Présentation de Microsoft Visual FoxPro juin 2012 Paris.

par Mike Gagnon

1. Slide #3 – question #1

Windows Management Instrumentation est une technologie de gestion de base de Windows, vous pouvez utiliser WMI pour gérer des ordinateurs locaux et distants. Le mot «Instrumentation» dans WMI se réfère au fait que WMI peut obtenir des informations sur l'état interne des systèmes informatiques, tout comme les instruments de bord des voitures peut récupérer et afficher des informations sur l'état du moteur. WMI fournit une approche cohérente à la réalisation de tâches de gestion au jour le jour avec les langages de programmation ou de scripting. On peut accéder aux données de WMI, la plupart du temps par requête SQL et le résultat de cette requête nous donnes une collection de propriétés.

Qu'est-ce que une ressource gérée? Pour les fins du présent chapitre, une ressource gérée est tout autre matériel informatique (hardware), logiciels, un service, un compte d'utilisateur, et ainsi de suite qui peut être gérée à l'aide de WMI

Les requêtes SQL sont exécutées sur différentes classes. Il existe 10 groupes de classes dans Windows.

i) Les classes Win32
ii) Les classes WMI de base de registre.
iii) Les classes WMI de système.
iv) Les classes WMI moniteur d'affichage.
v) Les classes IPMI (Intelligent Platform
vi) Management Interface)
vii) Les classes MSFT.
viii) Les classes CMI (Intelligent Platform Management Interface)
ix) Les classes de consommation courante.
x) Les classes MSMCA.
xi) Les classes WMI C++.

Nous allons aujourd'hui nous concentrer sur une ou deux de ces classes. Mais les requêtes à faire sur les autres classes sont habituellement faites dans le même style.

Les classes Win 32.

On retrouve 6 sous-catégories de classe Win32. Les classes Win32, comme Win32_NetworkAdapter ou Win32_Process, surveiller et gérer le matériel du système et des fonctionnalités. Généralement, ces classes sont situées dans le root \ cimv2 .

i) Classe de matériel informatique du système
ii) Classe d'applications installées.
iii) Classes de systèmes d'exploitation.
iv) Classes de compteur de performance.

v) Aide descripteur de sécurité de classe
vi) wmi service de gestion de classe

Classe de matériel informatique du système.

Cette classe est subdivisée en 10 sections et chaque subdivision a aussi ses sous-divisions. Alors on peut voir l'ampleur et l'utilité de WMI lorsque l'on veut développer une application pour gérer un parc informatique sous tous ses aspects. Microsoft nous fournit aussi un outil pour écrire nos requêtes (Code Creator – démontrer C:\June2012\WMI\code creator). Le code qui en ressort est en Visual Basic, mais la traduction à FoxPro est très facile.

On peut aussi profiter d'IntelliSense de Foxpro pour obtenir les propriétés que nous avons de besoin.

```
LOCAL lcComputerName, loWMIService, loItems, loItem, lcMACAddress
lcComputerName = "."
loWMIService = GETOBJECT("winmgmts:\\" + lcComputerName + "\root\cimv2")
loItems = loWMIService.ExecQuery("Select * from Win32_NetworkAdapter",,48)
FOR EACH loItem IN loItems
 ?loItem.GetObjectText_
ENDFOR
```

Exemples.

1. NetworkAdapterConfiguration.prg
2. ProcessorID.prg
3. Numerodeserie.prg
4. Adressemac.prg
5. DisplayControler1.prg
6. Operating system (op.exe)

Classe d'applications installées.

Vous pouvez trouver plusieurs exemples de scripts que Microsoft met à votre disposition site le site de Microsoft http://technet.microsoft.com/en-ca/scriptcenter/dd793613.

1. Plusieurs programmeurs FoxPro cherchent à découvrir ou utiliser l'information interne d'un ordinateur pour protéger par exemple contre l'utilisation illégale d'une application. Cette méthode n'est pas une bonne pratique puisque chaque manufacturier peut décider de stocker le numéro de série d'un disque dur (par exemple) dans un autre endroit, et votre code ne retournera pas le numéro de série et aussi les composantes d'un ordinateur peuvent changer plusieurs fois dans la vie de l'ordinateur donc empêcher une application de démarrer parce qu'elle ne trouve pas le bon numéro de série ne fait que frustrer le client. Surtout si ce client est éloigné. Une meilleur moyen de protéger une application est une clé électronique mais encore là il y a toujours moyen de contourner cette pratique. Une pratique moderne et courante mais pas parfaite est au lieu de juste chercher la présence d'une clé électronique, mais plutôt de mettre une fonction principale (comme la vérification du mot de passe) sur la clé électronique dans un DLL .Net, qui n'a pas besoin d'être enregistré, et si la clé électronique n'est pas présente, le logiciel ne part pas.

Windows API (Application programming interfaces)

Communément appelé Win32 API, qui reflète le support de 32 bit. Ce sont des DLL (en 16 bits ces DLL étaient des exécutables) que presque tous les programmes sous la plateforme Windows interagissent avec ces DLL d'une façon ou une autre. Certains de ces DLL qui exposent des interfaces (comctl32.dll) ont été porté aussi en OCX pour leur son utilisation dans les applications comme Foxpro et autres.

Nous retrouvons 8 catégories de base pour ces DLL.

Les services de base.

Les services avancés.

GDI

User Interface

Common dialog library.

Common control library

Windows shell

Windows services.

1. **Les services de base**. Fournir un accès aux ressources fondamentales disponibles à un système Windows. Inclus sont des choses comme les systèmes de fichiers, dispositifs, procédés et fils, et la gestion des erreurs. Sous la version 32 bits de Windows (et 64 bits), ces fonctions sont contenues dans le kernel32.dll. Voici un exemple d'utilisation (obtenir le nom du répertoire système) :

```
clear

DECLARE INTEGER GetSystemDirectory IN kernel32;
    STRING @ lpBuffer, INTEGER nSize

LOCAL lpBuffer, nSizeRet

lpBuffer = SPACE (250)
nSizeRet = GetSystemDirectory(@lpBuffer, Len(lpBuffer))
lpBuffer = SUBSTR (lpBuffer, 1, nSizeRet)
? lpBuffer
```

2. **Les services avancés**. Donner accès à des fonctionnalités supplémentaires au noyau. Inclus sont des choses comme le registre de Windows, arrêt / redémarrage du système (ou annuler), démarrer / arrêter / créer un service Windows, gérer les comptes utilisateurs. Ces fonctions sont contenues dans le DLL advapi32.dll. Voici un exemple de son utilisation (éteindre un ordinateur) (remarquez que ce code nous donne une erreur et nous allons voir plutard pourquoi)

```
Declare Integer GetLastError In kernel32

Declare Integer InitiateSystemShutdownA In advapi32;
AS InitiateSystemShutdown ;
STRING lpMachineName, String lpMessage,;
INTEGER dwTimeout, SHORT bForceAppsClosed,;
SHORT bRebootAfterShutdown


Local cMachineName, nResult
cMachineName=""

nResult=InitiateSystemShutdown(cMachineName,;
"System Shutdown initiated...", 10, 0, 1)
```

```
If nResult = 0
* Common reasons for failure include an invalid
* or inaccessible computer name or insufficient
privilege.
*   5 = ERROR_ACCESS_DENIED
*  53 = ERROR_BAD_NETPATH
* 120 = ERROR_CALL_NOT_IMPLEMENTED -- not supported in
Win9*
   ? "Error code:", GetLastError()
Endif
```

3. **Graphics Device Interface**. Fournit une fonctionnalité de sortie du contenu graphique pour les moniteurs, les imprimantes et autres périphériques de sortie. Il réside dans GDI.EXE sur Windows 16 bits, et gdi32.dll sur Windows 32 bits en mode utilisateur. Celui est le DLL que le nouvel engin que les reports de Visual Foxpro version 9 utilisent.
   Il n'y a pas d'exemple simple pour l'utilisation du GDI32.dll puisque pour l'utiliser on doit faire appel à plusieurs autres fonctions pour les combiner pour pouvoir obtenir le résultat désiré. Mais vous pouvez consulter le site Atoutfox ou vous trouverez quelques exemples que moi et d'autres on poste.
   http://www.atoutfox.org/articles.asp?ACTION=FCONSULTER&ID=0000000124
4. **User Interface.** Fournit les fonctionnalités pour créer et gérer des fenêtres d'écran et les contrôles les plus élémentaires, tels que des boutons et des barres de défilement, de recevoir la souris et du clavier, et d'autres fonctionnalités associées à la partie GUI de Windows. Cette unité fonctionnelle réside dans user.exe sur Windows 16 bits, et user32.dll sur Windows 32 bits. Voici un exemple de son utilisation (comment obtenir le hWnd de la fenêtre active, même si les formes Foxpro nous donnent cette information, parfois on veut savoir le hWnd d'une fenêtre autre qu'une forme Foxpro, comme la calculatrice de Windows)

```
Declare Integer GetActiveWindow In user32

HWnd = GetActiveWindow()
? HWnd
```

5. **Common Dialog Box Library (**Bibliothèque boîte de dialogue commune) Fournit des applications les boîtes de dialogue standard d'ouverture et de sauvegarde de fichiers, choisir la couleur et la police, etc. La bibliothèque se trouve dans un fichier appelé COMMDLG.DLL sur Windows 16 bits, et comdlg32.dll sur Windows 32 bits. Voici un exemple de son

