



Caché アクティベートの使 用法

Version 5.1

2006-03-14

Caché アクティベートの使用法

Caché Version 5.1 2006-03-14

Copyright © 2006 InterSystems Corporation.

All rights reserved.

このドキュメントは、Sun Microsystems、RenderX Inc.、アドビ システムズ および ワールドワイド・ウェブ・コンソーシアム (www.w3c.org) のツールと情報を使用して、Adobe Portable Document Format (PDF) で作成およびフォーマットされました。主要ドキュメント開発ツールは、InterSystemsが構築したCaché と Javaを使用した特別目的のXML処理アプリケーションです。



Caché 製品とロゴは InterSystems Corporation の登録商標です。



Ensemble 製品とロゴは InterSystems Corporation の登録商標です。



InterSystems という名前とロゴは InterSystems Corporation の登録商標です

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を稼動および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

Caché および InterSystems Caché、Caché SQL、Caché ObjectScript および Caché Object は、インターシステムズ社の商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems ワールドワイド カスタマサポート

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

目次

1 はじめに	1
1.1 アーキテクチャ	2
1.2 ActiveX / COM の概要	2
1.2.1 COM オブジェクトとは?	2
1.2.2 COM インタフェース	2
1.2.3 IDispatch インタフェース	3
1.2.4 タイプ・ライブラリ	3
2 Caché アクティベートの使用法	5
2.1 Caché アクティベート・ウィザード	5
2.2 生成されたラップ・クラスの使用法	7
2.2.1 例 : プロパティへのアクセス	7
2.2.2 例 : 列挙 COM インタフェース	7
2.2.3 プロパティに関する特別な考慮事項	8
2.3 例外処理	9
2.3.1 例 : 例外処理	9
2.4 %Activate.IDispatch および %Activate.GenericObject	10
2.4.1 例 : CreateObject の使用法	10
2.5 モニカ	10
2.5.1 例 : GetObject の使用法	11
2.6 Become メソッド	11
2.7 イベント	11
2.7.1 例 : COM イベントの使用法	12

1

はじめに

Caché アプリケーションは、Caché アクティベートによって Caché サーバ内から ActiveX (COM としても知られています) コンポーネントと簡単に相互運用することができます。ラップ・クラスを使用すると、ActiveX コンポーネントは Caché オブジェクト・クラスのインスタンスとして使用可能になり、他のクラスと同じ方法で利用できます。Caché アクティベートは、外部 COM オブジェクトをインスタンス化し、これをネイティブの Caché オブジェクトであるかのように操作する機能を提供します。

注釈: このドキュメントの中では、“ActiveX” と “COM” という用語は同じ意味で使用されています。

Caché アクティベートは、ActiveX (現在は Microsoft Windows) をサポートするプラットフォームでのみ利用できます。

Caché アクティベートは、以下のように動作します。

1. Caché アクティベート・ウィザードを使用して、1 つ以上のラップ・クラスを作成することができます。ラップ・クラスは、ActiveX コンポーネントのインタフェースに対応するメソッドを提供する Caché クラスです。
2. Caché アプリケーション内で、ActiveX ラップ・クラスのインスタンスを生成できます。Caché アクティベートは、適切な ActiveX コンポーネントのインスタンスを、同じプロセス内に透過的に生成します。ユーザがラップ・クラスのメソッドを呼び出すと、それらを適切な ActiveX インタフェースのメソッドへ自動的に送信します。

Caché 内で ActiveX コンポーネントを使用する際には、注意が必要です。Caché は、アプリケーション・コードが安全に実行できる環境を提供するように設計されています。すべての Caché サーバ・プロセスは、他のサービス・プロセスから独立して Caché 仮想マシンのインスタンスを実行するため、アプリケーション・エラーを安全に処理することができます。しかし、残念ながら ActiveX は安全なテクノロジーではありません。ActiveX を誤って使用したり、不適切に実装された ActiveX コンポーネントを使用すると、メモリ・リークや予想できないアプリケーションのクラッシュが起こる可能性があります。重要なアプリケーション内で ActiveX コンポーネントを使用する場合は、コンポーネントのインタ

