MIMEOCX

WILL

株式会社ウィル

白紙ページ

白紙ページ

目 次

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

MIMEOCX 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

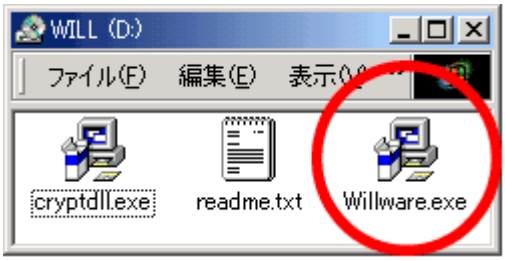
MIMEOCX をご使用いただくには、以下のいずれかのソフトウェアが必要です。

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

MIMEOCX は、Microsoft Visual C++ Ver 5.0 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。
※　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

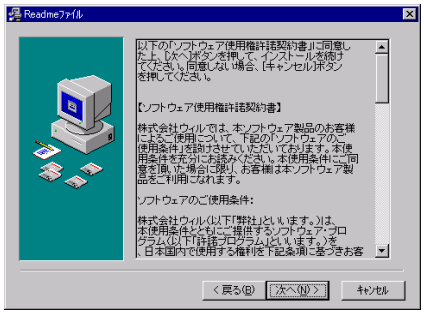
製品の CD-ROM に含まれているセットアップキット（Willware.exe）をダブルクリックします。



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。



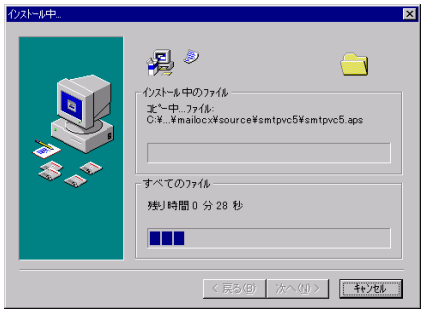
3. インストール先のフォルダを指定します。初期設定でよろしければ「次へ」をクリックして下さい。別のフォルダを指定したい場合は「参照」をクリックし、フォルダを指定して下さい。

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

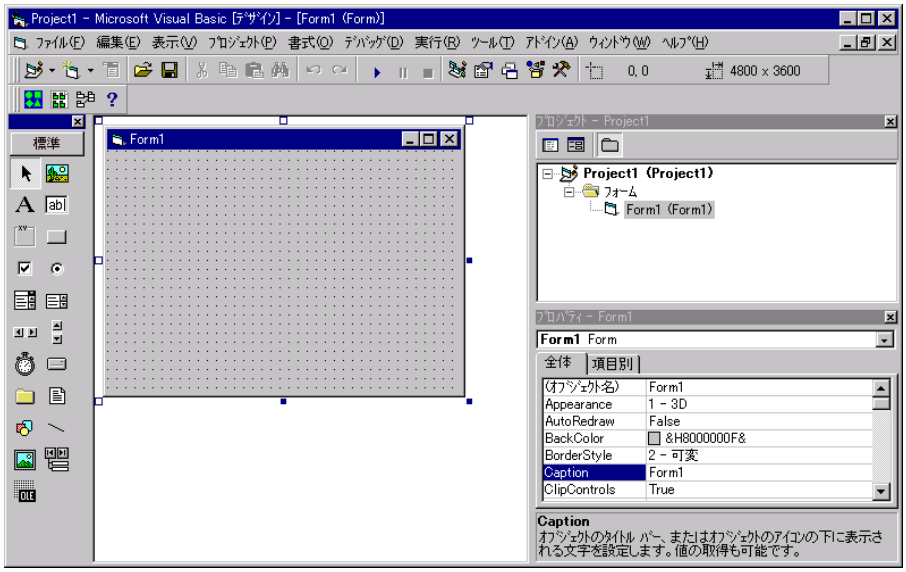
■レジストリファイルから登録する

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（EIXXXXXXXXX.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。）