utilisation (comment utiliser le sélecteur de couleur de Windows, qui est plus complet que celui de Foxpro)

```
DO declare

#DEFINE CC_RGBINIT 1
#DEFINE CC_FULLOPEN 2
#DEFINE CC_PREVENTFULLOPEN 4
#DEFINE CC_SHOWHELP 8
#DEFINE CC_SOLIDCOLOR 128
#DEFINE CC_ANYCOLOR 256
#DEFINE CC_WIDE 32

*| typedef struct {
*|   DWORD        lStructSize;      0:4
*|   HWND         hwndOwner;        4:4
*|   HWND         hInstance;        8:4
*|   COLORREF     rgbResult;       12:4
*|   COLORREF   * lpCustColors;    16:4
*|   DWORD        Flags;           20:4
*|   LPARAM       lCustData;       24:4
*|   LPCCHOOKPROC lpfnHook;        28:4
*|   LPCTSTR      lpTemplateName;  32:4
*| } CHOOSECOLOR, *LPCHOOSECOLOR; total=36 bytes

#DEFINE CHOOSECOLOR_SIZE     36

*| typedef DWORD COLORREF;
*| typedef DWORD *LPCOLORREF;
*| 0x00bbggrr
#DEFINE COLORREF_ARRAY_SIZE  64

LOCAL hWindow, lcBuffer, lnInitColor, lnFlags,;
    lnCustColors, lcCustColors, ii

hWindow = GetActiveWindow()

* the color initially selected when the dialog box is
created
lnInitColor = Rgb(128,0,0)

* allocating memory block for 16 COLORREF values
(DWORD)
* for the custom color boxes in the dialog box
* and filling this memory with zeroes

#DEFINE GMEM_FIXED 0
lnCustColors = GlobalAlloc(GMEM_FIXED,
COLORREF_ARRAY_SIZE)
= ZeroMemory(lnCustColors, COLORREF_ARRAY_SIZE)

* initialization flags
lnFlags = CC_FULLOPEN + CC_RGBINIT

* compiling the CHOOSECOLOR structure
lcBuffer = num2dword(CHOOSECOLOR_SIZE) +;
```

```
    num2dword(hWindow) +;
    num2dword(0) +;
    num2dword(lnInitColor) +;
    num2dword(lnCustColors) +;
    num2dword(lnFlags) +;
    num2dword(0) +;
    num2dword(0) +;
    num2dword(0)

IF ChooseColor(@lcBuffer) <> 0
* the OK button of the dialog box has been selected
    ? "Color selected:", buf2dword(SUBSTR(lcBuffer,
13,4))

    ? "Custom colors stored:"

    * copying the memory block content to a VFP string
    * just to have an opportunity to work with its
substring;
    * quite weird way, though not much choice
available
    * considering the VFP specifics

    lcCustColors = Repli(Chr(0), COLORREF_ARRAY_SIZE)
    = Heap2Str(@lcCustColors, lnCustColors,
COLORREF_ARRAY_SIZE)

    FOR ii=1 TO 16
        ? ii, buf2dword(SUBSTR(lcCustColors, (ii-
1)*4+1, 4))
    ENDFOR
ENDIF

* free the memory block if you do not use it for the
* following calls to this function
= GlobalFree(lnCustColors)

* end of main

FUNCTION num2dword (lnValue)
#DEFINE m0 0x100
#DEFINE m1 0x10000
#DEFINE m2 0x1000000
    LOCAL b0, b1, b2, b3
    b3 = Int(lnValue/m2)
    b2 = Int((lnValue - b3*m2)/m1)
    b1 = Int((lnValue - b3*m2 - b2*m1)/m0)
    b0 = Mod(lnValue, m0)
RETURN Chr(b0)+Chr(b1)+Chr(b2)+Chr(b3)

FUNCTION buf2dword (lcBuffer)
#DEFINE MAX_DWORD 0xffffffff
#DEFINE MAX_LONG 0x7FFFFFFF
    LOCAL lnResult
    lnResult = Asc(SUBSTR(lcBuffer, 1,1)) + ;
        Asc(SUBSTR(lcBuffer, 2,1)) * 256 +;
        Asc(SUBSTR(lcBuffer, 3,1)) * 65536 +;
```

```
        Asc(SUBSTR(lcBuffer, 4,1)) * 16777216
RETURN IIF(lnResult>MAX_LONG, lnResult-MAX_DWORD,
lnResult)

PROCEDURE declare
    DECLARE INTEGER ChooseColor IN comdlg32 STRING
@lpcc
    DECLARE INTEGER GetActiveWindow IN user32
    DECLARE INTEGER GlobalFree IN kernel32 INTEGER
hMem
    DECLARE RtlZeroMemory IN kernel32 As ZeroMemory;
        INTEGER dest, INTEGER numBytes

    DECLARE INTEGER GlobalAlloc IN kernel32;
        INTEGER wFlags, INTEGER dwBytes

    DECLARE RtlMoveMemory IN kernel32 As Heap2Str;
            STRING @, INTEGER, INTEGER
```

6. **Common Control Library** (Bibliothèque de contrôle commun) Donne des applications l'accès à certains contrôles avancés fournis par le système d'exploitation. Il s'agit notamment des choses comme les barres d'état, barres de progression, barres d'outils et les onglets. La bibliothèque se trouve dans un fichier DLL appelé commctrl.dll sur Windows 16 bits, et comctl32.dll sur Windows 32 bits. Voici un exemple de son utilisation (un peu compliquer de news2news :

```
LOCAL oForm As Tform
oForm=CREATEOBJECT("Tform")
oForm.Show
READ EVENTS
* end of main

DEFINE CLASS Tform As Form
    Caption = "SysLink Control"
    Height=160
    Width=360
    Autocenter=.T.
    ShowWindow=2

    * add before adding first comctl32 control
    ADD OBJECT Comctl32Manager1 As Comctl32Manager

    * the angled brackets are replaced with Chrs
    * for presentation purposes only, since they are
    * HTML special characters

    ADD OBJECT SysLink1 As SysLink WITH;
    Left=15, Top=10, Width=320,;
```

```
    LinkCaption=[Visit our ]+CHR(60)+;
        [a href="www.downloadpage.com"]+CHR(62)+;
        [download page]+CHR(60)+[/a]+CHR(62)+[ ] +;
        [or ]+CHR(60)+[a
href="mailto:support@domain.com"]+;
        CHR(62)+[contact us]+CHR(60)+[/a]+CHR(62)+[ by
email.]

    ADD OBJECT SysLink2 As SysLink WITH;
    Left=40, Top=40, Width=200,;
    LinkCaption=[Choose feedback:] + CHR(13) +;
        [   - ]+CHR(60)+[a ID="idComment"]+CHR(62)+;
            [Comment]+CHR(60)+[/a]+CHR(62)+[] + CHR(13)
+;
        [   - ]+CHR(60)+[a ID="idQuestion"]+CHR(62)+;
            [Question]+CHR(60)+[/a]+CHR(62)+[] + CHR(13)
+;
        [   - ]+CHR(60)+[a ID="idComplaint"]+CHR(62)+;
            [Complaint]+CHR(60)+[/a]+CHR(62)+[]

    ADD OBJECT SysLink3 As SysLink WITH;
    Left=25, Top=127, Width=100,;
    LinkCaption=CHR(60)+[a ID="idHelp"]+CHR(62)+;
        [Help]+CHR(60)+[/a]+CHR(62)

    ADD OBJECT shp As Shape WITH Left=10,;
        Top=116, Width=340, Height=1

    ADD OBJECT cmdClose As CommandButton WITH;
        Left=260, Top=124, Width=80, Height=27,;
        Caption="Close", Cancel=.T.

PROCEDURE Destroy
    CLEAR EVENTS

PROCEDURE cmdClose.Click
    ThisForm.Release

PROCEDURE SysLink1.OnClick
    ACTIVATE SCREEN
    WITH THIS.oNMLink
        ? THIS.CtrlId, .litem_iLink,;
            .litem_szID, .litem_szUrl
    ENDWITH

PROCEDURE SysLink2.OnClick
    ACTIVATE SCREEN
    WITH THIS.oNMLink
        ? THIS.CtrlId, .litem_iLink,;
            .litem_szID, .litem_szUrl
    ENDWITH

PROCEDURE SysLink3.OnClick
    ACTIVATE SCREEN
    WITH THIS.oNMLink
        ? THIS.CtrlId, .litem_iLink,;
            .litem_szID, .litem_szUrl
```

```
    ENDWITH

ENDDEFINE

DEFINE CLASS Comctl32Manager As Custom
#DEFINE GWL_WNDPROC -4
#DEFINE GWL_HINSTANCE -6
#DEFINE WS_VISIBLE 0x10000000
#DEFINE WS_CHILD 0x40000000
#DEFINE HWND_TOP 0
#DEFINE SWP_NOSIZE 0x0001
#DEFINE SWP_NOMOVE 0x0002
#DEFINE SWP_SHOWWINDOW 0x0040
#DEFINE WM_NOTIFY 0x004e
#DEFINE WM_USER 0x0400
#DEFINE LM_GETIDEALSIZE WM_USER+0x0301
#DEFINE LWS_NOPREFIX 0x0004
#DEFINE LWS_TRANSPARENT 0x0001
#DEFINE LWS_USEVISUALSTYLE 0x0008
#DEFINE NM_FIRST 0
#DEFINE NM_CLICK (NM_FIRST-2)
#DEFINE NM_RETURN (NM_FIRST-4)
#DEFINE NM_SETFOCUS (NM_FIRST-7)
#DEFINE NM_CUSTOMDRAW (NM_FIRST-12)
#DEFINE MAX_LINKID_TEXT 48
#DEFINE L_MAX_URL_LENGTH 2084
#DEFINE NMHDR_SIZE 12
#DEFINE LITEM_SIZE 4280
#DEFINE NMLINK_SIZE NMHDR_SIZE+LITEM_SIZE

    Visible=.F.
    hParentHwnd=0
    hOrigProc=0
    Comctl32Controls=NULL

PROCEDURE Init
    WITH THIS
        .declare
        .hParentHwnd=ThisForm.hWnd
        .hOrigProc=GetWindowLong(.hParentHwnd,
GWL_WNDPROC)
        .Comctl32Controls=CREATEOBJECT("Collection")

        * comctl32 messaging is conducted
        * via WM_NOTIFY messages
        BINDEVENT(THIS.hParentHwnd, WM_NOTIFY,;
            THIS, "WindowProc", 1)

        * release first
        BINDEVENT(ThisForm, "Destroy", THIS,
"ReleaseControls", 1)
    ENDWITH

PROCEDURE Destroy
    IF THIS.hParentHwnd <> 0
        UNBINDEVENTS(THIS.hParentHwnd, WM_NOTIFY)
        THIS.hParentHwnd=0
```

```
        THIS.hOrigProc=0
    ENDIF

PROCEDURE ReleaseControls
    DO WHILE THIS.Comctl32Controls.Count > 0
        THIS.Comctl32Controls.Remove(1)
    ENDDO

PROTECTED PROCEDURE KeyFromCtrlId(nCtrlId As Number) As
String
RETURN "c_" + PADL(TRANSFORM(m.nCtrlId),5,"0")

PROTECTED PROCEDURE ControlIsRegistered(nCtrlId As
Number) As Boolean
    LOCAL oControl As Comctl32Control, lResult
    oControl=THIS.ControlFromCtrlId(nCtrlId)
    lResult=NOT ISNULL(m.oControl)
    oControl=NULL
RETURN m.lResult

PROTECTED PROCEDURE ControlFromCtrlId(nCtrlId As Number)
As Comctl32Control
    LOCAL cCtrlKey, oControl As Comctl32Control,;
        ex As Exception
    cCtrlKey=THIS.KeyFromCtrlId(m.nCtrlId)
    TRY
        oControl=THIS.Comctl32Controls.Item(m.cCtrlKey)
    CATCH TO ex
        oControl=NULL
    ENDTRY
RETURN m.oControl

PROCEDURE RegisterControl(oControl As Comctl32Control)
    LOCAL cCtrlKey, ex As Exception
    IF EMPTY(oControl.CtrlId)
        oControl.CtrlId=0x1000 +;
            THIS.Comctl32Controls.Count
    ENDIF
    cCtrlKey=THIS.KeyFromCtrlId(oControl.CtrlId)
    TRY
        THIS.Comctl32Controls.Add(oControl, m.cCtrlKey)
    CATCH TO ex
    ENDTRY

PROTECTED PROCEDURE WindowProc
PARAMETERS hWindow as Integer, nMsgID as Integer,;
    wParam as Integer, lParam as Integer

    LOCAL nReturn, oNmhdr As NMHdrStruct

    oNmhdr=CREATEOBJECT("NMHdrStruct")
    oNmhdr.FromPtr(m.lParam)

    nReturn = CallWindowProc(THIS.hOrigProc, m.hWindow,;
        m.nMsgID, m.wParam, m.lParam)

    IF THIS.ControlIsRegistered( oNmhdr.idFrom )
```

```
        THIS.OnCtrlEvent(m.oNmhdr, m.hWindow,;
            m.nMsgID, m.wParam, m.lParam)
    ENDIF

RETURN m.nReturn

PROTECTED PROCEDURE OnCtrlEvent
PARAMETERS oNmhdr As NMHdrStruct, hWindow as Integer,
nMsgID as Integer,;
    wParam as Integer, lParam as Integer

    LOCAL oControl As Comctl32Control
    oControl=THIS.ControlFromCtrlId( oNmhdr.idFrom )
    WITH oControl
        .EventId=oNmhdr.EventId
        .OnCtrlEvent(m.hWindow, m.nMsgID, m.wParam,
m.lParam)
    ENDWITH

PROTECTED PROCEDURE declare
    DECLARE RtlMoveMemory IN kernel32 As MemToStr;
        STRING @, INTEGER, INTEGER

    DECLARE INTEGER DestroyWindow IN user32;
        INTEGER hWindow

    DECLARE INTEGER GetWindowLong IN user32;
        INTEGER hWindow, INTEGER nIndex

    DECLARE INTEGER CreateWindowEx IN user32 AS
CreateWindow;
        INTEGER dwExStyle, STRING lpClassName,;
        STRING lpWindowName, INTEGER dwStyle,;
        INTEGER x, INTEGER y, INTEGER nWidth, INTEGER
nHeight,;
        INTEGER hWndParent, INTEGER hMenu, INTEGER
hInstance,;
        INTEGER lpParam

    DECLARE INTEGER SendMessage IN user32;
    AS SendMessageStr;
        INTEGER hWindow, INTEGER Msg,;
        INTEGER wParam, STRING @lParam

    DECLARE INTEGER SetWindowPos IN user32;
        INTEGER hWindow, INTEGER hWndInsertAfter,;
        INTEGER x, INTEGER y, INTEGER cx, INTEGER cy,;
        INTEGER wFlags

    DECLARE INTEGER CallWindowProc IN user32;
        INTEGER lpPrevWndFunc, INTEGER hWindow, LONG
Msg,;
        INTEGER wParam, INTEGER lParam

ENDDEFINE
```

```
DEFINE CLASS Comctl32Control As Container && Container,
TextBox
    hParentWindow=0
    hWindow=0
    EventId=0
    CtrlId=0

PROCEDURE Init
    WITH THIS
        .hParentWindow=ThisForm.Hwnd
        .RegisterControl
        .DisplayObject
    ENDWITH

PROCEDURE Destroy
    THIS.DestroyObject

PROTECTED PROCEDURE DestroyObject
    IF THIS.hWindow <> 0
        = DestroyWindow(THIS.hWindow)
        THIS.hWindow=0
    ENDIF

PROTECTED PROCEDURE RegisterControl
    ThisForm.Comctl32Manager1.RegisterControl(THIS)

PROCEDURE DisplayObject  && abstract

PROCEDURE OnCtrlEvent  && abstract
PARAMETERS hWindow as Integer, nMsgID as Integer,;
    wParam as Integer, lParam as Integer

ENDDEFINE

DEFINE CLASS SysLink As Comctl32Control
    oNMLink=NULL
    LinkCaption=""

PROCEDURE Init
    Comctl32Control::Init()
    THIS.oNMLink=CREATEOBJECT("oNMLinkStruct")

PROCEDURE OnCtrlEvent
PARAMETERS hWindow as Integer, nMsgID as Integer,;
    wParam as Integer, lParam as Integer
    Comctl32Control::OnCtrlEvent(m.hWindow, m.nMsgID,;
        m.wParam, m.lParam)

    WITH THIS
        DO CASE
        CASE .EventId=NM_CLICK
            .oNMLink.FromPtr(m.lParam)
            .OnClick
        ENDCASE
    ENDWITH

PROCEDURE OnClick  && abstract
```

```
PROCEDURE DisplayObject
    LOCAL nStyle, hApp, cSizeBuffer

    WITH THIS
        .DestroyObject
*!*             nStyle = BITOR(WS_VISIBLE, WS_CHILD,;
*!*                 LWS_NOPREFIX, LWS_TRANSPARENT,;
*!*                 LWS_USEVISUALSTYLE)

        nStyle = BITOR(WS_VISIBLE, WS_CHILD)

        hApp = GetWindowLong(.hParentWindow,
GWL_HINSTANCE)

        .hWindow = CreateWindow(0, "SysLink",;
            .LinkCaption, nStyle, .Left, .Top,;
            .Width, .Height,;
            .hParentWindow, .CtrlId, hApp, 0)

        cSizeBuffer=REPLICATE(CHR(0), 8)

        = SendMessageStr(.hWindow, LM_GETIDEALSIZE,;
            .Width, @cSizeBuffer)

        .Height=buf2dword(SUBSTR(cSizeBuffer,5,4))

        = SetWindowPos(.hWindow, HWND_TOP,;
            0, 0, .Width, .Height,;
            BITOR(SWP_NOMOVE, SWP_SHOWWINDOW))
    ENDWITH

ENDDEFINE

DEFINE CLASS NMHdrStruct As Relation
    hwndFrom=0
    idFrom=0
    EventId=0

PROCEDURE FromPtr(nAddr As Number)
    LOCAL cBuffer
    cBuffer = REPLICATE(CHR(0), NMHDR_SIZE)
    MemToStr(@cBuffer, m.lParam, NMHDR_SIZE)
    WITH THIS
        .hwndFrom = buf2dword(SUBSTR(m.cBuffer,1,4))
        .idFrom = buf2dword(SUBSTR(m.cBuffer,5,4))
        .EventId = buf2dword(SUBSTR(m.cBuffer,9,4))
    ENDWITH

ENDDEFINE

DEFINE CLASS oNMLinkStruct As Relation
    NMHdr=NULL
    litem_iLink=0
    litem_szID=""
    litem_szUrl=""
```

```
PROCEDURE FromPtr(nAddr As Number)
    LOCAL cBuffer
    cBuffer = REPLICATE(CHR(0), NMLINK_SIZE)
    MemToStr(@cBuffer, m.lParam, NMLINK_SIZE)
    WITH THIS
        .NMHdr=CREATEOBJECT("NMHdrStruct")
        .NMHdr.FromPtr(m.nAddr)
        .litem_iLink = buf2dword(SUBSTR(m.cBuffer,17,4))
        .litem_szID = .GetStr(@cBuffer, 29,
MAX_LINKID_TEXT)
        .litem_szUrl = .GetStr(@cBuffer,
29+MAX_LINKID_TEXT*2,;
            L_MAX_URL_LENGTH)
    ENDWITH

PROTECTED FUNCTION GetStr(cBuffer, nStart, nLength) As
String
    LOCAL cResult
    cResult=STRCONV(SUBSTR(m.cBuffer, m.nStart,
m.nLength*2), 6)
RETURN STRTRAN(m.cResult, CHR(0), "")

ENDDEFINE

FUNCTION buf2dword(cBuffer)
RETURN Asc(SUBSTR(cBuffer, 1,1)) + ;
    BitLShift(Asc(SUBSTR(cBuffer, 2,1)),  8) +;
    BitLShift(Asc(SUBSTR(cBuffer, 3,1)), 16) +;
    BitLShift(Asc(SUBSTR(cBuffer, 4,1)), 24)
```