フェースを正しく使用すること、コンポーネントは徹底的にテストすることなどに細心の注意を払ってください。ご使用のアプリケーション内で使用する前に、Visual Basic などのツールでコンポーネントをテストすることをお勧めします。

1.1 アーキテクチャ

Caché アクティベートは、以下のコンポーネントからなります。

- ・ Caché アクティベート・ウィザード : Caché サーバで使用できる ActiveX コンポーネントのリストからコンポーネントを選択し、その Caché ラップ・クラスを自動的に生成することができます。Caché アクティベート・ウィザードは、Caché スタジオの開発環境の【ツール】メニューからアクセスできます。
- ・ Caché アクティベート・クラス階層 : ActiveX と通信するために、生成されたラップ・クラスが使用するヘルパー・クラスです。
- ・ Caché ActiveX ゲートウェイ : Caché プロセスが ActiveX コンポーネントで処理 (ロード、メソッド呼び出し、リリース) を実行するために使用し、ロードする共有ライブラリ (DLL) です。

1.2 ActiveX / COM の概要

以下は、Caché に関連する ActiveX / COM コンポーネント・アーキテクチャの簡単な概要です。ご使用のアプリケーション内で ActiveX を活用するには、それに関する多くの書籍などを調べてください。

1.2.1 COM オブジェクトとは？

COM オブジェクトは、COM 仕様に従うコードで、クライアント・プログラムで使用される 1 つ以上のサービスを提供します。COM オブジェクトのうち、自動化という概念をサポートする特定のクラスは、VisualBasic、Delphi、Caché などの高レベルのプログラミング言語から簡単にアクセスできるように設計されています。そのようなオートメーション・オブジェクトはダイナミック・リンク・ライブラリとして実装され、テキスト文字列の暗号化などの単純な関数を提供したり、Microsoft Excel や Microsoft Word のように、さまざまな異なるサービスを提供する本格的なアプリケーションにもなります。

1.2.2 COM インタフェース

COM オブジェクトは、機能をインタフェースとして公開します。インタフェースは、ある特定の機能をカプセル化するメソッドとプロパティの集合です。例えば、ワード・プロセッシング・オブジェクトは、出力インタフェースとスペル・チェック・インタフェースを提供します。COM オブジェクトの各実装は、ク

ラス ID の形式の一意の識別子を持ち、公開する各インタフェースも、インタフェース ID と呼ばれる一意の識別子を持ちます。一旦、特定のオブジェクトのクラス ID と要求されたインタフェースのインタフェース ID が判別されると、クライアント・アプリケーションは COM オブジェクトをインスタンス化し、要求されたインタフェースが提供するサービスを利用します。取り決めとして、インタフェースの名前を書く際には、先頭に大文字の “I” を付けます。例えば、SpellCheck インタフェースは **ISpellCheck** になります。

1.2.3 IDispatch インタフェース

異なるプログラミング言語は、バイナリ・レベルでは互換性を持たない異なる内部データ型を持ちます。例えば、Caché ローカル変数は、VisualBasic 文字列あるいは C++ 文字列のローカル変数とはまったく異なる実装を持ちます。したがって、C++ データ型から Caché 変数へ変換する、あるいはその逆に変換する必要があるため、1 つの言語から別の言語で記述されたオブジェクトの呼び出しが困難になります。この問題を解決し、異なるプログラミング言語との通信を可能にするために、VARIANT データ型の概念や **IDispatch** インタフェースが開発されました。

IDispatch は、メソッドまたはパラメータの名前を指定して適切な引数を渡すことにより、外部 COM オブジェクトのメソッドを呼び出したり、プロパティにアクセスする機能を提供します。引数は、Windows オペレーティング・システムが最も単純な形態としてサポートする標準的なデータ型である VARIANT 型で表されます。COM オートメーションの使用をサポートするすべてのプログラミング言語は、この標準型を “認識できます”。

