

## Re: Displaying WMV from a stream with the WM ASF Reader Filter in a filter graph

*Source:* <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2005-01/0424.html>

---

*From:* Bob Dellaca (*bobdlxyzy\_at\_xtra.co.nz*)

*Date:* 01/22/05

Date: Sat, 22 Jan 2005 16:14:23 +1300

For the record, here is the source code of a sample unit to play a WMA (or WMV or ASF) file from a stream (here a TStream (TFileStream)).

It uses DS Pack 2.3.4. As outlined earlier in this thread, it involves defining your own "custom file type" as a "protocol" and getting the ASF Reader filter to accept an IStream from your application when the filter is asked to render a file of that custom protocol

```
unit Unit2;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,  
  Dialogs, Menus,  
  ActiveX, DirectShow9;
```

```
type
```

```
  TmyProtStream = class(TInterfacedObject, IStream)  
  private  
    aStream: TStream;  
    aPosition, aLength: Integer;  
  public  
    constructor Create(Stream: Tstream);  
    function Read(pv: Pointer; cb: Longint; pcbRead: PLongint):  
      HRESULT;  
    stdcall;  
    function Write(pv: Pointer; cb: Longint; pcbWritten: PLongint):  
      HRESULT;  
    stdcall;  
    function Seek(dlibMove: Largeint; dwOrigin: Longint;  
      out libNewPosition: Largeint): HRESULT; stdcall;  
    function SetSize(libNewSize: Largeint): HRESULT; stdcall;  
    function CopyTo(stm: IStream; cb: Largeint; out cbRead: Largeint;  
      out cbWritten: Largeint): HRESULT; stdcall;
```

```
function Commit(grfCommitFlags: Longint): HRESULT; stdcall;
function Revert: HRESULT; stdcall;
function LockRegion(libOffset: Largeint; cb: Largeint;
    dwLockType: Longint): HRESULT; stdcall;
function UnlockRegion(libOffset: Largeint; cb: Largeint;
    dwLockType: Longint): HRESULT; stdcall;
function Stat(out statstg: TStatStg; grfStatFlag: Longint):
HRESULT;
                                stdcall;
function Clone(out stm: IStream): HRESULT; stdcall;
end;

TForm1 = class(TForm)
  MainMenu1: TMainMenu;
  File1: TMenuItem;
  Openwmafile1: TMenuItem;
  OpenFileDialog1: TOpenDialog;
  Exit1: TMenuItem;
  procedure Openwmafile1Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure Exit1Click(Sender: TObject);
private
  myProtStream: TmyProtStream;
  pFilgraphManager: IMediaControl;
  pGraph: IGraphBuilder;
  pStream: TFileStream;
  procedure CloseIt;
public
end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

constructor TmyProtStream.Create(Stream: TStream);
begin
  inherited Create;
  aStream := Stream;
  aLength := aStream.Size;
end;

function TmyProtStream.Read(pv: Pointer; cb: Longint; pcbRead:
PLongint): HRESULT;
var
  ReadLength: Integer;
begin
  if (aPosition + cb > aLength) then
    ReadLength := aLength - aPosition else
```

```
    ReadLength := cb;
    aStream.Read(pv^, ReadLength);
    pcbRead^ := ReadLength;
    aPosition := aPosition + ReadLength;
    Result := S_OK;
end;

function TmyProtStream.Seek(dlibMove: Largeint; dwOrigin: Longint;
    out libNewPosition: Largeint): HRESULT;
var
    newPos: Integer;
begin
    case dwOrigin of
        STREAM_SEEK_SET: newPos := dlibMove;
        STREAM_SEEK_CUR: newPos := aPosition + dlibMove;
        STREAM_SEEK_END: newPos := aLength - dlibMove;
    else
        begin
            Result := E_INVALIDARG;
            Exit;
        end;
    end;
    if (newPos < 0) or (newPos > aLength) then
        Result := E_INVALIDARG else
        begin
            aPosition := newPos;
            aStream.Position := aPosition;
            if @libNewPosition <> nil then
                libNewPosition := newPos;
            Result := S_OK;
        end;
    end;

function TmyProtStream.Stat(out statstg: TStatStg; grfStatFlag:
    Longint): HRESULT;
begin
    if (@statstg = nil) or (grfStatFlag <> STATFLAG_NONAME) then
        begin
            Result := E_INVALIDARG;
            Exit;
        end;
    FillChar(statstg, sizeof(TStatStg), 0);
    statstg.dwType := STGTY_STREAM;
    statstg.cbSize := aLength;
    Result := S_OK;
end;

function TmyProtStream.Write(pv: Pointer; cb: Longint; pcbWritten:
    PLongint): HRESULT;
begin
    Result := E_NOTIMPL;
```

```
end;

function TmyProtStream.SetSize(libNewSize: Largeint): HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.CopyTo(stm: IStream; cb: Largeint; out cbRead:
Largeint;
                            out cbWritten: Largeint): HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.Commit(grfCommitFlags: Longint): HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.Revert: HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.LockRegion(libOffset: Largeint; cb: Largeint;
dwLockType: Longint): HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.UnlockRegion(libOffset: Largeint; cb: Largeint;
dwLockType: Longint): HRESULT;
begin
    Result := E_NOTIMPL;
end;

function TmyProtStream.Clone(out stm: IStream): HRESULT;
begin
    Result := E_NOTIMPL;
end;

procedure TForm1.Openwmfile1Click(Sender: TObject);
const
    myProt = 'myProt://$%x';
var
    aFileName: string;
    FileName: WideString;
begin
    if OpenDialog1.Execute then
        begin
            CloseIt;
```

```
CoCreateInstance(CLSID_FilterGraph, nil, CLSCTX_INPROC_SERVER,
  IID_IFilterGraph2, pGraph);
pFilgraphManager := pGraph as IMediaControl;
pStream := TFileStream.Create(OpenDialog1.FileName, fmOpenRead);
try
  myProtStream := TmyProtStream.Create(pStream);
  aFileName := Format(myProt, [Integer(myProtStream as IStream)]);
  FileName := aFileName;
  pGraph.RenderFile(@FileName[1], nil);
except
  on E:Exception do
  begin
    MessageDlg(E.Message, mtError, [mbOk], 0);
    pGraph := nil;
    CloseIt;
    Exit;
  end;
end;
pFilgraphManager.Run;
end;
end;

procedure TForm1.CloseIt;
begin
  if pGraph <> nil then
  begin
    pFilgraphManager.Stop;
    pGraph := nil;
  end;
  pFilgraphManager := nil;
  myProtStream := nil;
  FreeAndNil(pStream);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  CloseIt;
end;

procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;

end.
```