7. **Windows Shell** . Composante de l'API Windows permet aux applications d'accéder aux fonctionnalités fournies par le shell du système d'exploitation, ainsi que modifier et l'améliorer. La composante réside dans shell.dll sur Windows 16 bits, et shell32.dll sur Windows 32 bits. Voici un exemple de son utilisation :

```
LOCAL form
form = CreateObject("Tform")
oForm.Show(1)
* end of main

DEFINE CLASS Tform As Form
   Width=480
   Height=300
   BorderStyle=2
   MaxButton=.F.
   MinButton=.F.
   Caption="Control Panel Functions"
   Autocenter=.T.

   ADD OBJECT lst As ListBox WITH;
```

```
   Left=8, Top=5, Width=464, Height=210

   ADD OBJECT Label1 As Label WITH;
   Left=10, Top=228, Caption="Command:", Autosize=.T.

   ADD OBJECT txtCmd As TextBox WITH;
   Left=80, Top=226, Height=24, Width=384, ReadOnly=.T.

   ADD OBJECT cmdRun As CommandButton WITH;
   Left=382, Top=256, Height=27, Width=80, Caption="Run"

PROCEDURE Init
   THIS.lst.InteractiveChange

PROCEDURE lst.Init
   ThisForm.AddCmd("System, General property page",
"shell32.dll,Control_RunDLL sysdm.cpl,,0")
   ThisForm.AddCmd("Desktop property page",
"shell32.dll,Control_RunDLL desk.cpl,,0")
   ThisForm.AddCmd("Add New Printer wizard",
"shell32.dll,SHHelpShortcuts_RunDLL AddPrinter")
   ThisForm.AddCmd("Add Hardware Wizard",
"shell32.dll,Control_RunDLL hdwwiz.cpl")
   ThisForm.AddCmd("Install/Uninstall tab selected",
"shell32.dll,Control_RunDLL appwiz.cpl,,1")
   ThisForm.AddCmd("Set Date & Time properties tab",
"shell32.dll,Control_RunDLL timedate.cpl")
   ThisForm.AddCmd("Fonts Folder in Explorer view",
"shell32.dll,SHHelpShortcuts_RunDLL FontsFolder")
   ThisForm.AddCmd("Internet Properties, General Tab",
"shell32.dll,Control_RunDLL inetcpl.cpl")
   ThisForm.AddCmd("Mouse Properties",
"shell32.dll,Control_RunDLL main.cpl @0")
   ThisForm.AddCmd("Keyboard Properties, Speed tab",
"shell32.dll,Control_RunDLL main.cpl @1")
   ThisForm.AddCmd("Multimedia/Audio property page",
"shell32.dll,Control_RunDLL mmsys.cpl,,0")

   WITH THIS
         .RowSourceType=2
         .RowSource="csCommands"
         .ListIndex=1
   ENDWITH

PROCEDURE lst.InteractiveChange
   ThisForm.txtCmd.Value = "rundll32.exe " +;
         ALLTRIM(csCommands.params)

PROCEDURE AddCmd(cName, cParams)
   IF Not USED("csCommands")
         CREATE CURSOR csCommands (cmdname C(50), params
C(100))
   ENDIF
   INSERT INTO csCommands VALUES (" "+m.cName, m.cParams)

PROCEDURE cmdRun.Click
```

```
   ThisForm.RunCmd1("rundll32.exe",
ALLTRIM(csCommands.params))

PROCEDURE RunCmd1(cApp, cParam)
    DECLARE INTEGER ShellExecute IN shell32;
         INTEGER hwnd, STRING lpOperation,;
         STRING lpFile, STRING lpParameters,;
         STRING lpDirectory, INTEGER nShowCmd
   = ShellExecute(0, "open", cApp, cParam, "", 1)

PROCEDURE RunCmd2(cApp, cParam)
   DECLARE INTEGER WinExec IN kernel32;
         STRING lpCmdLine, INTEGER nCmdShow
   = WinExec(cApp + " " + cParam, 1)
ENDDEFINE
```

8. **Network Services (**services de réseau) Donner accès à des capacités de réseautage différents du système d'exploitation. Ses sous-composants comprennent NetBIOS, Winsock, NetDDE, RPC et beaucoup d'autres. Vous pouvez trouver des exemples de Winsock avec vfpwinsock ici : **http://www.vfpwinsock.com/**
9. On peut aussi retrouver d'autres DLL qui peuvent être considérés comme DLL de base de Windows, même si ils ne le sont pas. En voici des exemples :
   a. shdocvw.dll, mshtml.dll, msxml*.dll, urlmon.dll (pour le web browser)
   b. Pour le multimedia, tous les DLL de DirectX.

Exemples : RawInput.prg

SetLayeredWindowAttribute.prg

SHBrowseForFolder.prg

iswow64process.prg

getnameinfo.prg

compression.PRG

roach.prg

TexttoImages.prg

Position.prg

Extendedmessagebox.prg

AUTOMATION AVEC VISUAL FOXPRO.

Il existe de nombreux logiciels qui peuvent être automatisées. Voici une liste d'entre eux et leur méthode d'appel.