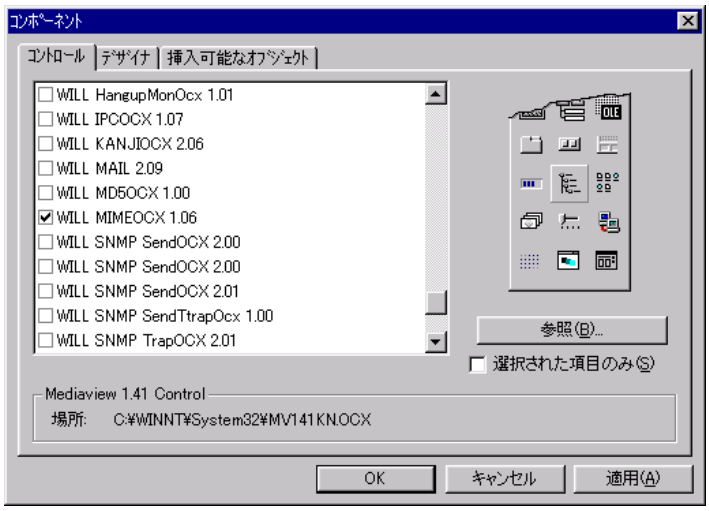
以下のメッセージボックスが表示され、ライセンスがレジストリに登録されます。

■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。



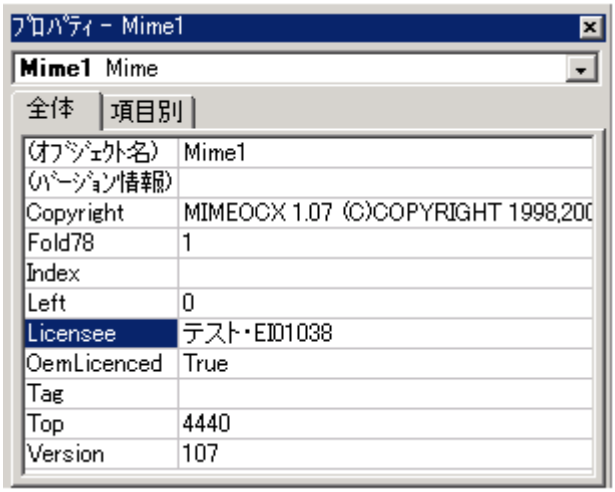
ツールバーの「プロジェクト」から、「コンポーネント」を選択し、「コンポーネント」画面を開きます。次にコントロールタブの一覧から MIMEOCX を選択して「OK」をクリックすると、MIMEOCX がツールボックスに追加され、アイコンが表示されます。



ツールボックスに追加された MIMEOCX 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 MIMEOCX 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、MIMEOCX 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。

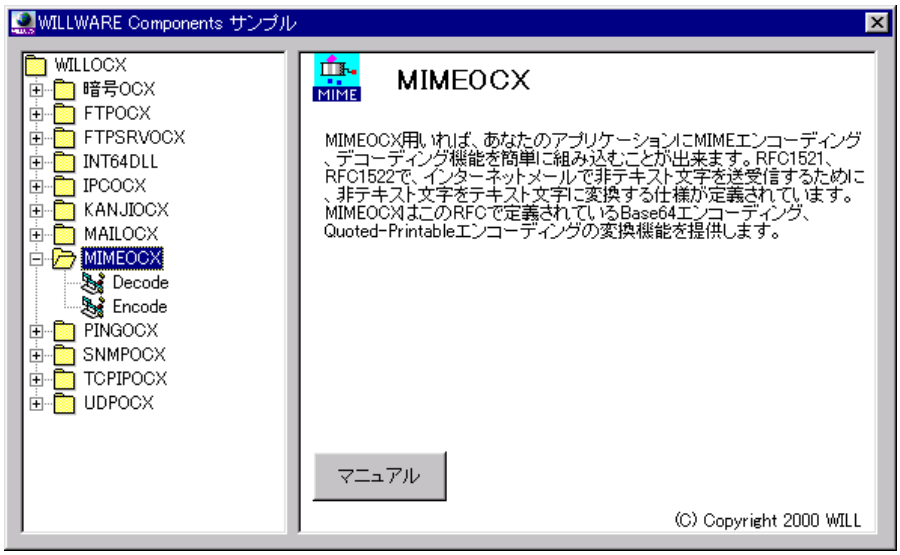
※トライアルライセンスで作成したアプリケーションはどうする！？

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

MIMEOCX 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

MIMEOCX 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。

- MIME.OCX
- 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

MIMEOCX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 (WindowsNT4.0, 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

- MIMEOCX 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。
- MIMEOCXを利用するアプリケーションはMIMEOCX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 MIMEOCX 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、MIMEOCX 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 MIMEOCX 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。