COM オブジェクトを生成し、**IDispatch** インタフェースを要求することにより、Caché のようなクライアント・プログラムまたは言語は、オブジェクトが公開する機能に簡単にアクセスできます。

IDispatch は COM オートメーション・オブジェクトにアクセスする汎用的な方法を提供しますが、本来は、プログラミング言語が内部的に利用し、その言語特有の構成要素を経由して、COM オブジェクト・サービスを提供するためのテクニックとして意図したものです。つまり、高レベルの言語は **IDispatch** 呼び出しの詳細を抽象化し、外部オブジェクトを簡単に使用できるようなプログラミング言語構成を提供します。理想としては、そのような COM オブジェクトが、ネイティブのオブジェクトであるかのようにプログラミング環境で動作することです。Caché アクティベートでこれを実現するためには、COM オブジェクト・タイプ・ライブラリに含まれる情報を活用することが重要です。

1.2.4 タイプ・ライブラリ

すべてではありませんが、ほとんどの COM オートメーション・オブジェクトは、タイプやメソッド、プロパティの詳細などのメタ・データを、タイプ・ライブラリの形式で公開しています。タイプ・ライブラリは、バイナリ・リソースとして実行可能なファイル内で .DLL (ダイナミック・リンク・ライブラリ) に結合することも、.tlb などの拡張子を付けて別のファイルに存在することもできます。タイプ・ライブラリ内では、各オブジェクトはクラス ID として識別され、CoClass として認識されます。CoClass は、既定インタフェースと呼ばれる **IDispatch** 由来のインタフェースを 1 つ公開することができます (ソース・インタフェースと呼ばれる別のインタフェースがある場合もありますが、これは直接呼び出すことはできないので、今のところ無視しても問題ありません)。IDispatch 由来のインタフェースを実装しないオブジェクトもあり、これらは **IDispatch** ベースのメカニズムから呼び出すことができません。

Caché アクティベートは、その情報を読み込んで解読し、定義されたメソッドやプロパティを公開した Caché クラスを生成します。タイプ・ライブラリに含まれるメタデータのタイプ・ライブラリには、1 つ以上の CoClass オブジェクトや多数の IDispatch 由来のインタフェース定義を含む可能性があります。CoClass は、既定インタフェースとしては 1 つを超える IDispatch 由来のインタフェースを公開しません。しかし、インタフェースを返す、あるいはインタフェースとしてタイプされるメソッドやプロパティを定義することは自由なので、多数のインタフェースがある可能性があります。実際、1 つの CoClass (オブジェクト) が機能豊富なオブジェクト・モデルを定義する場合に、よくそういうことがあります。例えば、ワード・プロセッシングについて考えてみます。これは、AboutBox や Exit などのメソッドを持つ **IApplication** の既定のインタフェースを提供することができます。また、**Documents** と呼ばれるプロパティとして、ドキュメントの集合 (IDocuments*) を提供することも可能です。

注釈: 多くの COM インタフェースは非常に複雑で、何百ものメソッドを含むものや、追加の COM オブジェクトをパラメータとして多数使用するものがあります。ご使用のアプリケーションが特定のインタフェースの小規模なサブセットを利用するだけであれば、ラップ COM コンポーネント (例えば Visual Basic) を使用して構築するという方法をお勧めします。これにより、ユーザが本当に必要なインタフェースのみを公開し、オリジナルの COM コンポーネントのこれらのインタフェースに要求を渡します。