AccPac Advantage Corporate: **oAccPac=CreateObject('ACCPAC.xapiSession')**
AccPac Report Master for Windows:
oImpApp=CreateObject("Impromptu.Application.30")
Bar Tender: **oBarTender = CreateObject("BarTender.Application")**
Crystal Reports: **oCRApplication = createobject ("CrystalRuntime.Application")**
This doesn't invoke an IDE actually.
EUDORA: **oEud = CreateObject("Eudora.EuApplication.1")**
FaxMaker: **oFax = CREATEOBJECT("fmfaxapi.application")**
File System Object **oFSO = CREATEOBJECT("Scripting.FileSystemObject")** (no IDE)
Group Wise **oGroupWise = CreateObject("NovellGroupWareSession")**
LotusNotes: **oNotes = CreateObject("Notes.NotesSession")**
oNotes = CreateObject("Notes.NotesUIWorkspace")
oNotes = CreateObject("Lotus.Notessession") (Domino 5.0.3)
MS Access: **oAccess = CreateObject("Access.Application")**
MS Common Dialog: **oCommmonDialog = CreateObject("MSComDlg.CommonDialog") && !!! You can't directly create this object without a development licence, so for dynamically doing it on another machine, see below.** -- Peter Crabtree
MSExcel: **oExcel = CreateObject("Excel.Application")**
MS Front Page: **oFP = CreateObject("FrontPage.Application")**
MS Graph: **oGraph = CreateObject("MSGraph.Application")**
MS Internet Explorer: **oIE = CreateObject("InternetExplorer.Application")**
MS Map Point: **oMapPoint = CreateObject("MapPoint.Application")**
MSN Messenger: **oMessenger = CREATEOBJECT("MSNMessenger.MessengerApp")**
MS Net Meeting: **oNetMeeting=CREATEOBJECT("netmeeting.app.1")**
MS Outlook: **oOutlook = CreateObject("Outlook.Application")**
MS PowerPoint: **oPP = CreateObject("PowerPoint.Application")**

MS Project: **oProj = CreateObject("msProject.Application")**
MS Source Safe: **oVSS = CreateObject("SourceSafe.0")**
MS Word: **oWord = CreateObject("Word.Application")**
MS Visio: **oVisio = CreateObject("Visio.Application")**
MS Visual Basic: **?**
MS Visual CPlus Plus: **oCpp = CreateObject("MSDev.Application")**
MS Visual Foxpro: **oVFP = CreateObject("VisualFoxPro.Application")**
MS Windows Scripting Host: **oWSH = CreateObject("WScript.Shell")**
Novell Group Wise **oGroupWise = CreateObject("NovellGroupWareSession")**
PCAnywhere (host): **oPCAHost = CreateObject("WinAWSvr.BeHostDataManager")**
PCAnywhere (remote): **oPCARem = CreateObject("WinAWSvr.RemoteDataManager")**
QuickBooks: **QBSessionManager = CreateObject("QBFC2.QBSessionManager")**
Rational Rose: **oRose = CreateObject("Rose.Application")**
Snag It: **oSnag = CreateObject("SnagIt.ImageCapture.1")**
TAPIFax: **oTAPIFax = CreateObject('FaxServer.FaxServer')**
WindowsShell: **oWSH = CreateObject("Shell.Application")**
Windows Media Player:

```
oWMP = CREATEOBJECT("WMPlayer.OCX")

oPlayList = oWmp.PlaylistCollection.GetAll()

oWmp.currentPlaylist = oPlayList.Item(0) && Zero based array

*!* Music starts!
```

Windows Messenger: **oMessenger = CREATEOBJECT("Messenger.MessengerApp")**
Win Fax: **oWinFax = CreateObject("WinFax.SDKSend")**
Win Print : **oWinPrint = CreateObject("WinPrint.WinPrintX")**
SQL-DMO object: **loSQL = CREATEOBJECT("SQLDMO.Application")**
Scripting Shell: **oShell = CreateObject("WScript.Shell")**
Scripting Network Object: **oNet = CreateObject("WScript.Network")**

Scripting Regular Expression Parser: **oReg = CreateObject("VBScript.RegExp")**
SQL DMO SQL Server: **CreateObject("SQLDMO.SQLServer")**

Vous remarquerez que ces automations n'ont pas tous des interfaces. Comme CDO (Collaboration Data Object).

CDO permet d'envoyer des courriels avec ou sans attachements, en HTML ou texte etc… CDO n'est plus un 'wrapper' de MAPI comme l'était CDO1.2 mais utilise. Même si CDO est beaucoup plus simple que MAPI, il permet d'accomplir la plupart des taches requises. CDO, à comparer à CDONTS, permet d'envoyer des courriels en utilisant un service SMTP local ou un serveur SMTP à distance. CDONTS peut juste utiliser un serveur SMTP à distance avec IIS. Il existe aussi une version de CDO qui agit avec Microsoft Exchange (CDOEX).

CDO est unidirectionnel, on peut seulement envoyer des courriels. Pour recevoir des courriels, il faut utiliser autre chose comme Winsock.

Le DLL principal pour envoyer des courriels est CDOSYS.DLL.

Voici quelques exemples (vous trouverez d'autres exemples ici http://www.atoutfox.org/articles.asp?ACTION=FCONSULTER&ID=0000000040)

Comment envoyer un simple courriel :

```
oMSG = CREATEOBJECT("cdo.message")
oMSG.To = "me@nowhere.com"
oMSG.From = "me"
oMSG.Subject = "Hello Email"
oMSG.TextBody = "This is an easy way to create an email"
oMSG.Send()
```

Comment envoyer un courriel avec attachement:

```
oMSG = createobject("CDO.Message")
oMSG.To = "me@nowhere.com"
oMSG.From = "me@nowhere.com"
oMSG.Subject = "Hello Email"
```

```
oAtt=oMSG.AddAttachment('c:\myfile.txt')
```

```
&&oAtt=oMSG.AddAttachment('c:\myfile2.txt') && Pour 2 attachements
```

```
oMSG.Send()
```

Si vous recevez un message comme

CDO.Message.1 error '80040220' The "SendUsing" configuration value is invalid.

Ceci veut dire que votre ordinateur n'est pas configure avec un service SMTP. On peut soit configurer un service SMTP sur le poste local, ou utiliser le code qui suit (en spécifiant un serveur SMTP externe)

```
#Define CRLF Chr(13)+Chr(10)
#Define cdoSendUsingPickup 1
#Define  cdoSendUsingPort  2
#Define  cdoAnonymous  0
#Define cdoBasic  1 && clear text
#Define cdoNTLM  2 && NTLM
&& Delivery Status Notifications
#Define cdoDSNDefault  0 &&None
#Define cdoDSNNever  1 && None
#Define cdoDSNFailure  2 && Failure
#Define cdoDSNSuccess  4 && Success
#Define cdoDSNDelay = 8 &&Delay
#Define cdoDSNSuccessFailOrDelay  14 &&Success, failure or delay
Local objMsg,objConf,objFlds
Store Null To objMsg,objConf,objFlds
objMsg = Createobject("CDO.Message")
objConf = Createobject("CDO.Configuration")
objFlds = objConf.Fields
With objFlds
      .Item("http://schemas.microsoft.com/cdo/configuration/sendusing") =
cdoSendUsingPort
      .Item("http://schemas.microsoft.com/cdo/configuration/smtpserverport") =
2525
      .Item("http://schemas.microsoft.com/cdo/configuration/smtpserver") =
"mail.carvertechnologies.com"
      .Item("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate")
= cdoBasic
      .Item("http://schemas.microsoft.com/cdo/configuration/sendusername") =
"mike.gagnon@carvertechnologies.com"
      .Item("http://schemas.microsoft.com/cdo/configuration/sendpassword") =
"moutarde"
      .Update
Endwith

strBody = "This is a sample message." + CRLF
strBody = strBody + "It was sent using CDO." + CRLF
```

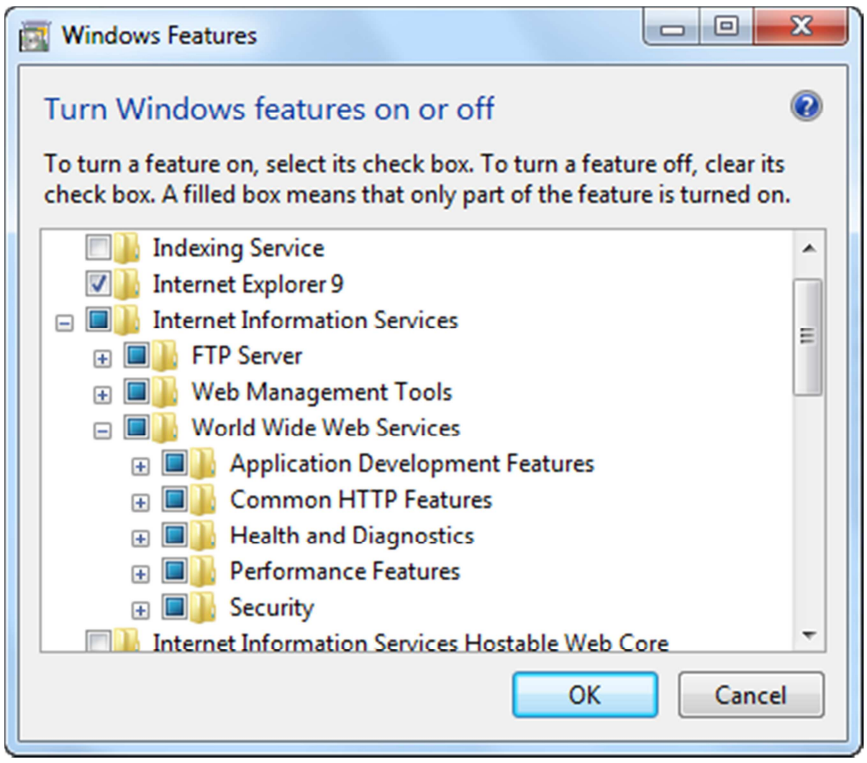
```
With objMsg
      .Configuration = objConf
      .To = "mgagnon23@hotmail.com"
      .From = "me@my.com"
      .Sender = 'Me'
      .Subject = "This is a CDO test message"
      .TextBody = strBody
&& use .HTMLBody to send HTML email.
*.Addattachment "c:\temp\Scripty.zip"
      .Fields("urn:schemas:mailheader:disposition-notification-to") =
"me@my.com"
      .Fields("urn:schemas:mailheader:return-receipt-to") = "me@my.com"
      .DSNOptions = cdoDSNSuccessFailOrDelay
      .Fields.Update
      .Send()
Endwith
```

Avec la venue de Windows 7, Microsoft a décidé d'inclure le service SMTP avec IIS 7. Qui rend la tâche d'installer un service SMTP plus difficile.

