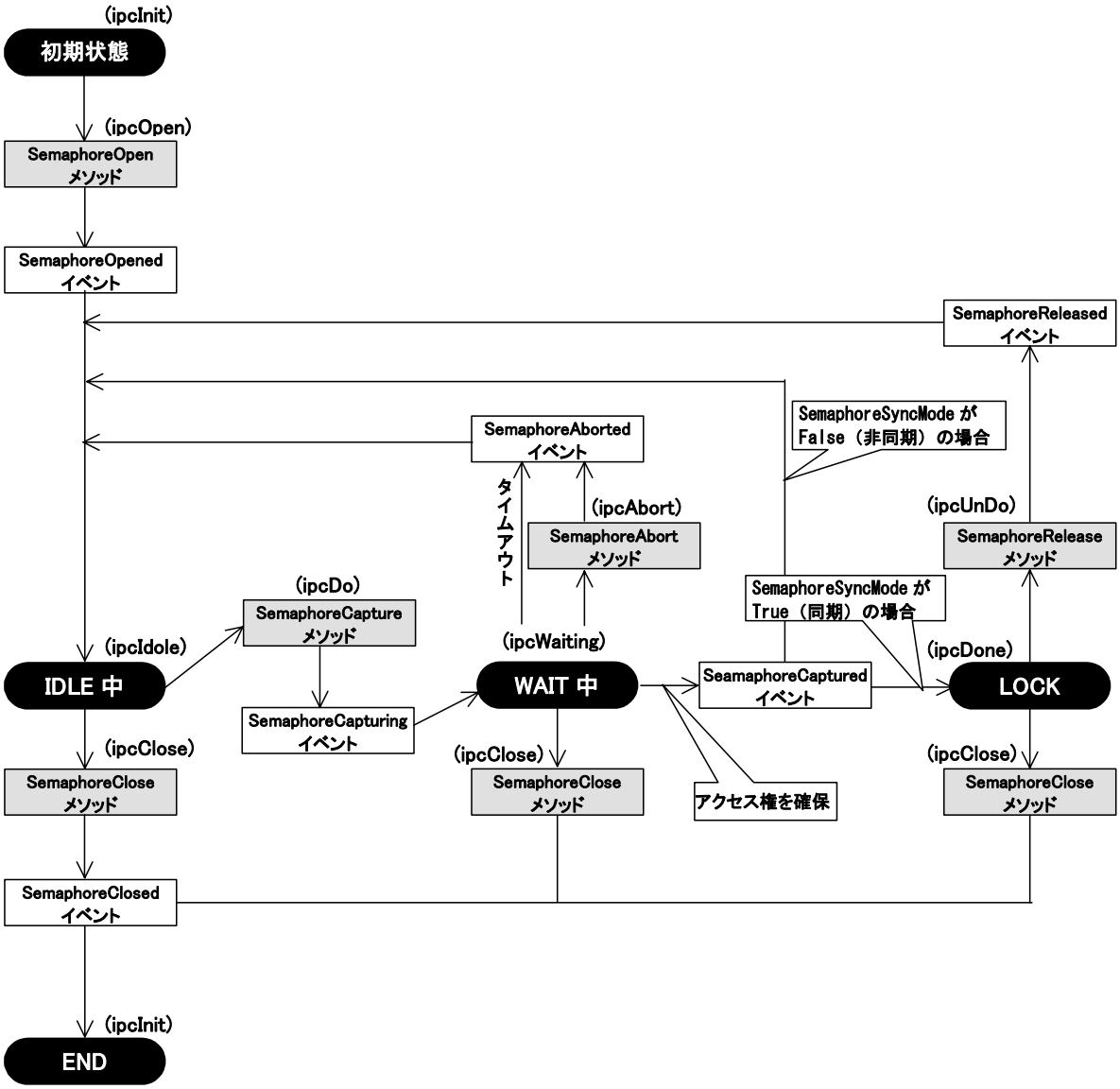
〈Process〉



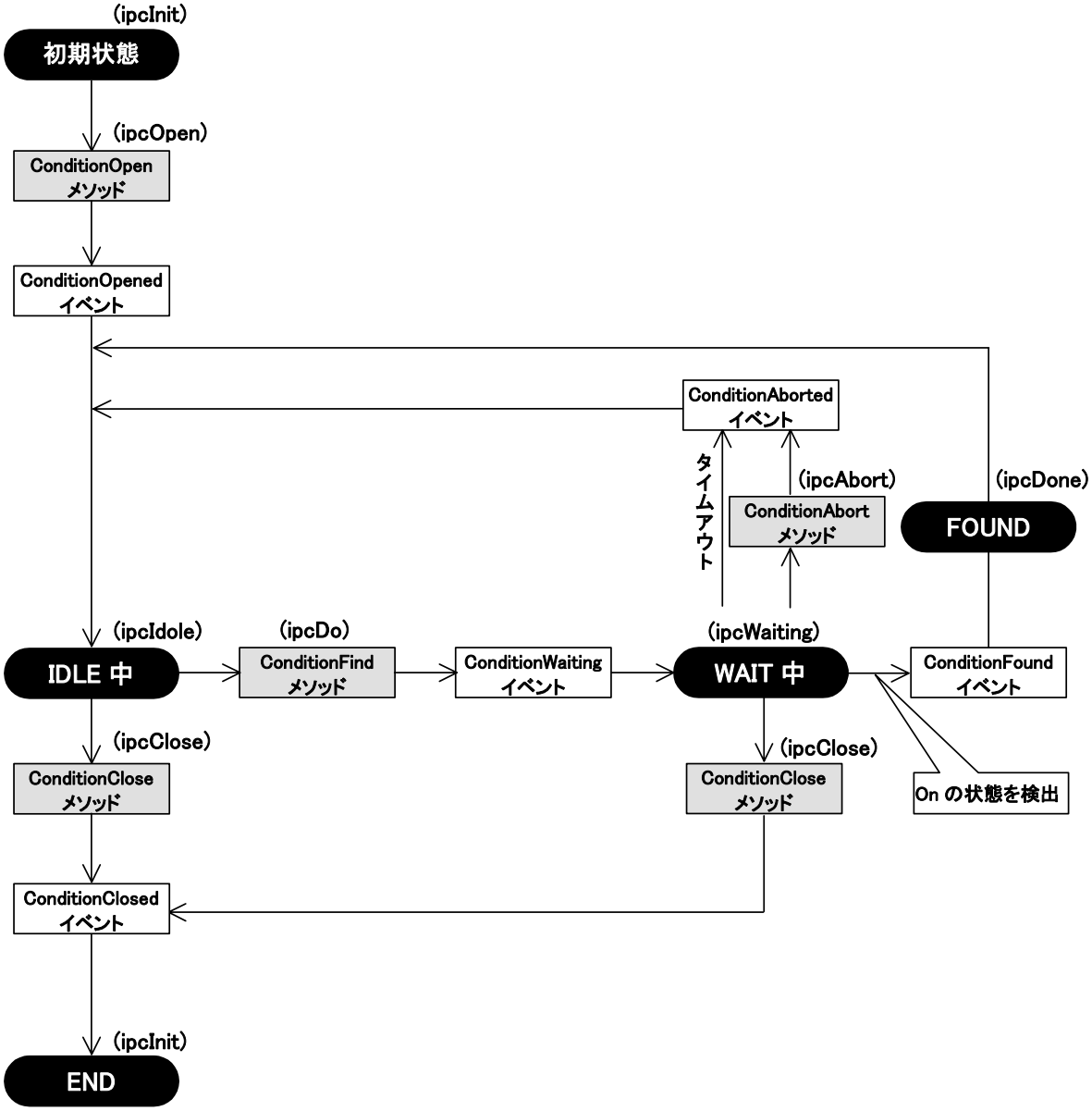
〈Mutex〉


(ipcInit)
初期状態
(ipcOpen)
MutexOpen
メソッド
MutexOpened
イベント
MutexUnlocked
イベント
MutexAborted
イベント
タイムアウト
(ipcAbort)
MutexAbort
メソッド
(ipcUnDo)
MutexUnLock
メソッド
(ipcIdole)
IDLE 中
(ipcDo)
MutexLock
メソッド
MutexLoking
イベント
(ipcWaiting)
WAIT 中
MutexLocked
イベント
(ipcDone)
LOCK
(ipcClose)
MutexClose
メソッド
(ipcClose)
MutexClose
メソッド
アクセス権を確保
(ipcClose)
MutexClose
メソッド
MutexClosed
イベント
(ipcInit)
END

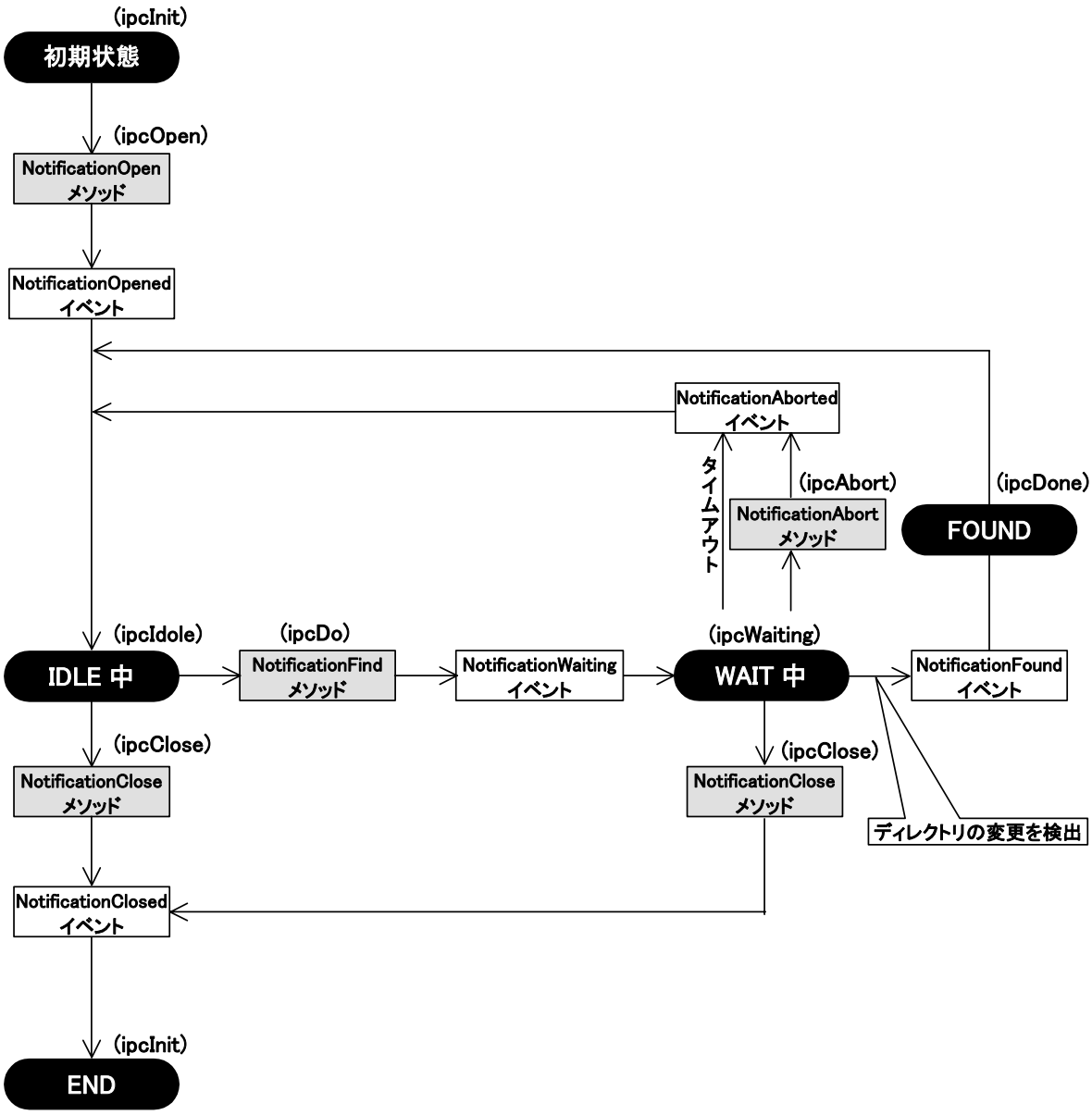
<Semaphore>


(ipcInit)
初期状態
(ipcOpen)
SemaphoreOpen
メソッド
SemaphoreOpened
イベント
SemaphoreReleased
イベント
SemaphoreAborted
イベント
SemaphoreSyncMode が
False（非同期）の場合
タイムアウト
(ipcAbort)
SemaphoreAbort
メソッド
(ipcUnDo)
SemaphoreRelease
メソッド
(ipcDo)
SemaphoreCapture
メソッド
SemaphoreSyncMode が
True（同期）の場合
(ipcIdole)
