Scanning MS Office documents using the MS Anti-virus API

The Microsoft Antivirus API enables software developers to develop applications that scan Microsoft Office documents before opening them. The Antivirus API also supports scanning Microsoft IE code downloads, such as ActiveX controls.

The primary purpose of this API is to give a software developers the ability to design and implement antivirus software that can be used by all applications. The antivirus component is a standard ActiveX component you register as an in-process server that supports the MSOfficeAntiVirus component category: `(CATID_MSOfficeAntiVirus: TGUID = '{56FFCC30-D398-11d0-B2AE-00A0C908FA49}')`.

IE and MS Office implement the antivirus component as follows:

1. Obtain the list of all the installed antivirus components registered as supporting the *MSOfficeAntiVirus* component category.
2. Launch the installed components.
3. Query for the *IOfficeAntiVirus* interface.
4. Call the *IOfficeAntiVirus.Scan* method to obtain all the installed components.
5. Continue to open the file after the virus scan, regardless of the *HRESULT* value. The antivirus software warns a user if a file has a known virus but opens the file after the warning. It is up to the user to take action concerning the warning.

```
unit msoav;

interface

uses Windows, SysUtils, ActiveX, ComObj, Classes;

const
IID_IOfficeAntiVirus : TGUID =    '{56FFCC30-D398-11d0-B2AE-00A0C908FA49}';
CATID_MSOfficeAntiVirus : TGUID = '{56FFCC30-D398-11d0-B2AE-00A0C908FA49}';

type

  TInfoStruct = record
    fIsFile : boolean;
    fIsReadOnly : boolean;
    fIsInstalled : boolean;
    fIsHTTPDownload : boolean;
  end;

  {Contains information about the file to be scanned:
  * cbSize      - Integer value that specifies the size of an MSOAVINFO
  *               structure.
  * hWnd        - Handle to the parent window of the Microsoft® Office 2000
  *               application.
  * pwzFullPath - Address of a wide character string that contains the full
  *               path of the file about to be opened.
  * lpStg       - Address of the OLE storage location of the file about to
  *               be opened.
  * pwzHostName - Address of a wide character string that contains the host
  *               application name for the antivirus scanner user interface.
  * pwzOrigURL  - Address of a wide character string that contains the URL
  *               of the origin of a downloaded file.}
  TMsoavinfo = record
    cbSize: integer;
    info: ULONG;
    wnd: HWND;
    FullPath: Pointer;
    pwzHostName: PWChar;
    pwzOrigURL: PWChar;
  end;

  {This is the interface an antivirus scanner uses to interact with a host
```

```
  application}
  IOfficeAntiVirus = interface(IUnknown)
  ['{56FFCC30-D398-11d0-B2AE-00A0C908FA49}']
    function Scan(pmsoavinfo : PChar) : HResult; stdcall;
  end;

function TestBit(const Value: Cardinal; const Bit: byte): Boolean;
procedure GetRegisteredAntiviruses(ProgIDs: TStrings);

implementation

function TestBit(const Value: Cardinal; const Bit: byte): Boolean;
begin
  Result := (Value and (1 shl (Bit mod 32))) <> 0;
end;

procedure GetRegisteredAntiviruses(ProgIDs: TStrings);
var
  CatInformation: ICatInformation;
  Enum: IEnumGUID;
  CLSID: TGUID;
  nFetched: Cardinal;
  CatId: TGUID;
begin
  CatInformation := CreateComObject(CLSID_StdComponentCategoryMgr)
    as ICatInformation;
  CatId := CATID_MSOfficeAntiVirus;
  OleCheck(CatInformation.EnumClassesOfCategories(1, @CatId, 0, nil, Enum));
  ProgIDs.BeginUpdate;
  try
    ProgIDs.Clear;
    while (Enum.Next(1, CLSID, nFetched) = S_OK) do begin
      ProgIDs.Add(GuidToString(clsid));
    end;
  finally
    ProgIDs.EndUpdate;
  end;
end;

end.
```