VB から VB 以外への文字列の受け渡し

VB の文字コードはすべて UNICODE を使用しますので、VB 以外のシステム(VCなど)と文字列を受け渡しする場合、文字コードを変換する必要があります。

UNICODE から ANSI へ変換

```
Text = StrConv(Text, vbFromUnicode)
```

ANSI から UNICODE へ変換

```
Text = StrConv(Text, vbUnicode)
```

Base64 でエンコードする場合

Base64Encode メソッドでエンコードした場合、エンコード後のデータは整形されておりません。
RFC2045 6.8.Base64 Concent-Transfer-Encoding の規定、「The output stream(encoded bytes)must be represented in lines of no more then 76 characters each.」に合わせて1行を 76 文字以下にする必要があります。エンコードされたデータに改行文字を埋め込む処理を追加します。

Base64 でエンコードする場合は、以下の 2 種類の方法があります。

1. 最初に全てのデータをメソッドに渡す方法
2. 最初に1行単位でデータをメソッドに渡す方法

状況に応じて、どちらかの方法でエンコードしてください。

1. 最初に全てのデータをメソッドに渡す方法

```
Dim src   'エンコードするデータの文字列
Dim code'エンコードされた Base64 の文字列

CRLF = ChrB$(13) & ChrB$(10)
src = Mime1.Base64Encode(src)
For i = 1 To LenB(src) Step 76
    tmp = MidB$(src, i, 76)
    If LenB(tmp) > 0 Then
        code = code & tmp & CRLF
    End If
Next
```

2. 最初に1行単位でデータをメソッドに渡す方法

ここでいう 1 行とは、エンコード後の 1 行をイメージします。エンコード後の文字数が 76 文字になるように、57 バイトごとに区切ってメソッドを呼び出します。エンコード後のデータには改行文字を付加します。

```
Dim src          'エンコードするデータの文字列
Dim code         'エンコードされた Base64 の文字列

CRLF = ChrB$(13) & ChrB$(10)
src = Mime1.Base64Encode(src)
length = LenB(src)
tmp = ''
```

```
For i = 1 To length Step 57
    tmp = MidB$(src, i, 57)
    If LenB(tmp) > 0 Then
        code = code & tmp & CRLF
    End If
Next
```

Quoted-Printable でエンコードする場合

Quoted-Printable は基本的にテキストを処理する方式です。処理する場合の Quoted-Printable の 1 行は、エンコード後もエンコード前も意味が変わりません。したがって、エンコード前のデータを 1 行ごとにエンコードします。(※ Quoted-Printable の場合、最初に全てのデータをメソッドに渡してエンコードすることはできません。)

また、エンコードする際、必ず文字列を ANSI に変換して渡してください。

```
Dim src       'エンコードするデータの文字列
Dim code      'エンコードされた Quoted-Printable の文字列

CRLF = ChrB$(13) & ChrB$(10)
div = LenB(CRLF)
src = StrConv(src, vbFromUnicode)        'UNICODE->ANSI へ変換
start = 1
Do While 1
    pos = InStrB(start, src, CRLF, vbBinaryCompare)
    If pos > 0 Then
        tmp = MidB(src, start, pos - start)
        code = code & Mime1.QuotedPrintableEncode(tmp) & CRLF
        start = pos + div
    Else
        If start < LenB(src) Then
            tmp = MidB(src, start, LenB(src) - start + 1)
            code = code & Mime1.QuotedPrintableEncode(tmp) & CRLF
        End If
        Exit Do
    End If
Loop
Text2 = StrConv(code, vbUnicode)        'ANSI -> UNICODE へ変換
```

uuencode でエンコードする場合

uuencode は、行頭にエンコードバイト数を示す文字が置かれます。この文字を生成するには、uuencodeByte メソッドにエンコードしたいデータを渡します。uuencode 本体は、uuencodeLine メソッドを呼び出して生成します。また、最初の行にはファイル名やパーミッションが記述され、最後は end のみの行で終わります。基本的には、45 バイト毎に区切ってエンコードすることをお勧めします。(uuencode の場合、最初に全てのデータをメソッドに渡してエンコードすることができません。)

```
Dim src         'エンコードするデータの文字列
Dim code        'エンコードされた uuencode の文字列

CRLF = ChrB$(13) & ChrB$(10)
fileName = StrConv("file1", vbFromUnicode)
permit = StrConv("600", vbFromUnicode)
code = Mime1.uuencodeBegin(fileName, permit) & CRLF
For i = 1 To LenB(src) Step 45
    tmp = MidB$(src, i, 45)
    uuHead = Mime1.uuencodeByte(tmp)
    uuBody = Mime1.uuencodeLine(tmp)
    code = code & uuHead & uuBody & CRLF
Next
code = code & Mime1.uuencodeEnd
```