(ipcWaiting)
(ipcDone)
IDLE 中
WAIT 中
SeamaphoreCaptured
イベント
LOCK
SemaphoreCapturing
イベント
(ipcClose)
(ipcClose)
(ipcClose)
SemaphoreClose
メソッド
SemaphoreClose
メソッド
アクセス権を確保
SemaphoreClose
メソッド
SemaphoreClosed
イベント
(ipcInit)
END

<Condition>


(ipcInit)
初期状態
(ipcOpen)
ConditionOpen
メソッド
ConditionOpened
イベント
ConditionAborted
イベント
タイムアウト
(ipcAbort)
ConditionAbort
メソッド
(ipcDone)
FOUND
(ipcIdole)
IDLE 中
(ipcDo)
ConditionFind
メソッド
ConditionWaiting
イベント
(ipcWaiting)
WAIT 中
ConditionFound
イベント
(ipcClose)
ConditionClose
メソッド
(ipcClose)
ConditionClose
メソッド
On の状態を検出
ConditionClosed
イベント
(ipcInit)
END

<Notification>



【プロパティ】

SemaphoreSyncMode プロパティ

［ 機　　能 ］ セマフォーのアクセス権の解放を義務付けるかどうかを設定します。

［ 構　　文 ］ object.SemaphoreSyncMode[=MODE]
SemaphoreSyncMode プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

［データ型］ ブール型(Boolean)