2

Caché アクティベートの使用法

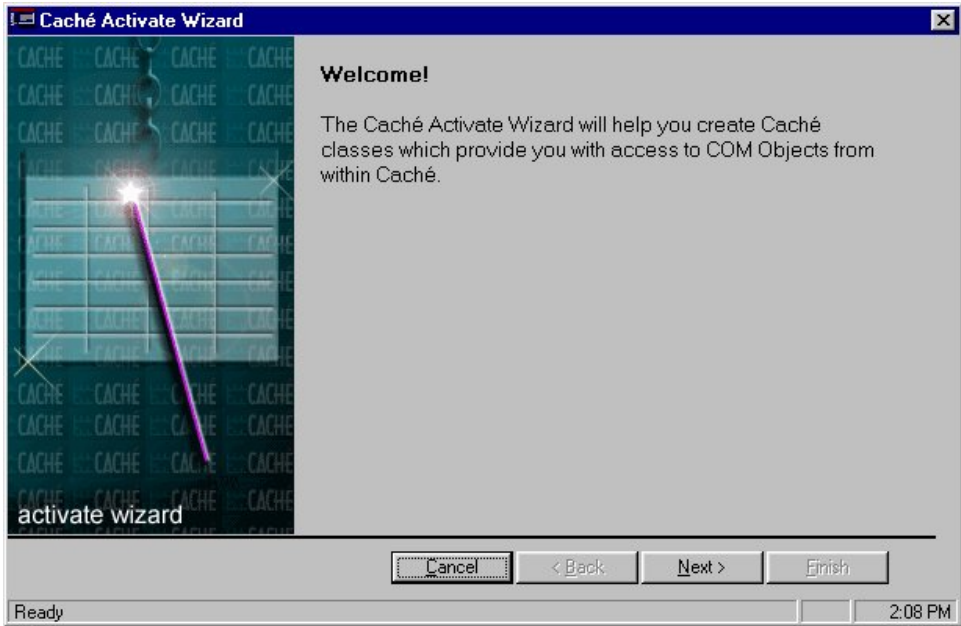
この章では、ActiveX コンポーネントの Caché ラップ・クラスを生成する方法、およびアプリケーションをそのラップ・クラスで使用方法について説明します。

2.1 Caché アクティベート・ウィザード

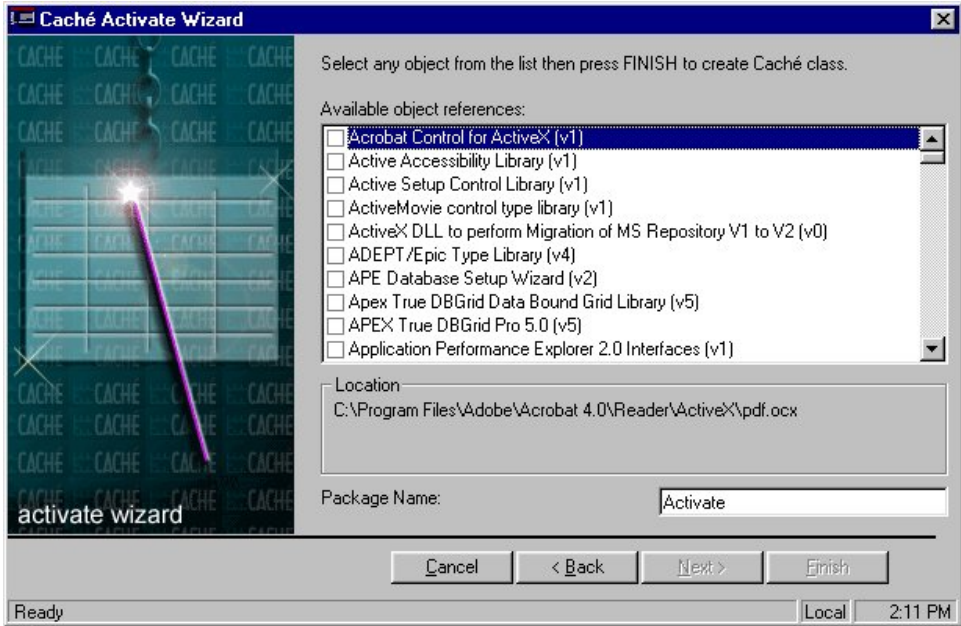
Caché アクティベート・ウィザードは、ActiveX インタフェースの任意のセットに対し1つ以上のラップ・クラスを自動的に生成します。