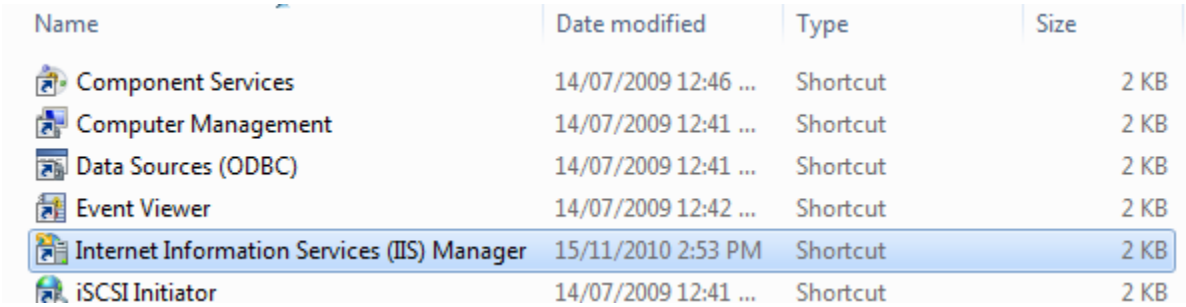
Voici les étapes :

1. Control panel-Programs and Features-Turn window features on or off

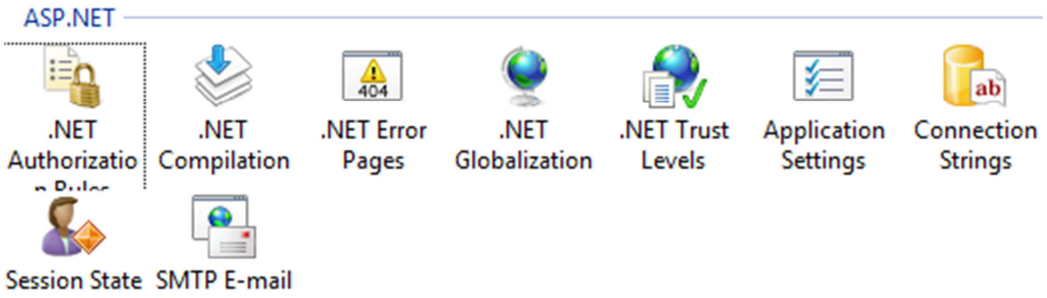
2. Cliquez sur les services requis



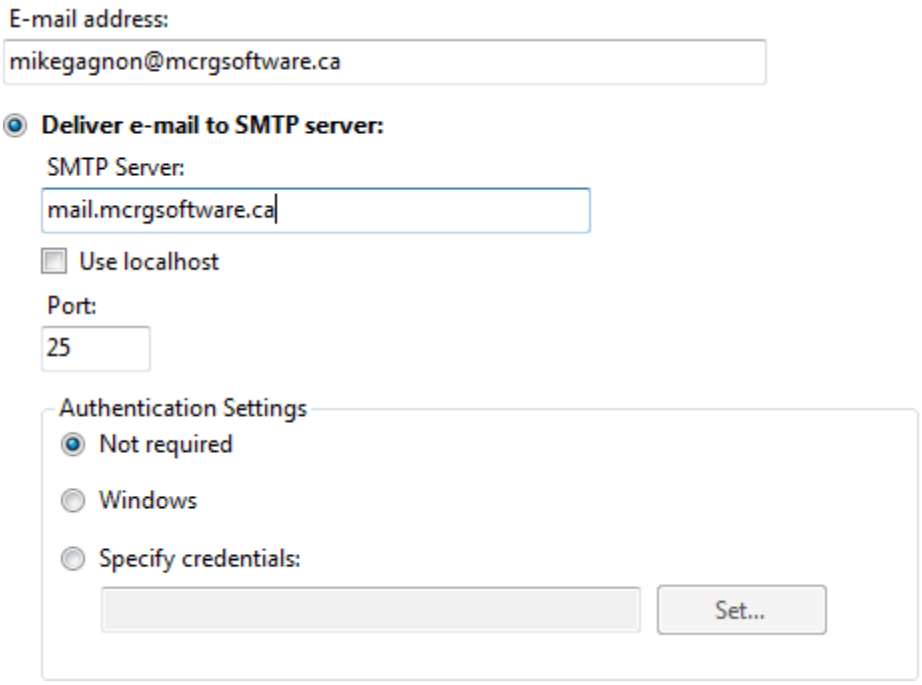
3. Control Pannel-administrative tools-Internet information Services (IIS)



4. SMTP E-mail



5. Configuration



L'avantage de configurer un service SMTP est que l'on plus à le refaire et le code utilisée pour CDO contient moins de lignes.

CDO dans sa première version, nous permettait de créer des appointments, personne , mais plus maintenant. C'est seulement disponible avec Exchange Server.

```
oApp= CREATEOBJECT("cdo.appointment")
```

CDO permet aussi de créer de documents MHTLM (page web incluant les images en binaire dans la page web elle-même),

```
Local lcFileName,lcStr && Variables locales
Declare Integer ShellExecute In "Shell32.dll" ;
INTEGER HWnd, ;
STRING lpVerb, ;
STRING lpFile, ;
STRING lpParameters, ;
STRING lpDirectory, ;
LONG nShowCmd
lcFileName = Sys(2015)+'.mht' && Nom du document final
```

```
oMSG = Createobject("CDO.Message") && Message couriel -- strictement pour la fonction suivante
oMSG.CreateMHTMLBody("http://www.microsoft.com") && Créer un document MHTML
lcStr = oMSG.getstream && Stocker le document dans une variable
lcStr.SaveToFile(lcFileName,2) && Sauvegarder le document en MHT.
ShellExecute(0,"Open",lcFileName,"","",0) && Ouvrir avec ShellExecute.
```

Comment automatiser Outlook avec FoxPro

L'association AtoutFox attire votre attention sur le respect du droit français : l'usage de cet exemple de code peut être illégal en contrevenant au respect du caractère privé de la correspondance.

Mise en situation :

Notre client nous demande de créer une piste de vérification pour tous les courriels sortant. C'est une société canadienne avec de très hautes demandes au niveau de la sécurité. Vu qu'il n'utilise pas Exchange, nous avons dû trouver un moyen avec Foxpro de satisfaire la demande de notre client. Cette méthode utilise l'interop d'Outlook, bindevevents, un DLL de Foxpro et VFPCOM.COMUTIL. Notre client nous a demande aussi que le tout soit 'invisible' à l'utilisateur. Donc nous devions utiliser l'interface d'Outlook, plutôt que l'automation.

Notre serveur COM va agir comme une piste de vérification pour certains types d'activités dans Outlook.

- Voici le code de base pour la première version de notre serveur COM. Il a une seule méthode publique nommée WriteLog qui utilise la fonction STRTOFILE VFP pour placer des informations dans un fichier journal.

```
Define Class OutTrack As Custom OlePublic
	Name = "OutTrack"
	Procedure writelog
		Lparameters tcInfo
		If Empty(tcInfo)
			tcInfo = ""
		Endif
		=Strtofile(Chr(13)+Chr(10)+Replicate("-",80)+Chr(13)+Chr(10)+"   "+Sys(0)+;
			' ' +Ttoc(Datetime())+' '+Chr(13)+Chr(10)+tcInfo;
			+Chr(13)+Chr(10),"\OUTLOOK.LOG",.T.)
	Endproc
Enddefine
```

Et pour tester ce DLL dans FoxPro, on utilise le code suivant.

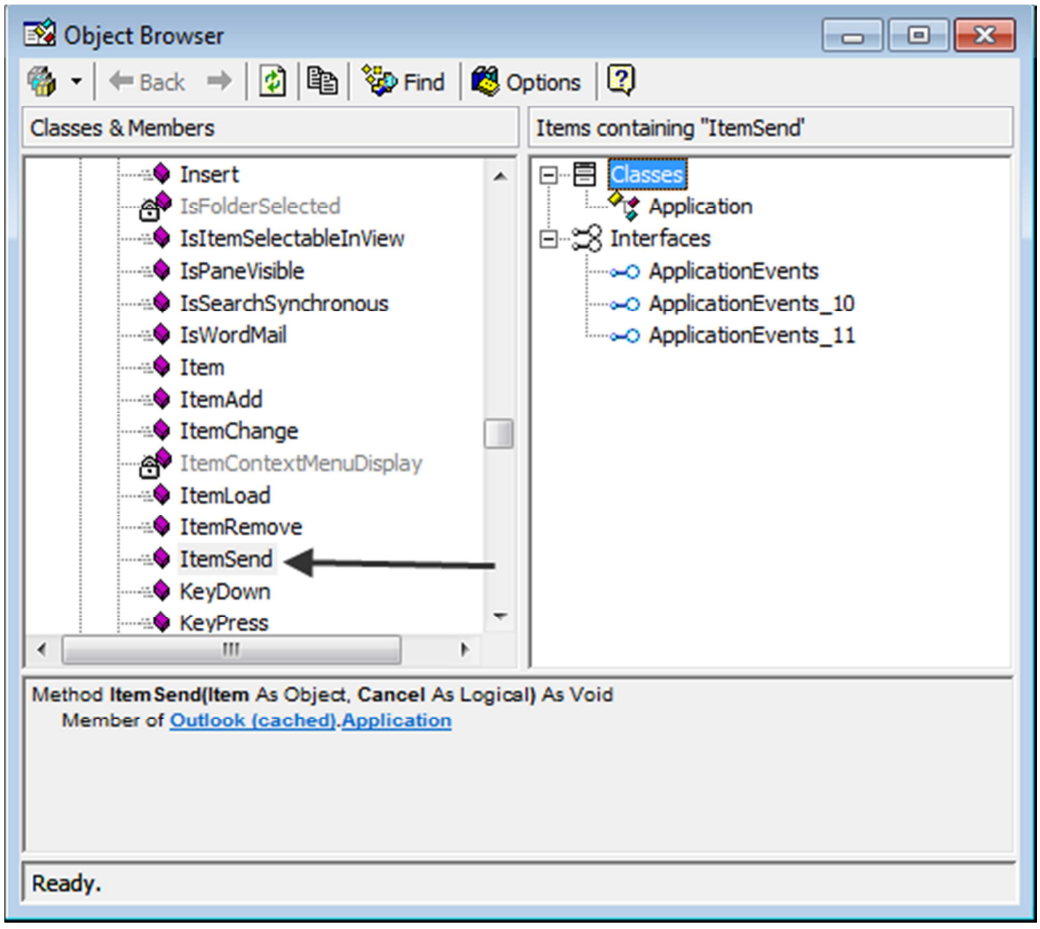
```
X = createobject('Outcome.OutTrack')

x.WriteLog("MA PISTE")
```

Vous devriez obtenir un résultat comme ceci dans un fichier texte.

MIKEGAGNON-PC # MikeGagnon 04/08/12 10:55:12 AM

MA PISTE

- Ensuite nous devons créer une application qui va agir en tant qu'interop entre Foxpro et Outlook, pour nous permettre d'intercepter les méthodes d'Outlook.
  Pour découvrir quelle méthode nous devons utiliser, on peut utiliser 'Object Browser' dans Foxpro, qui nous permet de voir toutes les méthodes accessibles. La méthode qui nous intéresse est Items end().



Donc voici le petit programme qui va interagir avec Outlook pour intercepter les courriels envoyés depuis un poste et faire appel à notre DLL pour écrire dans un fichier LOG le contenu du courriel.

```
#Define VFPCOM_CLSID  'VFPCOM.COMUTIL'
#Define OUTLOOK_CLSID  'OUTLOOK.APPLICATION'
#DEFINE OUTCOME_TRACKER 'OUTCOME.OUTTRACK'
#define CRLF chr(13)+chr(10)
Public goVFPCOM, goOutlook, goLink,goTrack
goVFPCOM = Create(VFPCOM_CLSID)
goOutlook = Create(OUTLOOK_CLSID)
goTrack = createobject(OUTCOME_TRACKER)
goLink = Create('OutlookApplicationEvents')
goVFPCOM.BindEvents(goOutlook, goLink)
read events

Define Class OutlookApplicationEvents As Custom
        Procedure ItemSend(Item,Cancel)
    goTrack.WriteLog("Courriel envoye a: "+Item.to+CRLF+ Item.Body)
        Endproc
Enddefine
```

Pour le but de l'exercice nous allons seulement utiliser la méthode ItemSend(). Mais toute autre méthode peut-être aussi utilisée. Donc nous créons une instance de VFPCOM, d'Outlook et notre DLL.
Donc tous les courriels qui vont être envoyés du poste, vont être enregistre dans notre fichier Outlook.log, sans que l'utilisateur n'y voit une différence.

P.S. À la demande du client, nous avons créé un Windows service do programme ci-haut pour qu'il puisse démarrer lorsque l'utilisateur part son ordinateur.

Automatiser OWC (office web components) avec FoxPro.