(値)	(状 態)
True	初期値。解放は必要
False	解放は不要

［ 解　　説 ］ このプロパティが True の場合、SemaphoreCapture メソッドで確保したセマフォーへのアクセス権は SemaphoreRelease メソッドを用いて解放します。解放しないまま、SemaphoreCapture メソッドを呼び出すことはできません。
逆に、このプロパティが False の場合、SemaphoreCapture メソッドで確保したセマフォーへのアクセス権は SemaphoreRelease メソッドを用いて解放することはできません。解放する場合は、SemaphoreIncrease メソッドを使用します。

SharedMemoryLockMode プロパティ

［ 機　　能 ］ 共有メモリのロック方法を指定します。

［ 構　　文 ］ object.SharedMemoryLockMode [=LOCKMODE]
SharedMemoryLockMode プロパティの構文の指定項目は次の通りです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

［データ型］ 長整数(Long)

(値)	(状 態)
ipcUseMutexLock =0	共有メモリへの読み書きに MutexLock を用いる(初期値)
IpcIgnoreTimeout =1	共有メモリへの読み書きに MutexLock を用いるが、エラーやタイムアウトを無視して読み書きする
ipcNoMutexLock=2	共有メモリへの読み書きに MutexLock を用いない

［ 解　　説 ］ 共有メモリで排他制御を行なわない場合は、ipcNoMutexLock を設定します。
共有メモリで排他制御を行なうが、タイムアウトやエラーが発生した場合、これを通知せずに強制的に読み書きを行なう場合は、ipcIgnoreTimeout を設定します。
通常は、ipcUseMutexLock を設定して、共有メモリへの読み書きの際に MutexLock を用いて排他制御を行ないます。初期値は、0(ipcUseMutexLock)。

SharedMemoryTimeout プロパティ

[機　能] 共有メモリで排他制御を行なう際のタイムアウト時間をミリ秒単位で設定します。
初期値は、10,000(10 秒)。
タイムアウトをするとトラップ可能なエラーが発生します。
-1 を設定するとタイムアウトしなくなります。

[構　文] object.SharedMemoryTimeout [=TIMEOUT]
SharedMemoryTimeout プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

[データ型] 長整数(Long)

(値)	(状態)
-1	タイムアウトしない。(設定しないでください)
0	待たない。
10,000	デフォルトタイムアウト値。(推奨)
正	タイムアウト値(ミリ秒)。
上記以外	不定。

[解　説] このプロパティを参照するメソッドは、
SharedMemoryOpen
SharedMemoryRead
SharedMemoryWrite
の 3 つです。

Timeout プロパティ

[機　能] メソッド実行時のタイムアウト時間をミリ秒単位で設定します。

[構　文] object.Timeout [=TIMEOUT]
Timeout プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

[データ型] 長整数(Long)

(値)	(状 態)
-1	初期値。タイムアウトしない。
0	待たない。
正	タイムアウト値。
上記以外	不定。

[解　説] このプロパティは、ProcessWait,MutexLock,SemaphoreCapture,
NotificationFindConditionFind の各メソッドで参照されます。
Timeout プロパティを-1 以外に設定するとタイムアウトイベントが発生します。
タイムアウトイベントは、起動メソッドによって名称が異なります。

Licensee プロパティ,Serial プロパティ

[機　能] IPCOCX のライセンス情報です。バージョン情報画面からいつでもライセンスの変更が可能です。トライアルライセンスから正規ライセンスに切り替えた場合、このプロパティでライセンス情報を確認してください。

[構　文] object.Licensee
object.Serial
Licensee プロパティ,Seria プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

[データ型] 文字列(String)

Copyright プロパティ

[機　能] IPCOCX のバージョン情報です。
アプリケーション作成者は、アプリケーションプログラムまたはマニュアルに、ここで表示される文字列を表記するようにしてください。

[構　文] object.Copyright
Copyright プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

[データ型] 文字列(String)

WorkDir プロパティ

[機　能] プログラムを実行する際のカレントディレクトリを指定します。

[構　文] object.WorkDir [=WORKING DIRECTORY]
WorkDir プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

[データ型] 文字列(String)

(値)	(状 態)
""	初期値。現在のカレントディレクトリのまま
上記以外	指定したディレクトリをカレントディレクトリにして起動する

[解　説] ProcessRun メソッド,ProcessExec メソッドを使うときに参照されます。
正しいディレクトリを指定しないと、これらのメソッドを呼び出した時点でエラーになります。

ShowStyle プロパティ

［機　能］ プログラムを実行する際の表示方法を指定します。

［構　文］ object.ShowStyle [=WINDOW STYLE]
ShowStyle プロパティーの構文の指定項目は次のとおりです。

(指定項目)	(内 容)
Object	IPCOCX オブジェクトです。

［データ型］ 整数(Short)

(値)		(説 明)
IpcHIDE	=0	非表示
ipcSHOWNORMAL	=1	通常表示
IpcSHOWMINIMIZED	=2	アイコン表示
ipcSHOWMAXIMIZED	=3	最大表示
IpcSHOWNOACTIVATE	=4	非アクティブ通常表示
ipcSHOW	=5	通常表示
IpcMINIMIZE	=6	非アクティブアイコン表示
ipcSHOWMINNOACTIVE	=7	非アクティブアイコン表示
IpcSHOWNA	=8	非アクティブアイコン表示
IpcRESTORE	=9	通常表示

［解　説］ ProcessRun メソッド,ProcessExec メソッドを使うときに参照されます。
値は、0 から 9 までの数値を指定することも可能です。
異なる値で同じ動作をする場合があります。
この値は初期値としてアプリケーションで使われるもので、この指示に従わない場合があります。

【メソッド】
Stat メソッド

［ 機　 能 ］ 現在実行中の待ち状態を示します。

［ 構　 文 ］ R = object.Stat()

Stat 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	IPCOCX オブジェクトです。

［データ型］ 長整数(Long)

(値)		(状 態)
ipcInit	=0	指示可能。
ipcOpen	=1	(*1)のメソッドが呼び出された。
ipcOpened	=2	スレッドが開始した。(*2)のイベントが発生。
ipcDo	=3	(*3)のメソッドが呼び出された。
ipcWaiting	=4	監視する直前。(*4)のイベントが発生。
ipcDone	=5	検出した。(*5)のイベントが発生。
ipcUnDo	=6	(*6)のメソッドが呼び出された。
ipcAbort	=7	(*7)のメソッドが呼び出された。
ipcClose	=8	(*8)のメソッドが呼び出された。

*1: ProcessRun
ProcessExec
MutexOpen
SemaphoreOpen
ConditionOpen
NotificationOpen

*2: ProcessOpened
MutexOpened
SemaphoreOpened
ConditionOpened
NotificationOpened

*3: ProcessWait
MutexLock
SemaphoreCapture
ConditionFind
NotificationFind

*4: ProcessWaiting
MutexLocking
SemaphoreCapturing
ConditionWaiting
NotificationWaiting

*5: ProcessDone
MutexLocked
SemaphoreCaptured
ConditionFound
NotificationFound

*6: MutexUnlock
SemaphoreRelease

*7: ProcessAbort
MutexAbort
SemaphoreAbort
ConditionAbort
NotificationAbort

*8: ProcessClose
MutexClose
SemaphoreClose
ConditionClose
NotificationClose

[解　説] ProcessRun メソッド、ProcessExec メソッド、MutexOpen メソッド、SemaphoreOpen メソッド、ConditionOpen メソッド、NotificationOpen メソッドを呼び出すと、状態の変化を検出するために専用のスレッドを作成してそのスレッド上で待機します。
このスレッドが存在する間は、これらのメソッドを平行して呼び出すことはできないようになっています。
Stat メソッドはスレッドの状態を示します。
(メソッドが呼び出されている場合は状態が変化していることを示します)