ウィザード使用法

1. (Caché キューブ・メニューから) Caché スタジオ を起動します。
2. 使用しているアプリケーションに新規のプロジェクトを作成します。
3. [ツール] メニューから [Activate ウィザード] を選択し、Caché Activate ウィザードを開始します。



- 4. ウィザードは、利用できる COM インタフェースのリストを表示します (これらは、スタジオを起動しているマシン上ではなく、Caché サーバ上で利用できるインタフェースです)。



1 つ以上のインタフェースを選択し、生成されるクラスに使用するパッケージ名を入力し、[次へ] ボタンをクリックします。

5. ウィザードは、指定されたパッケージ内にラップ・クラスを自動的に生成し、これをコンパイルします。

2.2 生成されたラップ・クラスの使用法

Caché で生成されたクラスは、COM オブジェクトのプロキシ・クラスであると考えられます。一旦、クラスが生成されコンパイルされると、Caché アプリケーション内でこれを使用できます。

例えば、アクティベート・ウィザードを使用して、システム・リソースに関する情報を提供する Microsoft SysInfo Control のラップ・クラスを生成することができます。

- ・ **Activate.SysInfoLib.ISysInfo** – **ISysInfo** インタフェースが提供するメソッドやプロパティを定義する抽象インタフェース・クラス。これは、インスタンス化することはできません。

BatteryLifePercent と呼ばれる計算プロパティと、そのプロパティに対応する `get` メソッドと `set` メソッドを含みます。

- ・ **Activate.SysInfoLib.SysInfo** – **ISysInfo** クラスから継承する具象クラス。これに含まれるコードは、外部 COM オブジェクトを見つけてインスタンス化し、そのオブジェクトへの“接続”を維持します。ユーザは、この具象クラスを使用して外部オブジェクトを操作します。オブジェクトをクローズすると、外部 COM オブジェクトもクローズ (解除) されます。

2.2.1 例 : プロパティへのアクセス

ここでは、**SysInfo** ラップ・オブジェクトを使用して、ラップトップ・コンピュータに残っているバッテリーの割合を取得します。

```
Set obj = ##Class(Activate.SysInfoLib.SysInfo).%New()
Write obj.BatteryLifePercent, !
Do obj.%Close()
```

このオブジェクトは、Caché 内のあらゆるオブジェクトと同じ方法で生成されます。**BatteryLifePercent** プロパティが書き出され、最後にオブジェクトをクローズします。単純で簡単です。

2.2.2 例 : 列挙 COM インタフェース

Caché アクティベート・ウィザードは、(**CacheSys¥Bin** ディレクトリにインストールされている) **TL.dll** と呼ばれる COM オブジェクトを使用して、Caché サーバ上でタイプ・ライブラリを列挙します。このオブジェクトから生成される Caché クラスは、**%Activate.TLLib** パッケージに事前にロードされます。

これらのクラスは、以下から構成されます。

- ・ `%Activate.TLLib.IUtils - ILibraries` タイプの 1 つのプロパティおよびライブラリを持つ抽象インタフェース・クラス。このプロパティを使用して、システムのタイプ・ライブラリを列挙する `ILibraries` インタフェースを取得します。
- ・ `%Activate.TLLib.ILibraries - Count` プロパティおよび `Item` プロパティを公開する抽象インタフェース・クラス。これらのプロパティを使用して、システムのタイプ・ライブラリを列挙します。
- ・ `%Activate.TLLib.Utils - IUtils` インタフェースを表す具象サブクラス。このクラスをインスタンス化して、`Libraries` プロパティにアクセスします。