Base64 でデコードする場合

Base64 にはエンコードデータの中に 1 行あたりのエンコード文字数が記述されておりません。必ず改行までの1行を渡してください。

Base64 の場合、行の途中でデータを渡してしまうと、そこで padding（穴埋め文字：「Base64 エンコード規則」参照）が現われた場合と同じ処理を行います。

※ デコードは、どのデコード方式でもエンコードデータを 1 行ごとにメソッドに渡す必要があります。

```
Dim src        'デコードする Base64 データの文字列
Dim org        'エンコードされた元の文字列

CRLF = ChrB$(13) & ChrB$(10)
div = LenB(CRLF)
start = 1
Do While 1
    pos = InStrB(start, src, CRLF, vbBinaryCompare)
    If pos > 0 Then
        tmp = MidB(src, start, pos - start)
        org = org & Mime1.Base64Decode(tmp)
        start = pos + div
    Else
        If start < LenB(src) Then
            tmp = MidB(src, start, LenB(src) - start + 1)
            org= org & Mime1.QuotedPrintableEncode(tmp) & CRLF
        End If
        Exit Do
    End If
Loop
```

Quoted-Printable でデコードする場合

※ デコードは、どのデコード方式でもエンコードデータを 1 行ごとにメソッドに渡す必要があります。

```
Dim src      'デコードする QuotedPrintable データの文字列
Dim org      'エンコードされた元の文字列

CRLF = ChrB$(13) & ChrB$(10)
div = LenB(CRLF)
VB からとった文字なら、ANSI に変換
'src = StrConv(src, vbFromUnicode)
start = 1
Do While 1
    pos = InStrB(start, src, CRLF, vbBinaryCompare)
    If pos > 0 Then
        tmp = MidB(src, start, pos - start)
        org = org & Mime1.QuotedPrintableDecode(tmp) & CRLF
        start = pos + div
    Else
        If LenB(src) > 0 Then
            tmp = MidB(src, start, LenB(src) - start + 1)
            org = org & Mime1.QuotedPrintableEncode(tmp) & CRLF
        End If
        Exit Do
    End If
Loop

'org = StrConv(org, vbUnicode)        'VB で表示する場合、unicode に変換
```

uuencode でデコードする場合

uuencode の場合、行頭にエンコード文字数が指定されているので、その文字数に満たない場合は、処理を行いません。

※ デコードは、どのデコード方式でもエンコードデータを 1 行ごとにメソッドに渡す必要があります。

```
Dim src       'デコードする uuencode データの文字列
Dim org       'エンコードされた元の文字列

CRLF = ChrB$(13) & ChrB$(10)
div = LenB(CRLF)
start = 1
pos = InStrB(start, src, CRLF, vbBinaryCompare)
tmp = MidB(src, start, pos - start)
fileName = StrConv(MidB(tmp, 11), vbUnicode)
permit = StrConv(MidB(tmp, 7, 3), vbUnicode)
lastBit = InStrB(1, src, CRLF & StrConv("end", vbFromUnicode))
start = pos + div
Do While 1
    pos = InStrB(start, src, CRLF, vbBinaryCompare)
    If pos > 0 And pos < lastBit Then
        tmp = MidB(src, start, pos - start)
        org = org & Mime1.UuDecodeLine(tmp)
        start = pos + div
    Else
        Exit Do
    End If
Loop
```

白紙ページ

プロパティ

白紙ページ

Copyright プロパティ

■機　能

　MIMEOCX 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 プロパティの構文の指定項目は次のとおりです。

(指定項目)	(内　容)
Object	MIMEOCX オブジェクトです。

■データ型

文字列(String)

Licensee プロパティ

■機　能

　MIMEOCX 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
Licensee 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	MIMEOCX オブジェクトです。

■データ型

文字列(String)

Fold78 プロパティ

■機　能

EncodeMimeHeader メソッドにおいて行の長さを 78 バイト(改行コードを含まず)に納める処理を行うかどうか示します。

■構　文

Object.Fold78[=Value]

Fold78 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	MIMEOCX オブジェクトです。
Value	改行を指示する長整数です。デフォルト値は 1 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	行が 78 バイトを超えるとき直前のカンマの位置で改行します。
2	行が 78 バイトを超えるとき直前の空白の位置で改行します。
4	行が 78 バイトを超えるときその位置で改行します。

■データ型

長整数(Long)