ProcessRun メソッド

[機　能] プロセスを起動します。

[構　文] object.ProcessRun ProgramName[,X,Y]
ProcessRun 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	IPCOCX オブジェクトです。
ProgramName	実行するプログラム名
X	表示開始位置 X 方向(オプション)
Y	表示開始位置 Y 方向(オプション)

[解　説] プロセスを起動します。起動できたら起動したプロセスの終了を待つための専用スレッド(プロセス監視スレッド)を生成し ProcessOpened イベントを発生します。
ProcessRun メソッドを実行しただけでは、プロセスの終了を検出できません。
プロセスの終了を検出するためには、ProcessOpened イベントで ProcessWait メソッドを実行します。

ProcessExec メソッド

[機　能] プロセスを起動します。

[構　文] object.ProcessExec ProgramName[,Operation]
ProcessExec 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	IPCOCX オブジェクトです。
ProgramName	ファイル名
Operation	オペレーション。デフォルトは"Open"

[解　説] ファイルに関連付けられたプロセスを起動します。起動できたら起動したプロセスの終了を待つための専用スレッド(プロセス監視スレッド)を作成し ProcessOpened イベントを発生します。ProcessExec メソッドを実行しただけでは、プロセスの終了を検出できません。
プロセスの終了を検出するためには、ProcessOpened イベントで ProcessWait メソッドを実行します。
オペレーションは、関連付けに関わりがあります。
"open"以外に"print"や"edit"などのオペレーションがサポートされていることがあります。

ProcessWait メソッド

[機　能] ProcessRun メソッドまたは、ProcessExec メソッドで起動したプロセスの終了を検出します。

[構　文] object.ProcessWait
ProcessWait 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	IPCOCX オブジェクトです。

[解　説] ProcessWait メソッドは ProcessRun メソッドまたは ProcessExec メソッドで生成されたスレッドにプロセスの終了の監視を依頼します。依頼するとまず、ProcessWaiting イベントが発生します。つぎに監視を開始します。監視をしている間そのスレッドは停止しています。プロセスの終了を検出するとスレッドは再開し、ProcessDone イベントを発生します。

ProcessAbort メソッド

[機　能] ProcessWait メソッドの実行をキャンセルします。

[構　文] object.ProcessAbort
ProcessAbort 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	IPCOCX オブジェクトです。

[解　説] ProcessAbort メソッドはプロセス終了監視状態を解除します。
解除に成功すると、ProcessAborted イベントが発生します。
解除をしてもプロセス監視スレッドは終了しません。

ProcessClose メソッド

［ 機　能 ］ プロセスの監視スレッドを終了します。

［ 構　文 ］ object.ProcessClose

ProcessClose 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	IPCOCX オブジェクトです。

［ 解　説 ］ プロセス監視スレッドに対し、スレッドを終了するよう要請します。
終了できたなら、ProcessClosed イベントを発生します。

ProcessKill メソッド

［ 機　能 ］ 起動しているプロセスを強制終了させます。

［ 構　文 ］ object.ProcessKill

ProcessKill 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	IPCOCX オブジェクトです。

［ 解　説 ］ ハングアップを検出した場合に使うということを想定しています。
通常の終了には絶対使用しないでください。
ProcessKILL メソッドを使用すると、そのプロセスで確保されていた資源を解放しないまま終了することがあります。

MutexOpen メソッド

［ 機　能 ］ ミューテクスをオープンします。

［ 構　文 ］ R = object.MutexOpen (MutexName)
object.MutexOpen MutexName

MutexOpen 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	IPCOCX オブジェクトです。
MutexName	ミューテクスの名前

［データ型］ ブール型(Boolean)

(R)	(戻 値)
True	新規にミューテクスオブジェクトを作成しました。
False	既存のミューテクスオブジェクトをオープンしました

［ 解　説 ］ MutexName で指定された名前のミューテクスオブジェクトをオープンます。
指定された名前のミューテクスオブジェクトが無い場合は新規に作成します。
正常にオープンできると、ミューテクスの状態を変更したり監視するためのスレッド（ミューテクス状態監視スレッド）を生成します。ミューテクス状態監視スレッドが生成されると、MutexOpened イベントが発生します。
MutexOpen メソッドだけでは、ミューテクスのアクセス権は確保されません。
MutexOpened イベントが発生した以降 MutexLock メソッドを使ってミューテクスのアクセス権を確保してください。
MutexClosed イベントが発生してミューテクス状態監視スレッドが終了するまで、
このオブジェクトの MutexOpen、SemaphoreOpen、ConditionOpen、ProcessRun、ProcessExec、NotificationOpen の各メソッドを呼び出すことはできません。

MutexLock メソッド

［ 機　能 ］ ミューテクスのアクセス権を確保します。

［ 構　文 ］ object.MutexLock
MutexLock 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	IPCOCX オブジェクトです。

［ 解　説 ］ MutexOpen メソッドでオープンしたミューテクスオブジェクトのアクセス権を確保します。
MutexLock メソッドは、MutexOpen メソッドで生成した監視スレッドに依頼してアクセス権の確保を行ないます。
ミューテクス状態スレッドがアクセス権の確保の監視に入る直前に MutexLocking イベントを発生します。その後、アクセス権を確保できたら、MutexLocked イベントが発生し、確保できないと、MutexAborted イベントが発生します。
MutexAborted イベントは、タイムアウトが発生したとき、または MutexAbort メソッドが呼ばれたときに発生します。
確保できたかどうかを監視するのは MutexOpen メソッドで生成したスレッドで行いますので、MutexLock メソッドがブロックすることはありません。
MutexLocked イベント発生までは、資源を利用することはできませんが、ほかの処理を行なうことは可能です。

MutexUnlock メソッド

［ 機　能 ］ ミューテクスのアクセス権を解放します。

［ 構　文 ］ object.MutexUnlock
MutexUnlock 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	IPCOCX オブジェクトです。

［ 解　説 ］ MutexOpen メソッドで生成したスレッドがアクセス権を保持していますので、このスレッドに対し、確保したアクセス権を解放するよう要請します。
解放されたなら、MutexUnocked イベントが発生します。
アクセス権の解放は、ミューテクス状態監視スレッドの終了とは違います。
MutexUnlocked イベントが発生したあとに、再度 MutexLock メソッドを使ってアクセス権の確保を試みることができます。

MutexAbort メソッド

［ 機　能 ］ ミューテクスのアクセス権の確保をあきらめます。

［ 構　文 ］ object.MutexAbort
MutexAbort 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	IPCOCX オブジェクトです。

［ 解　説 ］ MutexLock メソッドを実行しているスレッドに対し、監視を中断するよう要請します。
中断できたなら、MutexAborted イベントを発生します。
監視の中断をしても、ミューテクス状態監視スレッドが終了するわけではありませんので、MutexAborted イベントが発生したあとに、再度 MutexLock メソッドを使ってアクセス権の確保を試みることができます。
終了させるには、MutexClose メソッドを呼び出します。