以下は、これらのクラスを使用してシステムのタイプ・ライブラリを列挙する Caché ObjectScript メソッドの例です。 `Utils` クラスの具象インスタンスを生成し、`objlibs` プロパティを取得します。 Caché は現在、計算インデックス・プロパティをサポートしないため、`Item` プロパティは `ItemGet` メソッド経由で呼び出されることに注意してください。

```
Class MyApp.ActivateTest
{
// ...

/// Demonstrate COM object Access and provide type library enumeration ;
ClassMethod ListTypeLibs() {
    Set objUtils = ""
    Set objLibs = ""
    Set $ZT = "tlerr"
    Set objUtils = ##class(%Activate.TLLib.Utils).%New()
    Set objLibs = objUtils.Libraries
    For i = 1:1:objLibs.Count {
        Set tld = objLibs.ItemGet(i)
        // tld is a | delimited string
        Write !, $Piece(tld,"|"), !, $Piece(tld,"|",2), !, $Piece(tld,"|",3), !!
    }

xit          ; Exit point
    If objLibs="" Do objLibs.%Close()
    If objUtils="" Do objUtils.%Close()
Quit

tlerr        ; Exception handler
    Set $ZT = ""
    Goto xit
}
}
```

2.2.3 プロパティに関する特別な考慮事項

上記の例のコードにもあるように、COM ではパラメータを持つ プロパティ があります。さらに、“既定プロパティ”を持つオブジェクトもあります。つまり、プロパティ名を明示的に指定しないで、そのプロパティを参照することができます。

例えば、(上記の例にあるように) コレクションは常に `Count` プロパティと `Item` プロパティを持ちます。ここで `Item` プロパティは (明らかに) メソッドではありませんが、引数を持ちます。 `Item` プロパティは、コレクションの既定プロパティである場合もあります。したがって、例えば Microsoft Excel にワークブックの集合がある場合、Visual Basic で名前を指定したワークブックに以下のようにアクセスできます。

```
Application.Workbooks("Sheet1")
```

“Sheet1” という **Item** にアクセスしているにもかかわらず、**Item** は明示的に参照されていません。コードが実際に実行しているのは、以下の呼び出しです。

```
Application.Workbooks.Item("Sheet1")
```

Caché は括弧の有無によって、メソッド呼び出しとプロパティの参照を区別します。つまり、“person.Name” はプロパティで、“person.RaiseSalary()” はメソッドです。これにより、既定プロパティが扱いにくくなります。Caché は Visual Basic とは異なり、既定パラメータを定義する機能も、パラメータを渡す間にプロパティを参照する機能も備えていないためです。したがって、Caché は VB 構文 `Workbooks("Sheet1")` の、Item プロパティへの暗黙参照をサポートしません。また、Caché では Caché インタープリタは Item がメソッドであると認識するため、Item がプロパティである `Workbooks.Item("Sheet1")` もサポートしません。しかし、`Workbooks.ItemGet("Sheet1")` 呼び出しにより、この問題に対処することができます。ItemGet は Item プロパティを取得するメソッドです。

2.3 例外処理

COM オブジェクトは、メソッド呼び出し、あるいはプロパティの設定や取得などの操作の結果として、例外を引き起こすことがあります。例外が発生すると、例外は ZTrap 機能を通して Caché に伝搬されます。呼び出しコードは、エラー・コード <ZACTX> というエラーを受け取り、ローカル変数 `%objlasterror` はエラーの完全な詳細テキストを含みます。プログラムはこのエラーを考慮し、適切に対応してください。

2.3.1 例 : 例外処理

以下は、FTP によってファイルを取得する COM オブジェクトの使用例です。オブジェクトを生成し、`CurrentDirectory` プロパティを問い合わせます。FTP 接続が確立する前に現在のディレクトリを指定しようとするとう無効になるため、COM オブジェクトは例外を発生します。Caché コマンド行 (ターミナル・セッション) からこれを試行します。

```
Set obj = ##Class(Activate.RETRIEVERLib.FtpRetriever).%New()
Write obj.CurrentDirectory
```

この場合、上記はエラーを発生します。

```
<ZACTX>CurrentDirectoryGet+4^Activate.RETRIEVERLib.FtpRetriever.1
```

<ZATCX> エラーに関連するエラー・コードは、ローカル変数 `%objlasterror` 内にあります。以下の `system.OBJ.DisplayError` を呼び出して、エラー・メッセージの完全なテキストを取得することもできます。

```
Do system.OBJ.DisplayError(%objlasterror)
```