■解説

上記の設定値は組み合わせることができます。組み合わせの場合は数字の小さいものが優先されます。

白紙ページ

メソッド

白紙ページ

Base64Encode メソッド

■機　能

Source に与えられたバイト列を Base64 で符号化して返します。Source に指定されたバイト列を Base64 符号化規則で符号化します。エンコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します。

※「プログラミング概要」の「Base64 でエンコードする場合」も併せて参照してください。

■構　文

Code = Object.Base64Encode (Source As String)

Base64Encode 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	MIMEOCX オブジェクトです。
Source	バイト列を指定する文字列式です。
Code	エンコードされた文字列です。

■戻り値

String（文字列）

Base64Decode メソッド

■機　能

Source に与えられた Base64 符号化文字列をバイト列に変換して返します。Source に指定された Base64 符号化文字列を Base64 符号化規則でバイト列に変換します。デコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します

※「プログラミング概要」の「Base64 でデコードする場合」も併せて参照してください。

■構　文

Original = Object.Base64Decode(Source As String)

Base64Decode 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	MIMEOCX オブジェクトです。
Source	バイト列を指定する文字列式です。
Original	デコードされた文字列です。

■戻り値

String(文字列)

QuotedPrintableEncode メソッド

■機　能

Source に与えられたバイト列を Quoted-Printable で符号化して返します。Source に指定されたバイト列を Quoted-Printable 符号化規則で符号化します。エンコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します。

※「プログラミング概要」の「Quoted-Printable でエンコードする場合」も併せて参照してください。

■構　文

Code = Object.QuotedPrintableEncode(Source As String)

QuotedPrintableEncode 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	MIMEOCX オブジェクトです。
Source	Quoted-Printable 符号化文字列を指定する文字列式です。
Code	エンコードされた文字列です。

■戻り値

String（文字列）

QuotedPrintableDecode メソッド

■機　能

Source に与えられた Quoted-Printable 符号化文字列をバイト列に変換して返します。Source に指定された Quoted-Printable 符号化文字列を Quoted-Printable 符号化規則でバイト列に変換します。デコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します。

※「プログラミング概要」の「Quoted-Printable でデコードする場合」も併せて参照してください。

■構　文

Original = Object.QuotedPrintableDecode(Source As String)

QuotedPrintableDecode 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	MIMEOCX オブジェクトです。
Source	Quoted-Printable 符号化文字列を指定する文字列式です。
Original	デコードされた文字列です。

■戻り値

String（文字列）

UuEncodeLine メソッド

■機　能

Source に与えられたバイト列を uuencode で符号化して返します。Source に指定されたバイト列を uuencode 符号化規則で符号化します。戻り値に変換された文字列が設定されます。エンコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します。

※「プログラミング概要」の「uuencode でエンコードする場合」も併せて参照してください。

■構　文

Code = Object.UuEncodeLine(Source As String)

UuEncodeLine 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	MIMEOCX オブジェクトです。
Source	バイト列を指定する文字列式です。
Code	エンコードされた文字列です。

■戻り値

String(文字列)

UuEncodeBegin メソッド

■機　能

uuencode の begin 行を作成して返します。File で指定されたファイル名を begin 行に埋め込んで返します。このメソッドでは、begin 行を指定する場合パーミッションに”600”を指定してください。以下の文字列を作成します。

Begin 600 Filename

■構　文

Begin = Object.UuEncodeBegin(FileName As String, Permission As String)

UuEncodeBegin 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	MIMEOCX オブジェクトです。
Filename	begin 行に埋め込むファイル名を指定する文字列式です。
Permission	パーミッションを指定する文字列式です。
Begin	Uuencode の1行目の文字列です。

■戻り値

String(文字列)

UuEncodeEnd メソッド

■機　能

uuencode の end 行を作成し、返します。常に空行1行と end 行を作成します。
以下の文字列を作成します。

、
end

■構　文

End = Object.UuEncodeEnd()
UuEncodeEnd 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	MIMEOCX オブジェクトです。
End	Uuencode の末行の文字列です。

■戻り値

String(文字列)

uuencodeByte メソッド

■機　能

　1 行に含まれるバイト数を示す uuencode 文字を返します。uuencode では行頭に、その行に元のデータが何バイトのデータで含まれているかを示す文字が付加されます。UuEncodeByte はこの数値から文字への変換を行います。空文字を入れるとエラーが発生します。

■構　文

Num = Object.UuEncodeByte(Source As String)
UuEncodeByte 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	MIMEOCX オブジェクトです。
Source	バイト列を指定する文字列式です。
Num	バイト数を示す uuencode 文字列です。

■戻り値

String(文字列)