Avec la venue de BI (Business Intelligence), les utilisateurs nous demandent tous des chartes de toutes les dimensions, couleurs, flexibilité. Qui peuvent être utilisées sur des formes des rapports etc… Il existe plusieurs outils à notre disposition, comme MS Graph jusqu'au GDI (FoxCharts de mon ami César Shalom en est un bon exemple). Chacune de ces méthodes ont leurs avantages et leurs désavantages. MS Graph est un composant trop simple pour les besoins réels de BI, mais par contre OWC offre un peu plus de flexibilité. Premièrement il nous offre de sauvegarder le résultat de la charte en image de plusieurs formats (GIF, JPG, and PNG). On peut aussi créer des chartes tridimensionnelles, changer la luminosité etc...

Un des avantages avec OWC est comme on le sait tous, ce n'est pas tous les programmeurs dans le monde de Foxpro qui utilise FoxPro version 9. OWC par contre fonctionne à partir de Foxpro 5. FoxCharts requiert la Foxpro version 9 SP2.

Les Composantes Office Web ont été abandonnées dans Office 2007, et ne sont pas inclus, sauf une partie d'Office Project Server 2007. [7] Toutefois, ils seront toujours disponibles pour téléchargement sur le site de Microsoft. Microsoft n'a pas encore offert un remplacement complet pour les Office Web Components. Donc pour l'instant c'est un activex utilisable pour créer des chartes, tables pivot etc… Vous pouvez trouver le modèle de OWC ici (http://msdn.microsoft.com/en-us/library/aa166512(v=office.10)) et vous pouvez telecharge OWC ici http://www.microsoft.com/en-us/download/details.aspx?id=22276

Ce composant nous permet de créer :

- Tableur
- Chartspace
- Table pivot dynamique.
- Composant de sources de donnees.

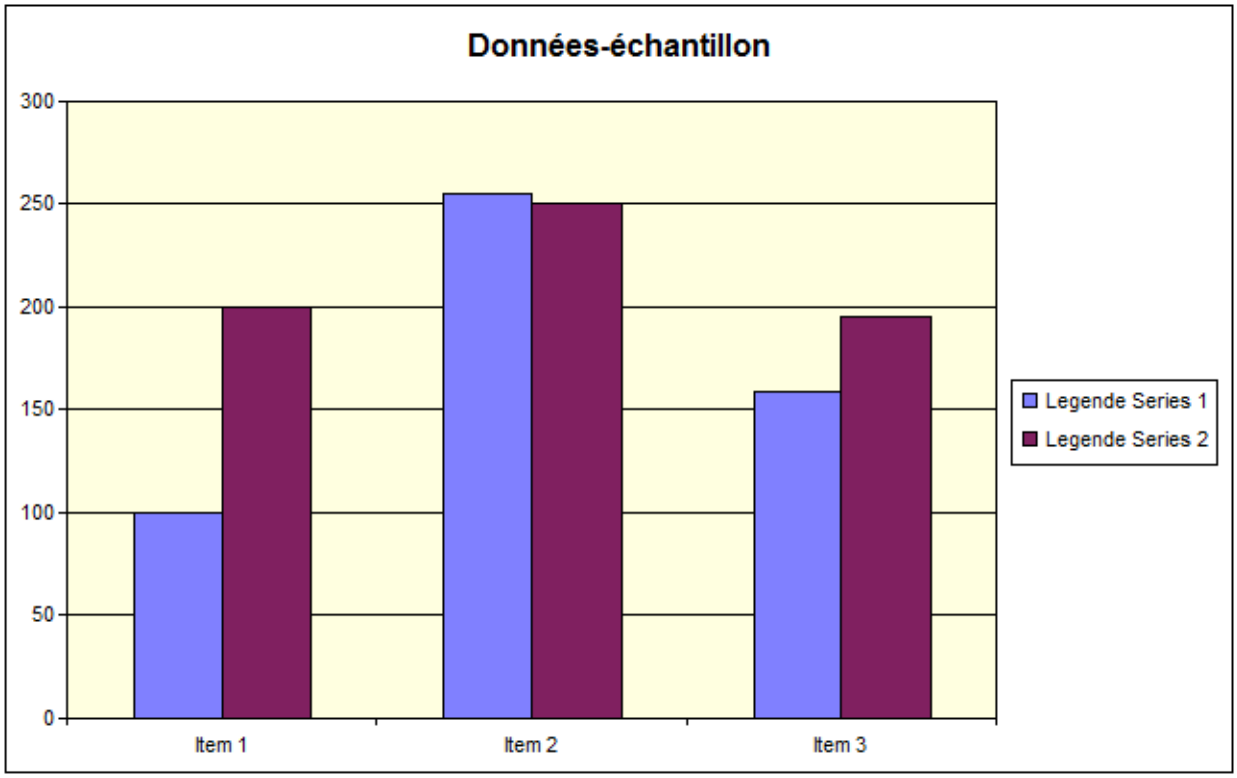
Voici un exemple simple de créer une charte avec OWC.

```
*** Data to render
local lcAction,lcFile
DECLARE INTEGER ShellExecute IN shell32.dll ;
  INTEGER hndWin, ;
  STRING cAction, ;
  STRING cFileName, ;
  STRING cParams, ;
  STRING cDir, ;
  INTEGER nShowWin
DIMENSION laLabels[3]
DIMENSION laData[3]
DIMENSION laData2[3]
laLabels[1] = "Item 1"
laLabels[2] = "Item 2"
laLabels[3] = "Item 3"
laData[1] = 100
laData[2] = 255
laData[3] = 159
laData2[1] = 200
laData2[2] = 250
laData2[3] = 195
loChartSpace = CREATEOBJECT("OWC11.ChartSpace")
oConst = loChartSpace.Constants
loGraph = loChartSpace.Charts.Add()
loGraph.Type = 0   && Column Graph
loGraph.HasLegend = .T.
loGraph.PlotArea.Interior.Color = "LightYellow"
loChartSpace.HasChartSpaceTitle = .T.
loChartSpace.ChartSpaceTitle.Caption = "Données-échantillon"
loChartSpace.ChartSpaceTitle.Font.Bold = .T.
loGraph.SeriesCollection.Add()
loGraph.SeriesCollection(0).Caption = "Legende Series 1"
loGraph.SeriesCollection(0).SetData(oConst.chDimCategories,
oConst.chDataLiteral, @laLabels)
```

```
loGraph.SeriesCollection(0).SetData(oConst.chDimValues, oConst.chDataLiteral,
@laData)
loGraph.SeriesCollection.Add()
loGraph.SeriesCollection(1).Caption = "Legende Series 2"
loGraph.SeriesCollection(1).SetData(oConst.chDimValues, oConst.chDataLiteral,
@laData2)
lcFile = SYS(2023) + "\" + SYS(2015) + ".gif"
loChartSpace.ExportPicture(lcFile, "gif",;
                           640,400)
cAction = "open"
ShellExecute(0,cAction,lcFile,"","",1)
```

Qui nous donne un charte comme ceci.

ky



Automatiser Skype avec FoxPro.

Avec la venue de Skype et VoIP (voice-over-IP), la téléphonie a changée et il est maintenant possible de faire des appels téléphoniques ‘gratuitement’ avec ce type d’outil.

Pour pouvoir automatiser Skype il nous faut 2 choses. Skype lui-meme et un dll appele Skype4Com.dll. On peut trouver la plus recente version de Skype ici http://www.skype.com/intl/en/get-skype/on-your-computer/windows/downloading et Skype4Com.dll ici http://developer.skype.com/accessories/skype4com

1. Pour placer un appel (message test) avec skype et foxpro.

```
      Public oform1
      oform1=Newobject("form1")
oform1.Show
Return


Define Class form1 As Form


      DoCreate = .T.
      Caption = "Essai de Skype"
      skypeaccount = "your skype account"
      oleskype = "null"
      ocall = "null"
      Name = "Form1"


      Add Object text1 As TextBox With ;
      Value = "echo123", ;
      Height = 25, ;
      Left = 144, ;
      Top = 66, ;
      Width = 157, ;
      Name = "Text1"


      Add Object label1 As Label With ;
      AutoSize = .T., ;
      Caption = "Number to call", ;
      Height = 17, ;
      Left = 48, ;
      Top = 68, ;
      Width = 82, ;
      Name = "Label1"


      Add Object commandgroup1 As CommandGroup With ;
      AutoSize = .F., ;
      ButtonCount = 2, ;
      BorderStyle = 0, ;
      Value = 1, ;
      Height = 37, ;
      Left = 110, ;
      Top = 144, ;
      Width = 154, ;
      Name = "Commandgroup1", ;
      Command1.AutoSize = .F., ;
      Command1.Top = 5, ;
      Command1.Left = 5, ;
      Command1.Height = 27, ;
      Command1.Width = 71, ;
      Command1.Caption = "Place call", ;
      Command1.Name = "Command1", ;
      Command2.AutoSize = .F., ;
      Command2.Top = 5, ;
```

```
	Command2.Left = 78, ;
	Command2.Height = 27, ;
	Command2.Width = 71, ;
	Command2.Caption = "End call", ;
	Command2.Name = "Command2"


	Procedure Init
	Thisform.oleskype = Createobject("Skype4COM.Skype", "Skype")
	If Thisform.oleskype.Client.IsRunning
		Messagebox("Skype is already started ")
		Thisform.oleskype.User(Thisform.skypeaccount)
	Else
		Messagebox("You have to start Skype")
		Return
	Endif
	Thisform.oleskype.Attach()
	Endproc


	Procedure commandgroup1.Command1.Click
	Try
		Thisform.ocall = Thisform.oleskype.PlaceCall(Thisform.text1.Value)
	Catch To loerror
	Endtry
	Endproc


	Procedure commandgroup1.Command2.Click
	Thisform.ocall.finish()
	Endproc


Enddefine
```

2. Comment envoyer un message SMS avec Skype

```
	LOCAL loSkype as skype4com.skype, lcPhoneNr as String, lcMessage
as String, lnStatus as number

lnStatus = 0

loSkype = CreateObject("Skype4COM.Skype", "Skype_")
lcPhoneNr = "+15146913876"
lcMessage = "this is a test"

If Not loSkype.Client.IsRunning Then
   do while Not loSkype.Client.IsRunning
      loSkype.Client.Start()
   enddo
EndIf

loSkype.SendSms( lcPhoneNr, lcMessage)

INKEY(.5)

?loSkype.Convert.SmsMessageStatusToText(lnStatus)
```

Trucs et astuces.

Foxpro VS SQL.

3. Nous avons un logiciel de ressources humaines et paie qui est développer en Visual Foxpro, et qui peux rouler avec des tables Foxpro ou MSSQL ou Oracle.
   Une des problèmes qui arrive souvent est que le code qu'un programmeur habitue à programmer en FoxPro font des erreurs de programmation très simples, qui fonctionne correctement en FoxPro mais lorsque que l'on change de base de données, et que l'on utilise SQL par exemple ces erreurs sont inacceptables. Il est donc suggérer de test votre code avec les deux environnements avec de livrer à votre client, comme cela vous n'aurez pas de surprises désagréables.
   En voici quelques exemples.
   Dans Foxpro ont peux utiliser ceci sans problème :

   ```
   select * from pers where e_persid => 1000
   ```

   Mais lorsque vous lorsque vous vous retournez vers une base de données SQL et testez le même code, vous allez obtenir le message d'erreur suivant :

   ```
   Msg 102, Level 15, State 1, Line 1
   Incorrect syntax near '>'.
   ```

   Vu que Foxpro accepte soit :

   ```
   select * from pers where e_persid => 1000
   ```
   Ou
   ```
   select * from pers where e_persid <= 1000
   ```

   Et que SQL n'accepte que
   ```
   select * from pers where e_persid <= 1000
   ```

   La conclusion est d'utiliser la deuxième requete, pour que les deux environnements fonctionne correctement. Il est toujours bon de tester les deux environnements.