その結果、以下が出力されます。

```
ERROR #1101: Com Exception: '-2147220888 Ftp Retriever Connection must  
be established before attempting this operation'
```

2.4 %Activate.IDispatch および %Activate.GenericObject

COM オブジェクトにはタイプ・ライブラリがないものや、COM オブジェクトのメソッド・タイプ、プロパティ・タイプの返りタイプが **IDispatch** インタフェースであるものもあります。そのようなオブジェクトに対するメソッドを呼び出し、プロパティにアクセスする方法について、以下で説明します。

Caché アクティベートはこの問題を解決するために、**%Activate.IDispatch** と **%Activate.GenericObject** という 2 つのクラスを提供します。

多くの COM オブジェクトは、いわゆる “ProgId” により識別されます。“ProgId” は、通常ライブラリ名、またはオブジェクト名からなる文字列で、オブジェクトの識別に使用します。Visual Basic には `CreateObject` 呼び出しがあります。これは ProgId を取得し、オブジェクトの操作に使用するオブジェクト参照を返します。Caché は、**%Activate.GenericObject** クラスのクラス・メソッドとして `CreateObject` メソッドも提供します。これは、以下のように使用します。

2.4.1 例 :CreateObject の使用法

上記と同じ Microsoft SysInfo オブジェクトを使用して、ProgId 経由でオブジェクトをインスタンス化します。この方法でインスタンス化すると、このオブジェクトに対するタイプ情報がない (汎用オブジェクトである) ため、**IDispatch** インタフェースから汎用メソッドを呼び出す必要があります。このインタフェースは、名前によってプロパティを取得、設定しメソッドを呼び出します。

```
Set obj = ##Class(%Activate.GenericObject).CreateObject("SYSINFO.SysInfo")  
Write obj.GetProperty("BatteryLifePercent")  
Do obj.%Close()
```

2.5 モニカ

COM は ProgId の代わりにモニカを使用して、オブジェクトを間接的にインスタンス化する別の方法も提供します。Visual Basic は `GetObject` 呼び出しを提供します。これはモニカを取得し、オブジェクトの操作に使用するオブジェクト参照を返します。Caché は、**%Activate.GenericObject** クラスのクラス・メソッドとして `GetObject` メソッドを提供します。これは、以下のように使用します。

2.5.1 例 :GetObject の使用法

以下は、アクティブ・ディレクトリ・サービスの LDAP プロトコルにアクセスするモニカです。現在のドメインのユーザを表すノードの集合への参照を返すために使用します。ユーザの数が書き出され、オブジェクトをクローズします。

```
Set obj = ##Class(%Activate.GenericObject).GetObject("LDAP://CN=USERS")
Write obj.Count()
Do obj.%Close()
```

2.6 Become メソッド

タイプ・ライブラリは、汎用 **IDispatch** インタフェースの返りタイプを持つメソッドやプロパティを指定する場合があります。ユーザが受け取るのは **%Activate.IDispatch** のインスタンスで、その上ではプロパティの取得や設定、メソッドの呼び出しのために (GetProperty などの) 汎用メソッドを使用する必要があるため、非常に不便です。(ドキュメントやその他から) 本当に必要なインタフェースが分かっている場合は、**%Activate.IDispatch** オブジェクトのインスタンスの **Become** メソッドを呼び出し、新規の (現在タイプされている) インタフェースを取得することができます。Become メソッドは、引数としてクラス名を持ちます。事実上、**%Activate.IDispatch** が、ユーザがメソッドに渡したクラス名のインスタンスになります。呼び出されたオブジェクトが新規にタイプされたインタフェースをサポートしていない場合、Become は例外を返します。