UuDecodeLine メソッド

■機　能

Source に与えられた uudecode 符号化文字列をバイト列に変換して返します。Source に指定された uudecode 符号化文字列を uudecode 符号化規則でバイト列に変換します。

デコードされた文字列を戻り値に返します。空文字を入れるとエラーが発生します。

■構　文

Code = Object.UuDecodeBegin(Source As String)

UuDecodeBegin 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	MIMEOCX オブジェクトです。
Source	uudecode 符号化文字列を指定する文字列式です。
Code	デコードされた文字列です。

■戻り値

String(文字列)

MimeHeaderEncode メソッド

■機　能

いわゆる MIME Header といわれる電子メールのヘッダに日本語を置くためのエンコードをサポートします。

MIME Header は次の書式で表されます。

=?charset?encoding?encoded_text?=

charset には各国の文字セットを指定できますが、現在の実装では ISO-2022-JP で固定です。encoding には Q エンコーディングと B エンコーディングが指定できますが、現在の実装では B エンコーディング固定です。Source をエンコードした結果 75 文字を超える場合は、RFC2046 の規定に合わせて、複数の encoded-word に分割する必要があります。(76 文字以上のエンコード文字は、動作を保証しません。)MIMEOCX は漢字コードの判定を行わないので、このメソッドの呼び出し側で文字数を限定して渡す必要があります。現在の実装では、元の文字コードが JIS の場合は少なくとも 24 文字ごとに、SJIS あるいは EUC の場合は、行を 28 文字ごとに切ってください。

■構　文

Header = Object.MimeHeaderEncode(Source As String)

MimeHeaderEncode 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	MIMEOCX オブジェクトです。
Source	エンコードする文字列を指定する文字列式です。
Header	MIME のヘッダー文字列です。

■戻り値

String(文字列)

MimeHeaderDecode メソッド

■機　能

Source にはフォーマットとして完全な encode-word を指定してください。通常 1 行分の文字列を指定すれば、完全な encoded-word になります。この時、行末の改行文字は Source に含めないでください。改行文字がある場合はエラーになってしまいます。また、1 行の中に encoded-word が2つ以上存在する場合1つ目のものをデコードし残りを Source に戻します。

■構　文

Original = Object.MimeHeaderDecode(Source As String)
MimeHeaderDecode 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	MIMEOCX オブジェクトです。
Source	デコードする文字列を指定する文字列式です。
Original	デコードされた文字列です。

■戻り値

String(文字列)

白紙ページ

付 録

白紙ページ

Base64 エンコード規則

Base64 符号化フォーマットは、3 バイトを単位として、それを 6 ビットの符号化単位 4 個に分割します。そして、これらの 6 ビット単位を表 1 に示した対応にしたがって符号化します。

表 1:Base64 の文字の割り当て

Val	Enc	Val	Enc	Val	Enc	Val	Enc	Val	Enc
0	A	13	N	26	a	39	n	52	0
1	B	14	O	27	b	40	o	53	1
2	C	15	P	28	c	41	p	54	2
3	D	16	Q	29	d	42	q	55	3
4	E	17	R	30	e	43	r	56	4
5	F	18	S	31	f	44	s	57	5
6	G	19	T	32	g	45	t	58	6
7	H	20	U	33	h	46	u	59	7
8	I	21	V	34	i	47	v	60	8
9	J	22	W	35	j	48	w	61	9
10	K	23	X	36	k	49	x	62	+
11	L	24	Y	37	l	50	y	63	/
12	M	25	Z	38	m	51	z	Pad	=

Base64 符号化データの終わりには、元のデータの長さによって、以下の 3 通りの状態が起り得ます。

1. 元のデータの長さが 3 倍である場合、符号化の結果、有効な文字 4 個が得られます。Padding 文字は必要なく、データの終わりは、メッセージ本体の終わりで判断します。

2. 最後の符号化グループに有効なデータが 1 バイト含まれる場合、この符号化グループには 0 か 2 バイト付加され、符号化の結果、2 個の文字に有効なデータが含まれることになります。残りの 2 文字には等号が使われ、デコーダに対してこれらの文字には有効なデータが含まれないことを知らせます。

3. 最後の符号化グループに有効なデータが2バイト含まれる場合、この符号化グループには 0 か 1 バイト付加され、符号化の結果、3 個の文字に有効なデータが含まれることになります。最後の文字には等号が使われ、デコーダに対してその文字には有効なデータが含まれていないことを知らせます。

■ メッセージのエンコード

元の文字列＝「Hello」

テキスト	H	e	l	l	o
16 進	48	65	6C	6C	6F
2 進	01001000	01100101	01101100	01101100	01101111

3 バイト(24 ビット)ごとに取り出し、6 ビットごとの4つのグループに分けます。

H	e	l
010010　00	0110　0101	01　101100

010010	000110	010101	101100
18	6	21	44
S	G	V	s

残りの文字を処理します。
残りの文字は 3 バイトに満たないので、穴埋め用の文字を使用します。
穴埋め用のデータは「00000000」を使用するが、有効な「00000000」と区別するため、ここでは「?」表現しています。

l	o	
011011　00	0110　1111	??　??????