4. La fonction EMPTY()
   Pour tester un champs vide tout le monde connait la fonction EMPTY(), mais cette fonction peut cree des problèmes lorsqu'on l'utilise sur un base de donnees SQL sur un champs varchar (champs Memo dans Foxpro. Par

exemple avec Foxpro si on veux determiner si un champs memo est vide un peux utiliser ceci sans problème :

```
select qJobhist
if empty(qJobhist.h_champsmemo)  && .T.
```

Mais quand vient le temps de creer un curseur d'une table SQL (avec le meme champs Varchar), Foxpro dans notre requete nous retourne un champs memo du meme champs, mais tout d'un coup le code ne fonctionne plus :

```
select qJobhist
if empty(qJobhist.h_champsmemo)  && .F.
```

Et pourquoi? Parce qu'en fait il n'est techniquement pas vide, il y s'y trouve un caractère (invisible) qui rendu la vérification à fausse. Donc une des solutions est ceci (il y a plusieurs autres, mais moi j'utilise celle-ci) :

```
if empty(qJobhist.h_champsmemo+'')  && .T.
```

Qui me donne la bonne réponse, dans les deux environnements (SQL et Foxpro). Je répète tester votre code dans les deux environnements.

5. Comment montrer la calculatrice de Windows à l'intérieure de votre application. Un vieux truc mais toujours bon à savoir :

```
#Define SW_SHOWNORMAL  1

Declare Integer GetActiveWindow In user32
Declare Integer GetDesktopWindow In user32

Declare Integer FindWindow In user32;
STRING lpClassName, String lpWindowName

Declare Integer WinExec In kernel32;
STRING lpCmdLine, Integer nCmdShow

Declare Integer SetParent In user32;
INTEGER hWndChild, Integer hWndNewParent
Local hCalc

Do While .T.
        hCalc = FindWindow (.Null., "Calculator")
        If hCalc = 0
                = WinExec ("calc.exe", SW_SHOWNORMAL)
        Else
                Exit
        Endif
Enddo
```

6. Comment créer une forme ronde (pour un mot de passe par exemple et on peut y mettre une image de fonds comme le logo de la société de votre client).

```
Declare Integer GetFocus In user32
Declare Integer DeleteObject In gdi32 Integer hObject
Declare Integer GetSystemMetrics In user32 Integer nIndex
Declare Integer ReleaseCapture In user32

Declare Integer SendMessage In user32;
INTEGER HWnd, Integer Msg,;
INTEGER wParam, Integer Lparam

Declare Integer CreateEllipticRgn In gdi32;
INTEGER nLeftRect, Integer nTopRect,;
INTEGER nRightRect, Integer nBottomRect

Declare Integer SetWindowRgn In user32;
INTEGER HWnd, Integer hRgn, Integer bRedraw
Public frm
frm = Createobject ("myform")
frm.Visible = .T.

Define Class Myform As Form
        #Define badgeDiameter 264
        #Define topMargin 4
        #Define leftMargin 2

        Width = 300
        Height = 350
        AutoCenter = .T.
        hRgn=0

        Add Object lbl As Label With;
        Caption="Votre mot de passe:", FontName="Arial", FontSize=14,;
        Bold=.T., ForeColor=Rgb(0,0,0), BackStyle=0,;
        Alignment=2, Left=45, Width=200, Top=105, Height=25

        Add Object txt As TextBox With;
        Width=100, Height=24, Left=82, Top=130,;
        PasswordChar="*"

        Add Object cmd As CommandButton With;
        Width=60, Height=25, Left=104, Top=165,;
        Caption="Ok", Default=.T.

        Procedure Activate
        This.RegionOn

        Procedure RegionOn
        #Define SM_CYSIZE  31
        #Define SM_CXFRAME 32
        #Define SM_CYFRAME 33
```

```
		If This.hRgn <> 0
			Return  && the region is already set
		Endif

		Local HWnd, x0, y0, x1, y1
		x0 = GetSystemMetrics (SM_CXFRAME) + leftMargin
		y0 = GetSystemMetrics (SM_CYSIZE) +;
		GetSystemMetrics (SM_CYFRAME) + topMargin
		x1 = x0 + badgeDiameter
		y1 = y0 + badgeDiameter
		This.hRgn = CreateEllipticRgn (x0, y0, x1, y1)
		HWnd = GetFocus()
		If SetWindowRgn (HWnd, This.hRgn, 1) = 0
			= DeleteObject (This.hRgn)
			This.hRgn = 0
		Endif
*** ----------------------------
		Endproc

		Procedure  MouseDown
		Lparameters nButton, nShift, nXCoord, nYCoord
		#Define WM_NULL         0
		#Define WM_SYSCOMMAND   274  && 0x112
		#Define MOUSE_MOVE    61458  && 0xF012

		If nButton = 1
			= ReleaseCapture()
			= SendMessage (GetFocus(), WM_SYSCOMMAND,
MOUSE_MOVE, WM_NULL)
		Endif

		Procedure cmd.Click
		Thisform.Release
Enddefine
```

7. Comment forcer un courriel de quitter le 'outbox' d'Outlook. Dans certain cas le courriel ne veux pas se faire envoyer.

```
oOutlook = GETOBJECT(,"Outlook.Application")
oNameSpace= oOutlook.GetNamespace("MAPI")
oFolder = oNameSpace.GetDefaultfolder(4)
oSend = oFolder.Items(1).Send
```

8. Comment creer un GUID.

```
* -- returns a Globally Unique IDentifier (GUID)

local cData1, cData2, cData3, cData4, cData5, cDataAll
private PGuid

store "" to cData1, cData2, cData3, cData4, cData5, cDataAll

* -- declare external function
DECLARE INTEGER CoCreateGuid IN OLE32.DLL STRING @pGuid

* -- Initialize the buffer that will hold the GUID with nulls
pGuid = REPLICATE(CHR(0),17)

* -- Call CoCreateGuid

IF CoCreateGuid(@pGuid) = 0   && success

  * -- Store the first eight characters of the GUID in data1
  cData1 = RIGHT(TRANSFORM(strtolong(LEFT(pGuid,4)),"@0"),8)

  * -- Store the first group of four characters of the GUID in data2
  cData2 = RIGHT(TRANSFORM(strtolong(SUBSTR(pGuid,5,2)),"@0"),4)

  * -- Store the second group of four characters of the GUID in data3
  cData3 = RIGHT(TRANSFORM(strtolong(SUBSTR(pGuid,7,2)),"@0"),4)
```

```
  * -- Store the third group of four characters of the GUID in data4
  cData4 = RIGHT(TRANSFORM(strtolong(SUBSTR(pGuid,9,1)),"@0"),2) + ;
    RIGHT(TRANSFORM(strtolong(SUBSTR(pGuid,10,1)),"@0"),2)

  * -- Initialize data5 to a null string
  cData5 = ""

  * -- Convert the final 12 characters of the GUID and store in data5
  FOR nGuidLen = 1 TO 6

cData5=cData5+RIGHT(TRANSFORM(strtolong(SUBSTR(pGuid,10+nGuidLen,1))),2)
  ENDFOR

  * -- Check the length of data5. If less than 12, the final 12-len(data5)
  * -- characters are '0'
  IF LEN(cData5) < 12
    cData5=cData5+REPLICATE("0",12-LEN(cData5))
  ENDIF

  * -- Assemble the GUID into a string
  cDataAll = cData1+"-"+cData2+"-"+cData3+"-"+cData4+"-"+cData5

ENDIF


* -- Done with the call to CoCreateGuid, so clear the DLLs from memory
CLEAR DLLS


? cDataAll


**********

FUNCTION strtolong
* -- Passed:  4-byte character string (lcLongstr) in low-high ASCII format
* -- Returns:  long integer value
* -- Example:
* -- m.longstr = "1111"
* -- m.longval = strtolong(m.longstr)
PARAMETERS lcLongstr
LOCAL lnByte, lnRetval
lnRetval = 0
FOR lnByte = 0 TO 24 STEP 8
  lnRetval = lnRetval + (ASC(lcLongstr) * (2^lnByte))
  lcLongstr = RIGHT(lcLongstr, LEN(lcLongstr) - 1)
NEXT
RETURN lnRetval
```

9. Comment éteindre un ordinateur (toute versions jusqu'à Windows 7)
   shutdownwin(.t.,.t.)

```
procedure shutdownwin
LPARAMETERS tlShutdownRequested, tlInteractiveShutdown

#DEFINE SE_SHUTDOWN_NAME "SeShutdownPrivilege"   && Privilege to shut down
windows
#DEFINE SE_PRIVILEGE_ENABLED 2              && Enable privilege flag
#DEFINE TOKEN_QUERY 2                    && Token may query status
#DEFINE TOKEN_ADJUST_PRIVILEGE 0x20          && Token may adjust privileges
#DEFINE EWX_SHUTDOWN 1
#DEFINE EWX_REBOOT 2                     && Initiate a reboot
#DEFINE EWX_FORCE 4                      && Force processes to close
#DEFINE SIZEOFTOKENPRIVILEGE 16

*  Windows Shutdown API - all platforms
DECLARE ExitWindowsEx IN WIN32API INTEGER uFlags, INTEGER dwReserved &&
API call to shut down Windows

*  Check the OS to see if we need to request the privilege
IF  ('4.0' $ OS() OR '5.0' $ OS() OR 'NT' $ OS())  OR '6.0' $ OS()
  *  API Calls below are NT/2K specific or only needed to manipulate process
permissions

  *  Retrieve the LUID of a specific privilege name - changes each time Windows
restarts
  DECLARE SHORT LookupPrivilegeValue IN ADVAPI32 ;
    INTEGER lpSystemName, ;
    STRING @ lpPrivilegeName, ;
    STRING @ pluid

  *  Get an hToken with specific permission sets for a given process handle
  DECLARE SHORT OpenProcessToken IN Win32API ;
    INTEGER hProcess, ;
    INTEGER dwDesiredAccess, ;
    INTEGER @ TokenHandle

  *  Alter the privileges granted to a Process via a specific hToken
  DECLARE INTEGER AdjustTokenPrivileges IN ADVAPI32 ;
    INTEGER hToken, ;
    INTEGER bDisableAllPrivileges, ;
    STRING @ NewState, ;
    INTEGER dwBufferLen, ;
    INTEGER PreviousState, ;
    INTEGER @ pReturnLength

  *  Get the Process handle for this process
  DECLARE INTEGER GetCurrentProcess IN WIN32API

  LOCAL cLUID, nhToken, cTokenPrivs, nFlag

  cLUID = REPL(CHR(0),8)  && 64 bit Locally Unique IDentifier for the Privilege

  IF LookupPrivilegeValue(0, SE_SHUTDOWN_NAME, @cLUID) = 0
    RETURN .F.  &&  Privilege not defined for this process
```

```
  ENDIF

  nhToken = 0  &&  Process Token to be used to manipulate process privilege set

  IF OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY +
TOKEN_ADJUST_PRIVILEGE , @nhToken) = 0
    RETURN .F.  &&  OS will not grant handle with necessary rights
  ENDIF

  *  Create a TOKEN_PRIVILEGES structure, a 4 byte DWORD indicating #
permissions,
  *  followed by an array of 8 byte LUIDs and the 4 byte DWORD permission
attributes
  *  for it.  One privilege, the LUID retrieved for shutdown, enable attribute set
  cTokenPrivs = CHR(1) + REPL(CHR(0),3) + cLUID + CHR(SE_PRIVILEGE_ENABLED) +
REPL(CHR(0), 3)
  IF AdjustTokenPrivileges(nhToken, 0, @cTokenPrivs, SIZEOFTOKENPRIVILEGE, 0, 0)
= 0
    RETURN .F.  && Privilege denied
  ENDIF
ENDIF

CLOSE ALL    &&  Start shutting down VFP tables
FLUSH        &&  Suggest that the OS flush all buffers
CLEAR EVENTS &&  Avoid VFP being obstinate
ON SHUTDOWN  &&  don't try to run anything extra
*  Select shutdown flags
DO CASE
CASE tlShutdownRequested AND tlInteractiveShutdown
  nFlag = EWX_SHUTDOWN
CASE tlShutdownRequested
  nFlag = EWX_SHUTDOWN + EWX_FORCE
CASE tlInteractiveShutdown
  nFlag = EWX_REBOOT
OTHERWISE
  nFlag = EWX_REBOOT + EWX_FORCE
ENDCASE
=ExitWindowsEx(nFlag, 0)  && Force a reboot to be initiated
QUIT  &&  Start an orderly shutdown of VFP
```

10. Fontionalitées de _CLIPTEXT.
    Pour de nombreux programmeurs de la fonction _cliptext est un moyen de mettre une chaîne de caractères dans la mémoire tampon, mais _cliptext est beaucoup plus que cela. Premièrement lorsque l'on utilise la fonction _CLIPTEXT la chaine de caractères est mise dans la mémoire tampon telle qu'elle, mais elle est aussi enregistrée sous plusieurs autres formats que l'on peut utiliser. Le nombre de formats depend de la provenance du contenu de l'infomation qui se trouve dans le _CLIPTEXT.
    Voici un fonction que enumère les formats disponibles de _CLIPTEXT.
    Cliptextformats.prg

Vous remarquerez que lorsque je copie de Notepad, on obtient 4 formats de la chaine de caractères.