2.7 イベント

COM コンポーネントには、メソッドの処理中にイベントを引き起こす機能を持つものがあります。そのイベントは、イベント、あるいは任意の名前の “ソース” ・インタフェースにまとめられます。例えば、**MyClass** という COM オブジェクトを与えられた場合、インタフェースは “MyClassEvents”、あるいは Visual Basic で生成された COM オブジェクトならば “_MyClass” と呼ばれます。

Caché アクティベートは、**%Activate.RegisterEvents** と **%Activate.HandleEvents** という 2 つのクラスを通してイベント処理を提供します。COM オブジェクトがイベントを発生する場合には、生成された Caché クラスは **%Activate.RegisterEvents** インタフェース・クラスを継承します。これは、**%RegisterHandler** と **%UnRegisterHandler** の 2 つのメソッドを追加します。通常の COM オブジェクトのプロキシ・クラスに加え、Event インタフェースを表す別のクラスが生成されます。これは **%Activate.HandleEvents** を継承し、event インタフェースにより定義される特定のイベントを処理するメソッドや **%Advise** および **%UnAdvise** メソッドを実装します。

2.7.1 例 :COM イベントの使用法

以下で例を使用して、わかりやすく説明します。FTP 転送を実行する仮想 COM オブジェクトを持つとします。そのオブジェクトは、Connect、Close、Download などのメソッドの実装に加え、BytesTransferred という 1 つのメソッドを表す **Event** インタフェースを実装します。接続に成功しダウンロードが開始すると、FTP オブジェクトは、1 キロバイトのデータをダウンロードするたびに“BytesTransferred” イベントを引き起こします。そのイベントは整数の Bytes と、参照により渡されるブーリアン値の Cancel という 2 つのパラメータを持つ BytesTransferred メソッドにより表されます。イベントが起こると、BytesTransferred メソッドが呼び出され、引数 Bytes と Cancel の現在値を渡します。その後、これらの値を使用して処理を実行します。通常、Bytes 引数はユーザ・インタフェースを通して表示されます。Cancel 引数は参照により渡されるため、その値を設定してイベントを起こした COM オブジェクトに返すことができます。このインスタンスで、Cancel に真 (COM では -1) を設定すると、現在の処理を中断し Download への呼び出しをすぐに返すように、COM オブジェクトに指示されます。通常通りにダウンロードが完了すると、Download への呼び出しは呼び出し側にコントロールを返し、それ以上イベントは起こりません。Caché では、FTP COM オブジェクトは生成されたクラス **Activate.SomeLibrary.FTP** やクラス **Activate.SomeLibrary.FTPEvents** によるイベント・インタフェースなどにより表されます。

この例は、以下のようになります。まず、FTP オブジェクトのインスタンスが生成されます。

```
Set FTP = ##Class(Activate.SomeLibrary.FTP).%New()
```

イベントを処理するために、イベント・ハンドラのインスタンスを生成します。

```
Set FTPHandler = ##Class(Activate.SomeLibrary.FTPEvents).%New()
```

イベントが処理される前に、イベント・ハンドラは実際にイベントを引き起こすオブジェクトに登録する必要があります。したがって、以下を呼び出します。

```
Do FTP.%RegisterHandler(FTPHandler)
```

次に接続し、ダウンロードします。

```
Do FTP.Connect("ftp.intersys.com")
Do FTP.Download("/public/somefile.txt")
```

ダウンロード中に、**Activate.SomeLibrary.FTPEvents** クラスで以下のメソッドが呼び出されます。

```
Class MyApp.Test
{
//...

Method BytesTransferred(Bytes As %Integer,Cancel As %Boolean)
{
//...
}
}
```

注釈: 開発者は、実際に BytesTransferred メソッドを実装するために、**Activate.SomeLibrary.FTPEvents** クラスを直接編集するか、あるいはクラスを分類して、サブクラスの実装を提供するか (推奨) を決定できます。

ダウンロードの後、これ以上イベントを処理する必要がないため、ハンドラの登録を解除します。

```
Do FTP.%UnRegisterHandler(FTPHandler)
```

その後、後処理をします。

```
Do FTPHandler.%Close()  
Do FTP.%Close()
```