Now I will show a small example how to use the *IOfficeAntiVirus* interface to implement own antivirus program for Microsoft Office.

```
library msoavtest;

uses
  ComServ,
  msoav,
  umsoavtest;

exports
  DllGetClassObject,
  DllCanUnloadNow,
  DllRegisterServer,
  DllUnregisterServer;

begin
end.
```

```
unit umsoavtest;

interface

uses
  Windows, ActiveX, ComObj, ShlObj, Dialogs, msoav;

type
  TMSOTest = class(TComObject, IOfficeAntiVirus)
  protected
   function Scan(pmsoavinfo : PChar) : HResult; stdcall;
  end;

const
```

```
  Class_MsoTest: TGUID = '{F56BE781-C8BE-11D7-8601-00E0184D1E9D}';

implementation

uses ComServ, SysUtils, ShellApi, Registry;

procedure UpdateCat(Register: Boolean;  const ClassID:  string);
const
  SCatImplBaseKey = 'CLSID\%s\Implemented Categories';
  SCatImplKey = SCatImplBaseKey + '\%s';
var
  CatReg: ICatRegister;
  Rslt: HResult;
  CatInfo: TCATEGORYINFO;
  Description: string;
begin
  Rslt := CoCreateInstance(CLSID_StdComponentCategoryMgr, nil,
    CLSCTX_INPROC_SERVER, ICatRegister, CatReg);
  if Succeeded(Rslt) then
  begin
    if Register then
    begin
      CatInfo.catid := CATID_MSOfficeAntiVirus;
      CatInfo.lcid := $0409;
      StringToWideChar('', CatInfo.szDescription,
        Length('') + 1);
      OleCheck(CatReg.RegisterCategories(1, @CatInfo));
      OleCheck(
        CatReg.RegisterClassImplCategories(
          StringToGUID(ClassID), 1, @CATID_MSOfficeAntiVirus
        )
      );
    end
    else
    begin
      OleCheck(
        CatReg.UnRegisterClassImplCategories(
          StringToGUID(ClassID), 1, @CATID_MSOfficeAntiVirus
        )
      );
      DeleteRegKey(Format(SCatImplBaseKey, [ClassID]));
    end;
  end
  else
  begin
    if Register then
    begin
      CreateRegKey(
        'Component Categories\' + GUIDToString(CATID_MSOfficeAntiVirus),
        '409',
        ''
      );
      CreateRegKey(
        Format(
          SCatImplKey, [ClassID, GUIDToString(CATID_MSOfficeAntiVirus)]
        ),
        '',
        ''
      );
    end
    else
    begin
      DeleteRegKey(
        Format(
          SCatImplKey, [ClassID, GUIDToString(CATID_MSOfficeAntiVirus)]
        )
      );
      DeleteRegKey(Format(SCatImplBaseKey, [ClassID]));
    end;
  end;
  if Register then
  begin
    Description := GetRegStringValue('CLSID\' + ClassID, '');
    CreateRegKey('AppID\' + ClassID, '', Description);
```

```
    CreateRegKey('CLSID\' + ClassID, 'AppID', ClassID);
  end
  else
    DeleteRegKey('AppID\' + ClassID);
end;

{ TMSOTest }

function TMSOTest.Scan(pmsoavinfo: PChar): HResult;
var
Info   : TMsoavinfo;
Struct : TInfoStruct;
p : pointer;
begin
  p := pointer(pmsoavinfo);
  if not Assigned(p) then
   begin
     //no information available
     Result := S_OK;
     Exit;
   end;

  Move(P^, Info, SizeOf(Tmsoavinfo));
  if Info.cbSize <> SizeOf(Tmsoavinfo) then
   begin
     //wrong size of the structure
     Result := S_OK;
     Exit;
   end;
  Struct.fIsFile := TestBit(Info.Info, 0);
  Struct.fIsReadOnly := TestBit(Info.Info, 1);
  Struct.fIsInstalled := TestBit(Info.Info, 2);
  Struct.fIsHTTPDownload :=  TestBit(Info.Info, 3);
  if struct.fIsFile then
   begin
     MessageDlg(PWChar(Info.FullPath), mtWarning, [mbOK], 0);
   end;
  Result := S_OK;
end;

type
  TMSOAvFactory = class(TComObjectFactory)
  public
    procedure UpdateRegistry(Register: Boolean); override;
  end;

procedure TMSOAVFactory.UpdateRegistry(Register: Boolean);
var
  ClassID: string;
begin
  ClassID := GUIDToString(Class_MsoTest);
  if Register then
  begin
    inherited UpdateRegistry(Register);
    UpdateCat(true, ClassID);
  end
  else
  begin
    UpdateCat(false, ClassID);
    inherited UpdateRegistry(Register);
  end;
end;

initialization
  TComObjectFactory.Create(
    ComServer, TMsoTest, Class_MsoTest, 'MsoTest', '',
    ciMultiInstance, tmApartment
  );
end.
```

Author:	Shlomo Abuisak
Contributor:	Shlomo Abuisak
Added:	2009-11-05
Last updated:	2009-11-05