MutexClose メソッド

［ 機　能 ］ ミューテクス状態監視スレッドを終了します。

［ 構　文 ］ object.MutexClose
MutexClose 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	IPCOCX オブジェクトです。

［ 解　説 ］ ミューテクス状態監視スレッドに対し、スレッドを終了するよう要請します。
終了できたなら、MutexClosed イベントを発生します。
終了後は、MutexLock,MutexUnlock,MutexAbort,MutexClose メソッドを使用できません。

SemaphoreOpen メソッド

［ 機　能 ］ セマフォーをオープンします。

［ 構　文 ］ R = object.SemaphoreOpen (SemaphoreName,MaxCount,InitialCount)
object.SemaphoreOpen SemaphoreName SemaphoreName,MaxCount,InitialCount
SemaphoreOpen 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	IPCOCX オブジェクトです。
SemaphoreName	セマフォーの名前
MaxCount	許容する最大の同時アクセス数(1 以上)

InitialCount　　初期に許される同時アクセス数(0 以上 MaxCount 以下)

[データ型] ブール型(Boolean)

(R)	(戻 値)
True	新規にセマフォーオブジェクトを作成しました。
False	既存のセマフォーオブジェクトをオープンしました。

[解　説] SemaphoreName で指定された名前のセマフォーオブジェクトをオープンします。
指定された名前のセマフォーオブジェクトが無い場合は新規に作成します。
正常にオープンできると、セマフォーの状態を変更したり監視するためのスレッド(セマフォー状態監視スレッド)を生成します。セマフォー状態監視スレッドが生成されると、SemaphoreOpened イベントが発生します。
SemaphoreOpen メソッドだけでは、セマフォーのアクセス権は確保されません。SemaphoreOpened イベントが発生した以降 SemaphoreCapture メソッドを使ってセマフォーのアクセス権を確保してください。
SemaphoreClosed イベントが発生してセマフォー状態監視スレッドが終了するまで、このオブジェクトの MutexOpen、SemaphoreOpen、ConditionOpen、ProcessRun、ProcessExec、NotificationOpen の各メソッドを呼び出すことはできません。

SemaphoreCapture メソッド

[機　能] セマフォーのアクセス権を確保します。

[構　文] object.SemaphoreCapture
SemaphoreCapture 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	IPCOCX オブジェクトです。

[解　説] SemaphoreOpen メソッドでオープンしたセマフォーオブジェクトのアクセス権を確保します。
SemaphoreCapture メソッドは、SemaphoreOpen メソッドで生成したセマフォー状態監視スレッドに依頼してアクセス権の確保を行ないます。
スレッドがアクセス権の確保の監視に入る直前に SemaphoreCapturing イベントを発生します。その後、アクセス権を確保できたら、SemaphoreCaptured イベントが発生し、確保できないと、SemaphoreAborted イベントが発生します。
SemaphoreAborted イベントは、タイムアウトが発生したとき、または SemaphoreAbort メソッドが呼ばれたときに発生します。
確保できたかどうかを監視するのは SemaphoreOpen メソッドで生成したスレッドで行いますので、SemaphoreCapture メソッドがブロックすることはありません。
SemaphoreCaptured イベント発生までは、資源を利用することはできませんが、ほかの処理を行なうことは可能です。

SemaphoreRelease メソッド

［ 機　能 ］ セマフォーのアクセス権を解放します。

［ 構　文 ］ object.SemaphoreRelease
SemaphoreRelease 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	IPCOCX オブジェクトです。

［ 解　説 ］ SemaphoreOpen メソッドで生成したスレッドがアクセス権を保持していますので、このスレッドに対し、確保したアクセス権を解放するよう要請します。
解放されたなら、SemaphoreReleased イベントが発生します。
アクセス権の解放は、セマフォー状態監視スレッドの終了とは違います。
SemaphoreReleaseed イベントが発生したあとに、再度 SemaphoreCapture メソッドを使ってアクセス権の確保を試みることができます。
SemaphoreRelease メソッドを利用できるのは、SemaphoreSyncMode が True の状態でアクセス権を確保した場合だけです。

SemaphoreAbort メソッド

［ 機　能 ］ セマフォーのアクセス権の確保をあきらめます。

［ 構　文 ］ object.SemaphoreAbort
SemaphoreAbort 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	IPCOCX オブジェクトです。

［ 解　説 ］ SemaphoreCapture メソッドを実行しているスレッドに対し、監視を中断するよう要請します。
中断できたなら、SemaphoreAborted イベントを発生します。
監視の中断は、セマフォー状態監視スレッドが終了するわけではありませんので、SemaphoreAborted イベントが発生したあとに、再度 SemaphoreCapture メソッドを使ってアクセス権の確保を試みることができます。
終了させるには、SemaphoreClose メソッドを呼び出します。

SemaphoreClose メソッド

［ 機　能 ］ セマフォーの監視スレッドを終了します。

［ 構　文 ］ object.SemaphoreClose
SemaphoreClose 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	IPCOCX オブジェクトです。

［ 解　説 ］ セマフォー状態監視スレッドに対し、スレッドを終了するよう要請します。
終了できたなら、SemaphoreClosed イベントを発生します。
終了後は、SemaphoreCapture,SemaphoreRelease,SemaphoreAbort,SemaphoreClose メソッドを使用できません。

SemaphoreIncrease メソッド

[機　能] 指定したセマフォーの許可されるアクセス権を増加します。

[構　文] R = object.SemaphoreIncrease(SemaphoreName,Count)
object.SemaphoreIncrease SemaphoreName,Count
SemaphoreIncrease 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	IPCOCX オブジェクトです。
SemaphoreName	セマフォーの名前
Count	増加するアクセス可能なプロセス数(1 以上)

[データ型] 長整数(Long)

(R)	(戻 値)
>0	現在のセマフォーの残り数(Increase する直前)

[解　説] 指定したセマフォーに対し同時アクセスできるプロセス数を増加します。
SemahphoreOpen メソッドの maxcount で指定した数(1以上)までのプロセス数を指定することができます。
maxcount を 0 にして SemaphoreOpen している場合、必ずこのメソッドをどこかでだれかが呼び出すことになります。
また、SemaphoreSyncMode を False にした状態で、SemaphoreReCapture を使用して、与えられたアクセス権を消費していくようなコーディングをしている場合にも、このメソッドで、アクセス権を補っていくことになります。
指定されたセマフォーが存在しない場合、トラップ可能なエラーが発生します。

ConditionOpen メソッド

[機　能] コンディションをオープンします。

[構　文] R = object.ConditionOpen (ConditionName[,AutoOff][,OnStart])
object.ConditionOpen ConditionName[,AutoOff][,OnStart]
ConditionOpen 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	IPCOCX オブジェクトです。
ConditionName	コンディションの名前
AutoOff	On の状態の継続モードを指示 True :する False:しない(初期値)
OnStart	On の状態でコンディションを作成します。 True :On 状態で開始 False:Off 状態で開始(初期値)

[データ型] ブール型(Boolean)