Csresult

Fmtid	Fmtname	Fmtvalue	Datasize	Fmtdata
13	Unicode Text	3177688	1002	Memo
16	Locale Identifier handle	2982336	4	Memo
1	Text format	3039680	501	Memo
7	Text format with OEM charset	3178728	501	Memo

Mais lorsque l'on copie le même texte d'un document Word par exemple, on obtient plusieurs autres formats. Comme RTF et HTML.

Csresult

Fmtid	Fmtname	Fmtvalue	Datasize	Fmtdata
49161	DataObject	2982336	4	Memo
49166	Object Descriptor	2999560	148	Memo
49345	Rich Text Format	3080280	32047	Memo
49365	HTML Format	3112336	25819	Memo
1	Text format	3178728	501	Memo
13	Unicode Text	3177264	1002	Memo
14	Enhanced Metafile	675680400	0	memo
3	Metafile Picture format	3059624	16	Memo
49163	Embed Source	3138168	14671	Memo
49156	Native	3152848	14665	Memo
49155	OwnerLink	3043824	43	Memo
49171	Ole Private Data	3178280	312	Memo
16	Locale Identifier handle	2982384	4	Memo
7	Text format with OEM charset	3179248	501	Memo

On pourrait par exemple copier un document Word que l'on veut sauvegarder en

En document RTF. Avec le curseur ci-haut on n'a qu'à prendre l'information du champ Fmtdata et utiliser STRTOFILE() vers un document RTF.
Voici un autre exemple qui démontre la technique.
Extendedcliptext.prg

11. Comment convertir une image en un icone.
    Avec les API de GDI il a facile de produire un icone à partir d'une image.
    Imagetoicon.prg
    Resultat testicon.ico

12. La lecture de la mémoire partagée.
    Voici un exemple comment passer de l'information d'une application à une autre en etablissant une mémoire partagée acessible des deux applications.
    (rouler le meme prg dans deux instances de foxpro)
    Sharedmemory.prg

13. Comment encrypter et decrypter un fichier texte.

Voici comment encrypter un fichier texte et le decryter.
La classe utilise plusieurs fonctions EncryptDecrypt API de cryptographie à mettre en œuvre protégé par mot de passe et le décryptage des fichiers.
Encrypttext.prg

14. Comment dectecter un ajout d'un ficher dans un repertoire.
    Directorychanges.prg

15. Comment créer un windows service avec Foxpro.
    Pour créer un windows service il nous faut 3 choses.
    - Une entree dans la base de registre, qui pointe vers l'executable.
    - Instsrv.exe et srvany.exe (qui font parti de Windows SDK). Le premier installe le service et le deuxième roule le service.
    - Et finallement notre service lui-meme (executable Foxpro).

    a) Créer un fichier .reg avec queleque chose comme ceci.
    Windows Registry Editor Version 5.00

    [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SymcodService\Parameters]

    "AppDirectory"="C:\\symcodservice"

    "Application"="C:\\symcodservice\\SymcodService.exe"

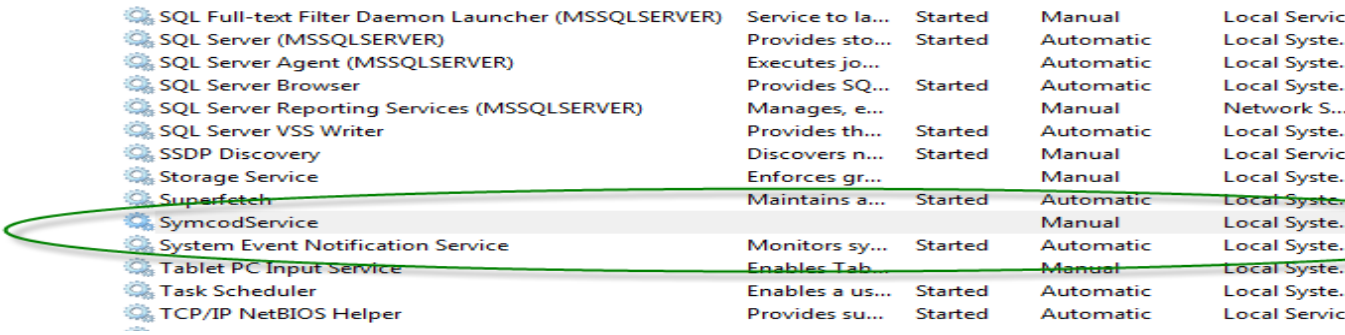
    "AppParameters"="myparm1 myparm2"

    Les deux parametres peuvent par exemple le chemin du fichier ini. Et le deuxième peut-etre le programme que l'on veut rouler.

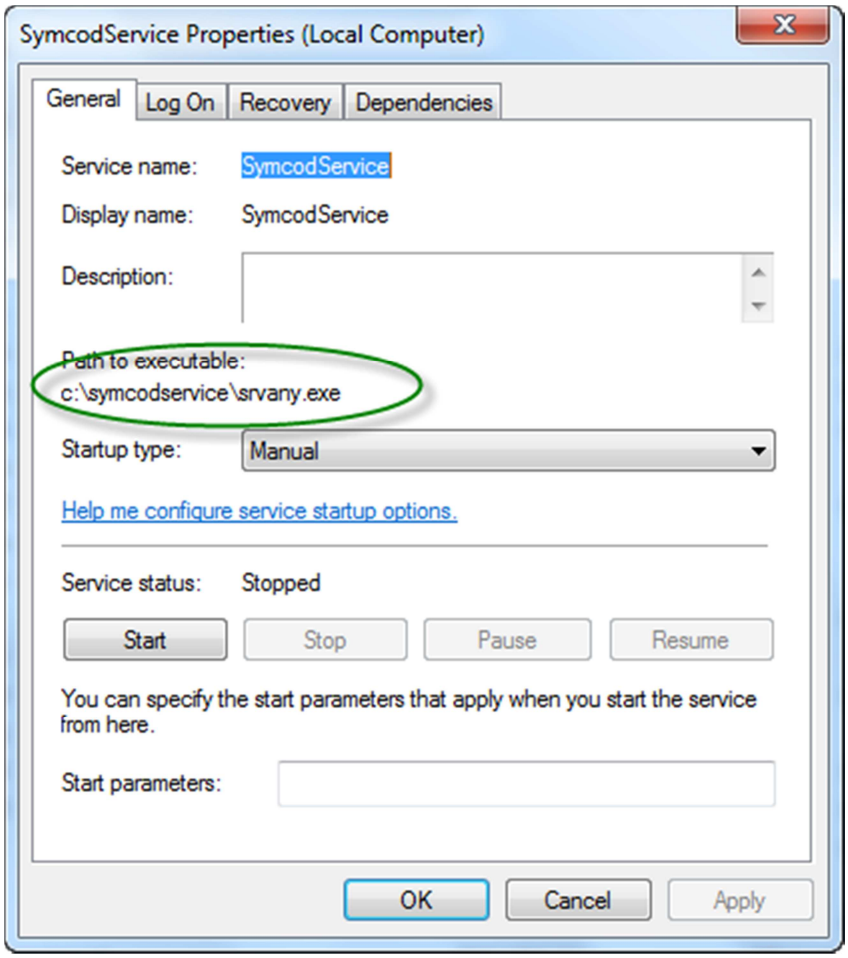
    b) Pour installer le service nous avons qu'utiliser la ligne de commande suivante :
    C:\Program Files\Windows Resource Kits\Tools>instsrv SymcodService "d:\Program Files\Windows Resource Kits\Tools\srvany.exe"

    Le programme srvany.exe doit rester dans le repertoire de l'application, puisque c'est lui qui va indiquer au service quel executable rouler, en lisant la base de registre.

    On peut confirmer que le service est bien installe en consultant le panneau de service.



Vous remarquerez que le service pointe vers srvany.exe et non vers le service lui-meme, puisque la base de registre contient l'information.



c) Notre executable est bien simple, Il prend le parametre de la base de registre est excecute le programme indique.

```
*  Program...........: service.prg

*  Author............: Mike Gagnon

*  Project...........: Carver Human Resources
```

```
*  Created.....: June 17, 2012

*  Code page.....: 1252 (WINDOWS)

*  Copyright.....: (c) Carver Technologies
Inc. 2012

*  Description.....: Start up program for a
sevice (symcodservice or others)

*                   : This is to replace the
symcod watcher application

*                   : But can be used to start
other services.

*  Classes.....: None

*                   :

*                   :

*                   :

Lparameters tcInifile,tcProgram

#Define EVENT_SUCCESS  0

local oService, loShell

oService = null


on shutdown myShutDown()


oService=Newobject("vfpsrv",'','',tcInifile,tcProg
ram) &&& create the service with thee parameters

*** from the windows registry


oService.logstr("Starting Service: got some
params: "+Transform(tcProgram)+'
'+Transform(tcInifile))


*** A windows event log to record the success of
the service start.
```

```
loShell = Createobject("Wscript.Shell")

loShell.LogEvent(EVENT_SUCCESS,"Symcod service est
parti avec succès.  "+Ttoc(Datetime()))


Read Events       && message loop

*#################################################
########

Define Class vfpsrv As custom

*#################################################
########

*================================================

Procedure Init

            Lparameters tcInifile, tcProgram

            This.logstr(Curdir())

            This.logstr(tcInifile+' '+tcProgram)

            *** Run the program that was sent as a
parameter

            Do &tcProgram  With tcInifile

endproc

*================================================

Procedure logstr(Str As String)

   *** write to a log to confirm the service has
started.

      Str=Transform(Datetime())+"
"+Str+Chr(13)+Chr(10)

      Strtofile(Str,"c:\vfpsrv.log",.T.)

Endproc


*#################################################
########

Enddefine
```

```
*================================================


Procedure MyShutdown
      Clear Events
      Quit
Endproc
```

Comme cela on peut utiliser ce service pour partir n'importe quelle application.

Une chose à noter le service ne permet aucune interface (forme, wait windows, messagebox etc…)

Voici un example qui est fonctionnel….