011011	000110	1111??	??????
27	6	60	
b	G	8	=

したがって、「Hello」という文字列は Base64 符号化により、「SGVsbG8=」となります。

■ メッセージのデコード

元の文字列「SGVsbG8=」は、4 バイトごとに取り出し、変換テーブルから数字に変換し、6 ビット表現にします。その後 8 ビットごとにまとめます。

S	G	V	s
18	6	21	44
00010010	00000110	00010101	00101100
010010	000110	010101	101100
010010 00	0110 0101	01 101100	
H	**e**	**l**	

残りの文字を処理します。

b	G	8	=
27	6	60	?
00011011	00000110	00111100	????????
0011011	000110	111100	??????
01101100	01101111	00??????	
l	**o**		

最後の 1 文字は穴埋め文字なので捨てます。

Quoted-Printable 規則

Quoted-Printable の符号化法は、データの各バイトを、等号(=)に続けてバイト値を 16 進2桁で表記することで表現します。印刷可能な ASCII 文字(33 から 60 までと 62 から 126 まで)の範囲の値のバイトは ASCII 文字もそのもので表すことも出来ます。また、Quoted-Printable は、行末(the end of line)に「空白」か「タブ」を置くことを禁止しています。その場合、それぞれ「=20」「=09」に符号化しなければなりません。Non-whitespace 文字が続いている場合は、符号化しません。

元の文字列＝「Hello」

H		e		l		l		o	
01001000		01100101		01101100		01101100		01101111	
0100	1000	0110	0101	0110	1100	0110	1100	0110	1111
4	8	6	5	6	12	6	12	6	15
3	7	5	4	5	B	5	B	5	F

変換後の文字列
=37=54=5B=5B=5F
但し、通常 ASCII 文字列を Quoted-Printable 符号化することはありません。

uuencode 規則

uuencode 符号化ファイルは次のようになります。

begin <permissions><file name>
<lines of encoded data>
<zero line>
end

最初の行は、begin で始まり、その後にファイルのセキュリティ許可情報を符号化した番号、最後にファイル名が続きます。

begin 行のあとには、符号化されたデータ行が続きます。符号化データの終わりには、長さが 0 の行が置かれます。uuencode の終わりには、end 行が置かれます。

begin 行のセキュリティの指定は、UNIX のファイルシステムの許可ビットと同じです。8 進数 3 桁で表示されます。たとえば、640 という指定はファイルの所有者に読み書き許可を与え、同じグループに属するユーザに読み込み許可だけを与え、その他のユーザにはすべてのアクセスを許可しないという指定になります。ファイル名には DOS の制限（大文字小文字を区別しません。長さが 8+3 文字まで。）は適用されません。そのため、DOS を相手にする可能性がある場合は、DOS の適当なファイル名を作り出すか、ユーザーがファイル名を指定できるようにしなければなりません。

begin 行のあとには、符号化データ行が続きます。符号化データ行は、符号化されたデータのバイト数を示す文字で始まり、そのあとに符号化バイトが続きます。データの符号化方法は、3 バイトを単位として、それを 6 ビットの符号化単位 4 個に分割します。ここで得られた値にスペース文字の ASCII 値(10 進数で 32)を加え、印刷可能な ASCII 文字を得ます。

元の文字列＝「Hello」

テキスト	H	e	l	l	o
16 進	48	65	6C	6C	6F
2 進	01001000	01100101	01101100	01101100	01101111

3 バイト(24 ビット)ごとに取り出し、6 ビットごとの4つのグループに分けます。

H	E	l
010010 00	0110 0101	01 101100

010010	000110	010101	101100
18	6	21	44
S	G	V	s

残りの文字を処理します。

l	O	
011011 00	0110 1111	?? ??????