(R)	(戻 値)
True	新規にコンディションオブジェクトを作成しました
False	既存のコンディションオブジェクトをオープンしました

［ 解　説 ］ ConditionName で指定された名前のコンディションオブジェクトをオープンます。
指定された名前のコンディションオブジェクトが無い場合は新規に作成します。
正常にオープンできると、コンディションの状態を監視するためのスレッド（コンディション状態監視スレッド）を生成します。コンディション状態監視スレッドが生成されると、ConditionOpened イベントが発生します。
ConditionOpen メソッドだけでは、コンディションが on の状態を検出できません。ConditionOpened イベントが発生した以降 ConditionFind メソッドを使ってコンディションが on の状態を検出してください。
ConditionClosed イベントが発生してコンディション状態監視スレッドが終了するまで、このオブジェクトの MutexOpen 、SemaphoreOpen 、ConditionOpen 、ProcessRun 、ProcessExec、NotificationOpen の各メソッドを呼び出すことはできません。

ConditionFind メソッド

［ 機　能 ］ コンディションが On の状態を検出します。

［ 構　文 ］ object.ConditionFind
ConditionFind 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	IPCOCX オブジェクトです。

［ 解　説 ］ ConditionFind メソッドは ConditionOpen メソッドで生成されたスレッドにコンディションの状態の監視を依頼します。依頼するとまず、ConditionWaiting イベントが発生します。つぎに監視を開始します。監視をしている間そのスレッドは停止しています。コンディションが On の状態を検出するとスレッドは再開し、ConditionFound イベントを発生します。

ConditionAbort メソッド

［ 機　能 ］ ConditionFind メソッドの実行をキャンセルします。

［ 構　文 ］ object.ConditionAbort
ConditionAbort 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	IPCOCX オブジェクトです。

［ 解　説 ］ ConditionAbort メソッドはコンディションの監視状態を解除します。
解除に成功すると、ConditionAborted イベントが発生します。
解除をしてもコンディション状態監視スレッドは終了しません。

ConditionClose メソッド

［ 機　能 ］ コンディション状態監視スレッドを終了します。

［ 構　文 ］ object.ConditionClose
ConditionClose 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	IPCOCX オブジェクトです。

［ 解　説 ］ コンディション状態監視スレッドに対し、スレッドを終了するよう要請します。
終了できたなら、ConditionClosed イベントを発生します。

ConditionOn メソッド

［ 機　能 ］ Condition の状態を On にします。

［ 構　文 ］ object.ConditionOn()
ConditionOn 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	IPCOCX オブジェクトです。

［ 解　説 ］ コンディションを On の状態にします。
ConditionFind メソッドを実行して、コンディションが On の状態になるのを待っているオブジェクトで ConditionFound イベントが発生します。

ConditionOff メソッド

［ 機　能 ］ Condition の状態を Off にする。

［ 構　文 ］ object.ConditonOff()
ConditionOff 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	IPCOCX オブジェクトです。

［ 解　説 ］ コンディションを Off の状態にします。
コンディションが Off の状態のとき ConditionFind メソッドを発行すると監視スレッドは On になるまで待ち状態になります。

ConditionOnAndOff メソッド

［ 機　能 ］ condition の状態を一瞬 On にした後すぐに Off にする。

［ 構　文 ］ object.ConditionOnAndOff()
ConditionOnAndOff 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	IPCOCX オブジェクトです。

［ 解　説 ］ コンディションを On の状態にし、その時 ConditionFind で待っているオブジェクトで ConditionFound イベントを発生させます。そしてその直後すぐに Off の状態にします。

NotificationOpen メソッド

［ 機　能 ］ ディレクトリの状態変更の検出を開始する。

［ 構　文 ］ object.NotificationOpen　Directory,WatchSubTree,WatchFilter
NotificationOpen 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	IPCOCX オブジェクトです。
Directory	監視するディレクトリパス名
WatchSubTree	サブディレクトリも監視対象にするかどうかのフラグ
WatchFilter	監視する項目

(WatchSubTree)	(意 味)
True	サブディレクトリも監視対象にする
False	サブディレクトリは監視対象にしない

(WatchFilter)	(値)	(検出する項目)
ipcFILE_NAME	=1	ファイルの追加、削除、名前変更
ipcDIR_NAME	=2	ディレクトリの追加、削除、名前変更
ipcATTRIBUTES	=4	属性の変更
ipcSIZE	=8	ファイルサイズの変更
ipcLAST_WRITE	=16	最終更新日時の変更
ipcLAST_ACCESS	=32	最終参照日時の変更(NT のみ)
ipcSECURITY	=256	セキュリティの変更(NT のみ)

[解　説] Directory で指定したディレクトリの状態変更を検出します。
検出項目は、WatchFilter で指定します。複数の項目を指定するにはそれらの値を OR したものを設定してください。
ディレクトリの状態変更を検出するために専用のスレッド（ディレクトリ変更監視スレッド）を生成します。ディレクトリ変更監視スレッドが生成されると、NotificationOpened イベントが発生します。
この状態では、ディレクトリの状態変更の検出は行われていますが、通知されません。通知されるには、NotificationFind メソッドを呼び出します。
このメソッドが呼び出されると、まず、NotificationWaiting イベントが発生します。
その後ディレクトリで監視する項目の変更を検出すると、NotificationFound イベントが発生します。さらに検出を続けるには、再度、NotificationFind メソッドを呼び出してください。
ディレクトリに変化があったことはわかりますが、どのファイルに変化があったかまでは通知されません。

NotificationFind メソッド

[機　能] ディレクトリの状態変更の通知を待ちます。

[構　文] object.NotificationFind
NotificationFind 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	IPCOCX オブジェクトです。

[解　説] ディレクトリ状態変更の通知を待ちます。
待ち状態になる直前に NotificationWaiting イベントが発生します。
状態変更が検出されると、NotificationFound イベントが発生します。

NotificationAbort メソッド

[機　能] ディレクトリの状態変更の通知処理を中断します。

[構　文] object.NotificationAbort
NotificationAbort 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	IPCOCX オブジェクトです。

[解　説] 中断に成功すると、NotificationAborted イベントが発生します。

NotificationClose メソッド

[機　能] ディレクトリの状態変更監視スレッドを終了します。

[構　文] object.NotificationClose
NotificationClose 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	IPCOCX オブジェクトです。

[解　説] ディレクトリの状態変更監視スレッドに対し監視を終了するよう要請します。
終了したなら NotificationClosed イベントを発生します。
終了後は、NotificationFind、NotificationAbort、NotificationClose メソッドは、使用できません。

SharedMemoryOpen メソッド

[機　能] 共有メモリをオープンします。

[構　文] R = object.SharedMemoryOpen(SharedMemoryName, DefaultData[,ReserveByteLen])
object.SharedMemoryOpen SharedMemoryName, DefaultData[,ReserveByteLen]
SharedMemoryOpen 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	IPCOCX オブジェクトです。
SharedMemoryName	共有メモリの名前
DefaultData	初期値
ReserveByteLen	確保バイト数(オプション)。

[データ型] ブール型(Boolean)

(R)	(戻 値)
True	新規に共有メモリを作成した。
False	既存既存の共有メモリをオープンした。

［解　説］ 共有メモリを SharedMemoryName という名前で作成します。
作成する共有メモリのサイズは、DefaultData を格納できます。
サイズまたは ReserveByteLen で指定されたサイズの大きい方です。
DefaultData の型と共有メモリのサイズは共有メモリに記憶され、他のプロセスがこの共有メモリに対し SharedMemoryOpen を発行したときに検査されます。
共有メモリに保存できるのは、文字列の配列およびバリアントの配列、オブジェクト、Empty 値以外のすべての型およびその配列です。
すでに作成されている場合は、それをオープンし、共有メモリの型、最大サイズが SharedMemoryOpen の引数の指定と一致するかを調べます。

SharedMemoryWrite メソッド

［機　能］ 共有メモリにデータを保存します。

［構　文］ object.SharedMemoryWrite Data
SharedMemoryWrite 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	IPCOCX オブジェクトです。
Data	保存するデータ

［解　説］ SharedMemoryOpen で作成された共有メモリにデータを保存します。
SharedMemoryOpen で作成されたデータ型と同じデータしか保存することはできません。データサイズもチェックされ、共有メモリよりも大きな値は保存できずにエラーになります。
共有メモリに保存できるのは、文字列の配列およびバリアントの配列、オブジェクト、Empty 値以外のすべての型およびその配列です。

SharedMemoryRead メソッド

［機　能］ 共有メモリからデータを読み出します。

［構　文］ R = object.SharedMemoryRead()
SharedMemoryRead 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	IPCOCX オブジェクトです。

[データ型] バリアント(Variant)

［解　説］ 共有メモリに保存されているデータを読み出し、VARIANT 変数に格納して返します。
配列でなければ VARIANT ではなく格納したときの型の変数で受けることも可能です。
配列データは必ず VARIANT 変数で受けてください。
但し、バイト配列は例外で、可変長バイト配列で受けることが可能です。
また、バイト配列は可変長文字列変数でも受けることが可能です。

SharedMemoryClose メソッド

[機　能] 共有メモリをクローズする。

[構　文] object.SharedMemoryClose

SharedMemoryClose 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	IPCOCX オブジェクトです。

[データ型] バリアント(Variant)

Win32ErrNumber メソッド

[機　能] メソッド内で発生した詳細エラーコードを示します。

[構　文] R = object.Win32ErrNumber()

Win32ErrNumber 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	IPCOCX オブジェクトです。

[データ型] 長整数型(Long)

[解　説] 0 以外の値の場合は、メソッド内で呼び出した Win32 API のエラーコードを示しています。

Win32ErrDescription メソッド

[機　能] メソッド内で発生した詳細エラーコードを示します。

[構　文] R = object.Win32ErrDescription()

Win32ErrDescriptionr 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	IPCOCX オブジェクトです。

[データ型] 文字列(String)

[解　説] メソッド内で呼び出した Win32 API のエラーコードの説明です。

【イベント】

ProcessOpened イベント

［機　能］ プロセス監視スレッドが開始したときに発生します。

［構　文］ Private Sub object_ProcessOpened()
ProcessOpened 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	IPCOCX オブジェクトです。

［解　説］ ProcessRun メソッドおよび ProcessShell メソッドの実行によりプロセスのプロセス監視スレッドが開始したときに発生します。

ProcessWaiting イベント

［機　能］ プロセスの終了を検出する直前に発生します。

［構　文］ Private Sub object_ProcessWaiting()
ProcessWaiting 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	IPCOCX オブジェクトです。

［解　説］ ProcessWait メソッドの実行によりプロセスの終了の検出を開始する直前に発生します。

ProcessDone イベント

［機　能］ 起動したプロセスが終了した時に発生します。

［構　文］ Private Sub object_ProcessDone(ByVal Code As Long)
ProcessDone 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	IPCOCX オブジェクトです。
Code	プロセスの終了コードです。

［解　説］ 起動したプロセスの終了を検出するとこのイベントが発生します。

ProcessAborted イベント

［機　能］ 起動したプロセスが終了した時に発生します。

［構　文］ Private Sub object_ProcessAborted(ByVal Reason As Long)
ProcessAborted 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	IPCOCX オブジェクトです。
Reason	中断の理由 1: タイムアウトによる中断 2: ProcessAbort メソッドによる中断

［解　説］ 起動したプロセスの終了を検出するのを中断すると発生します。

ProcessClosed イベント

［機　能］ プロセス監視スレッドが終了すると発生します。

［構　文］ Private Sub object_ProcessClosed(ByVal Code As Long, ByVal Msg As String)
ProcessClosed 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	IPCOCX オブジェクトです。
Code	終了コード 0:正常終了 その他:Win32 API エラーコード
Msg	Win32 API エラーメッセージ

［解　説］ プロセス監視スレッドが終了すると発生します。
プロセス監視スレッドが終了するのは、ProcessClose メソッドが実行されたときまたは待ち状態(WAIT 状態)でエラーになったとき、または、コントロールがアンロードされるときですが、コントロールがアンロードされるときは、このイベントは発生しません。

MutexOpened イベント

［機　能］ ミューテクス状態監視スレッドが生成されると発生します。

［構　文］ Private Sub object_MutexOpened()
MutexOpened 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	IPCOCX オブジェクトです。

［解　説］ MutexOpen メソッドの結果ミューテクス状態監視スレッドが生成されると発生します。

MutexLocking イベント

［機　能］ MutexLock メソッドによりミューテクスのアクセス権の確保を行なおうとしています。

［構　文］ Private Sub object_MutexLocking()
MutexLocking 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	IPCOCX オブジェクトです。

［解　説］ MutexLock メソッドを実行するとその直後に発生します。

MutexLocked イベント

［機　能］ MutexLock メソッドの結果、ミューテクスのアクセス権を確保できたことを通知します。

［構　文］ Private Sub object_MutexLocked()
MutexLocked 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	IPCOCX オブジェクトです。

［解　説］ このイベントのあと MutexUnlock メソッドを発行するまで、資源へのアクセス権を保持します。

MutexUnlocked イベント

［機　能］ ミューテクスのアクセス権がなくなったことを通知します。

［構　文］ Private Sub object_MutexUnlocked()
MutexUnlocked 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	IPCOCX オブジェクトです。

［解　説］ MutexUnock メソッドを実行するとその直後に発生します。

MutexClosed イベント

［機　能］ ミューテクス状態監視スレッドが終了すると発生します。

［構　文］ Private Sub object_MutexClosed(ByVal Code As Long, ByVal Msg As String)
MutexClosed 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	IPCOCX オブジェクトです。
Code	終了コード 0:正常終了 その他:Win32 API エラーコード
Msg	Win32 API エラーメッセージ

［解　説］ ミューテクス状態監視スレッドが終了すると発生します。
ミューテクス状態監視スレッドが終了するのは、MutexClose メソッドが実行されたときまたは待ち状態(WAIT 状態)でエラーになったとき、または、コントロールがアンロードされるときですが、コントロールがアンロードされるときは、このイベントは発生しません。

SemaphoreOpened イベント

［機　能］ SemaphoreOpen メソッドによりセマフォー状態監視スレッドが開始されたことを通知します。

［構　文］ Private Sub object_SemaphoreOpened()
SemaphoreOpened 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	IPCOCX オブジェクトです。

［解　説］ このイベントのあと SemaphoreCapture メソッドを発行することによりセマフォーのアクセス権を確保することができます。

SemaphoreCapturing イベント

［機　能］ SemaphoreCapture メソッドによりセマフォーを獲得しようとしています。

［構　文］ Private Sub object_SemaphoreCapturing()
SemaphoreCapturing 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	IPCOCX オブジェクトです。