011011	000110	1111??	??????
27	6	60	
59	38	92	
;	&	¥	

符号化文字列　　2&5L;&¥

符号化データ行は、最後の行を除き、符号化文字 4 個を単位とし、そのグループを何個か含んでいなければなりません。

元のデータの長さが 3 で割り切れない場合は、最後の符号化グループには 3 バイト分のデータが含まれなくなります。この場合、必要なだけ 0 を埋めて、完全な 3 バイトグループにして、通常どおりの符号化を行い、すべての符号化文字を最後の行に含めます。行頭の長さ文字には、元のデータバイト数を示しておき、デコーダは必要な情報を得られるようにします。

データの最後の行は、0 バイトを示す長さ文字(スペース文字)で構成されます。このあとに、空行(CRLF のみの行)が来なければなりません。

最後に、uuencode フォーマットの終わりの行として、end だけの行を示します。

なお、6 ビットの符号化用データ値が 0 のとき、符号化文字はスペース文字(' ')が割り当てられるはずですが、一部の MTA (Mail Transfer Agent)は行末の一連のスペース文字を削除してしまいます。この問題を回避するために、多くの uuencode プログラムはスペース文字をバックティック文字('`'、ASCII コードで 10 進の 96)を割り当てるようにしています。

サンプル

白紙ページ

Encode

(Ver 1.00)

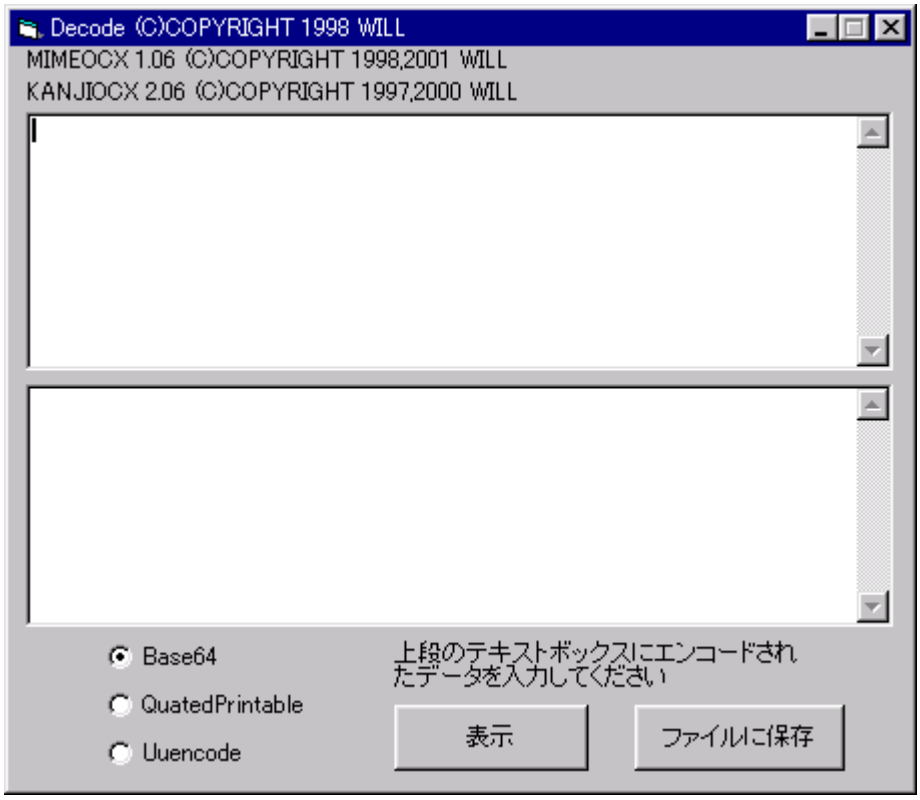
MIME エンコードサンプルです。

　左のフレームにエンコードするファイルをドロップすると、右のフレームに Base64 方式でエンコードされたデータが表示されます。

Decode

(Ver 1.00)

MIME デコードサンプルです。
　エンコードされたデータを上段のテキストボックスに入力し、エンコード方式を選択して「表示」ボタンをクリックしてください。下段にデコードされたデータが表示されます。
　「ファイルに保存」ボタンで、デコードしたデータを指定したディレクトリに保存することができます。



索 引

白紙ページ

MIMEOCX マニュアル

1998 年　4 月 21 日　　初　版
1998 年 10 月　1 日　　第 2 版
1998 年 12 月　3 日　　第 3 版
1999 年　4 月　1 日　　第 4 版
1999 年　7 月　1 日　　第 5 版
2002 年　5 月　8 日　　第 6 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町 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