［解　説］ このイベントのあとセマフォー状態監視スレッドはセマフォーのアクセス権を確保するまで停止します。

SemaphoreCaptured イベント

［機　能］ SemaphoreCapture メソッドの結果アクセス権を獲得できたことを通知します。

［構　文］ Private Sub object_SemaphoreCaptured()
SemaphoreCaptured 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	IPCOCX オブジェクトです。

［解　説］ このイベントのあと SemaphoreRelease メソッドを発行するまで、資源へのアクセス権を保持します。

SemaphoreReleased イベント

［機　能］ セマフォーのアクセス権がなくなったことを通知します。

［構　文］ Private Sub object_SemaphoreReleased()
SemaphoreReleased 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	IPCOCX オブジェクトです。

［解　説］ SemaphoreRelease メソッドを発行するとこのイベントが発生します。

SemaphoreClosed イベント

［機　能］ セマフォー状態監視スレッドが終了すると発生します。

［構　文］ Private Sub object_SemaphoreClosed(ByVal Code As Long, ByVal Msg As String)
SemaphoreClosed 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	IPCOCX オブジェクトです。

Code	終了コード 0:正常終了 その他:Win32 API エラーコード
Msg	Win32 API エラーメッセージ

[解　説] セマフォー状態監視スレッドが終了すると発生します。
セマフォー状態監視スレッドが終了するのは、SemaphoreClose メソッドが実行されたときまたは待ち状態(WAIT 状態)でエラーになったとき、または、コントロールがアンロードされるときですが、コントロールがアンロードされるときは、このイベントは発生しません。

ConditionOpened イベント

[機　能] ConditionOpen メソッドによりコンディション状態監視スレッドが開始されたことを通知します。

[構　文] Private Sub object_ConditionOpened()
ConditionOpen 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	IPCOCX オブジェクトです。

[解　説] このイベントのあとConditionFindメソッドを発行することによりコンディションがOnの状態を検出することができます。

ConditionWaiting イベント

[機　能] ConditionFind メソッドによりコンディションが On の状態を検出しようとしています。

[構　文] Private Sub object_ConditionWaiting()
ConditionWaiting 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	IPCOCX オブジェクトです。

[解　説] このイベントのあとコンディション状態監視スレッドはコンディションが On の状態を検出するまで停止します。

ConditionFound イベント

[機　能] ConditionFind メソッドの結果、コンディションが On の状態を検出できたことを通知します。

[構　文] Private Sub object_ConditionFound()
ConditionFound 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	IPCOCX オブジェクトです。

[解　説] Condition が On の状態が検出するとこのイベントが発生します。

ConditionClosed イベント

［機　能］ コンディション状態監視スレッドが終了すると発生します。

［構　文］ Private Sub object_ConditionClosed(ByVal Code As Long, ByVal Msg As String)
ConditionClosed 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	IPCOCX オブジェクトです。
Code	終了コード 0:正常終了 その他:Win32 API エラーコード
Msg	Win32 API エラーメッセージ

［解　説］ コンディション状態監視スレッドが終了すると発生します。
コンディション状態監視スレッドが終了するのは、ConditionClose メソッドが実行されたときまたは待ち状態(WAIT 状態)でエラーになったとき、または、コントロールがアンロードされるときですが、コントロールがアンロードされるときは、このイベントは発生しません。

NotificationOpened イベント

［機　能］ NotificationOpen メソッドによりディレクトリ変更監視スレッドが開始されたことを通知します。

［構　文］ Private Sub object_NotificationOpened()
NotificationOpened 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	IPCOCX オブジェクトです。

［解　説］ このイベントの後、NotificationFind メソッドを発行するとディレクトリの変更を通知するようになります。

NotificationWaiting イベント

［機　能］ NotificationFind メソッドが発行された直後に発生します。

［構　文］ Private Sub object_NotificationWaiting()
NotificationWaiting 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	IPCOCX オブジェクトです。

NotificationFound イベント

[機　能] ディレクトリの変更を検出した時に発生します。

[構　文] Private Sub object_NotificationFound()
NotificationFound 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	IPCOCX オブジェクトです。

[解 説] NotificatioOpen メソッドで指定したディレクトリの変更があるとき、NotificationFind メソッドが実行されるとこのイベントが発生します。

NotificationClosed イベント

[機　能] ディレクトリ変更監視スレッドが終了すると発生します。

[構　文] Private Sub object_NotificationClosed(ByVal Code As Long, ByVal Msg As String)
NotificationClosed 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	IPCOCX オブジェクトです。
Code	終了コード 0:正常終了 その他:Win32 API エラーコード
Msg	Win32 API エラーメッセージ

[解 説] ディレクトリ変更監視スレッドが終了すると発生します。
ディレクトリ変更監視スレッドが終了するのは、NotificationClose メソッドが実行されたときまたは待ち状態(WAIT 状態)でエラーになったとき、または、コントロールがアンロードされるときですが、コントロールがアンロードされるときは、このイベントは発生しません。

【発生しうるエラーコード】

メソッドまたはプロパティ	発生しうるトラップ可能なエラーコード							
ConditionAbort	29922							
ConditionFind	29922							
ConditionOff	29922							
ConditionOn	29922							
ConditionOnAndOff	29922							
ConditionOpen	29901	29907	29912	29920				
MutexAbort	29922							
MutexLock	29922							
MutexOpne	29901	29912	29920					
MutexUnlock	29922							
NotificationAbort	29922							
NotificationFind	29922							
NotificationOpen	29901	29911	29912	29914				
ProcessAbort	29922							
ProcessExec	29901	29902	29911	29912	29919			
ProcessRun	29901	29902	29911	29912	29919			
ProcessWait	29922							
SemaphoreAbort	29922							
SemaphoreCapture	29922							
SemaphoreIncrease	29910	29913						
SemaphoreOpen	29901	29912	29913					
SemaphoreRelease	29922							
SharedMemoryOpen	29901	29905	29907	29908	29915	29916	29917	29924
SharedMemoryRead	29904	29918	29924					
SharedMemoryWrite	29904	29907	29908	29915	29917	29924		
ShowStyle	29910							

【エラーの内容】

エラーコード	内　　容
29900	win32API 呼び出しでエラーが発生致しました。
29901	同時に実行できないメソッドを実行中です。
29902	既存のプロセスがあります。プロセスは育成されませんでした。
29904	先にオープンして下さい。
29905	引数は省略できません
29907	引数の方が正しくありません。
29908	共有メモリの型または大きさが一致しません。
29910	引数の範囲が正しくありません。
29911	引数が正しくありません。
29912	待ちスレッドを作成できません。
29913	セマフォーを作成できません。
29914	ディレクトリ監視に失敗しました。
29915	この型の配列は指定できません。
29916	共有メモリを作成できません。
29917	引数からデータを取り出すことが出来ません。
29918	共有メモリからデータを取り出すことが出来ません。
29919	プロセスを作成できません。
29920	ミューテックスを作成できません。
29921	ミューテックスがありません。
29922	操作が正しくありません。
29923	コンディションを作成できません。
29924	タイムアウトしました。
その他	エラーが発生しました。

索引

IPCOCX マニュアル

2000 年 4 月 7 日 発行　　第 1 版
発行所　株式会社ウィル
住所 神奈川県横浜市保土ヶ谷区西久保 15
グランディシンヤ 302
〒240－0022
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