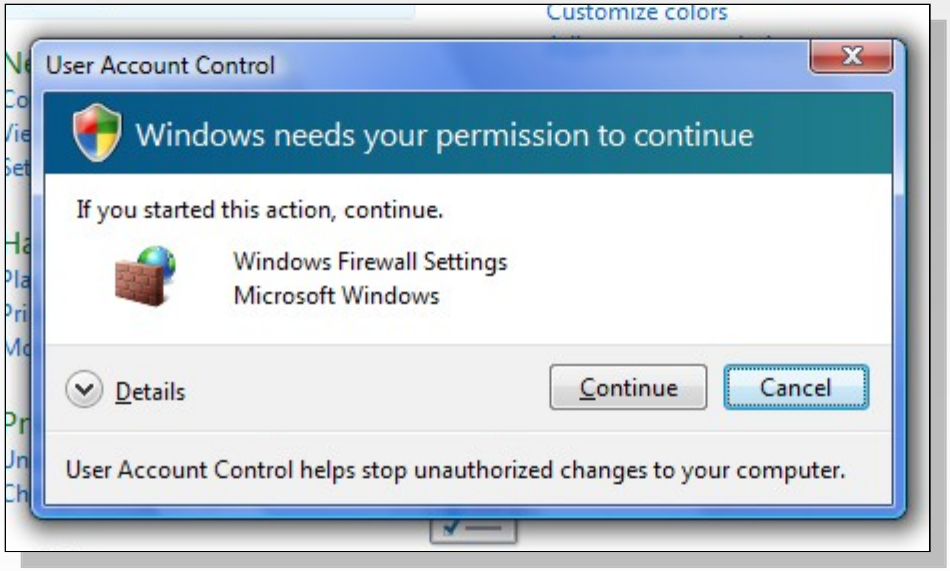
Windows Vista
User Account Control (UAC) and Delphi

Fredrik Haglund
Developer Evangelist

CodeGear™
The Developer Company

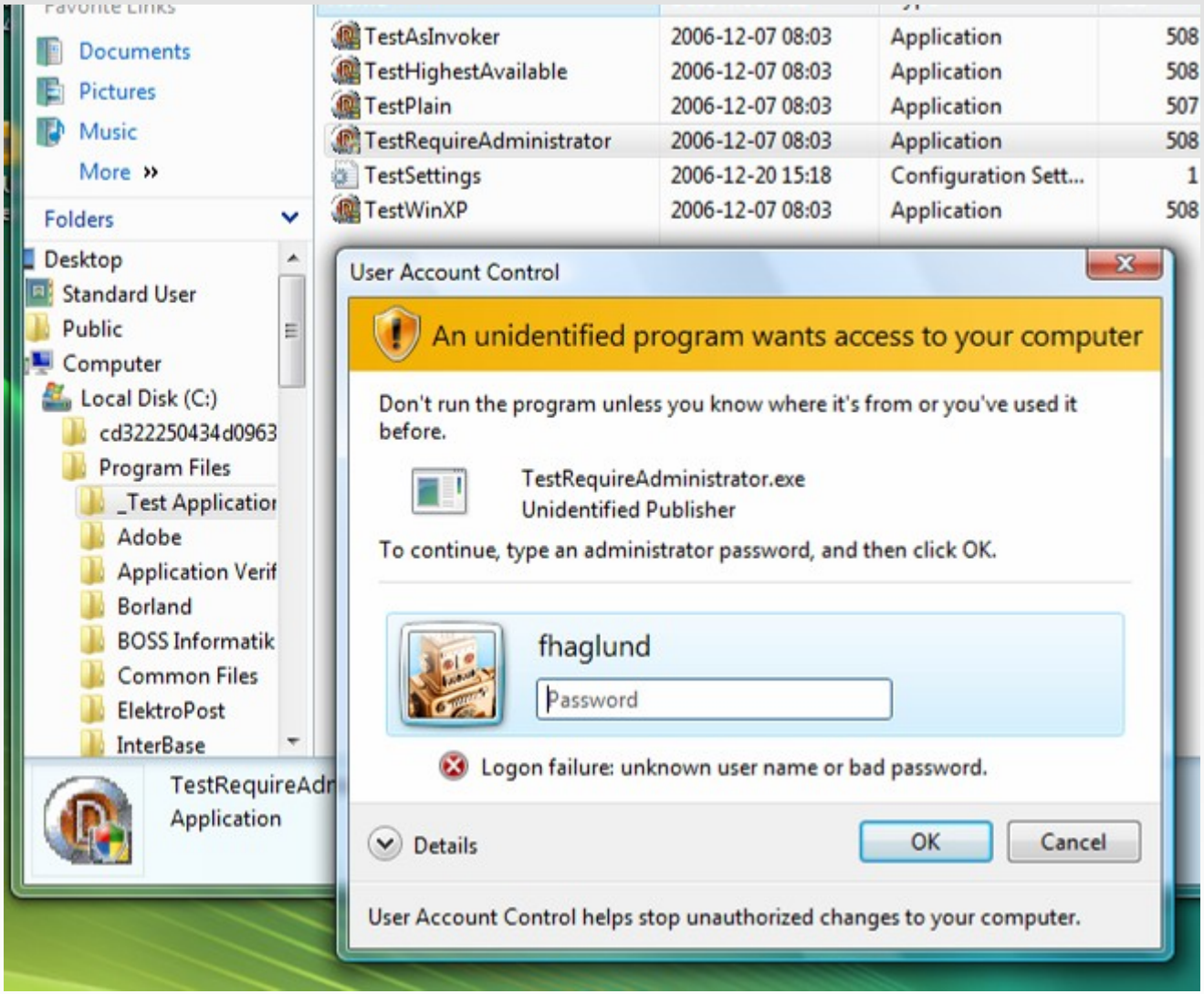
User Account Control (UAC)

- Security token split during logon
  - one **user token** and one **admin token**
- Administrator shell run with **Standard User** token
- You have to explicitly consent every time you create a process with administrator token – this is called “elevation”



CodeGear™
The Developer Company

Standard User – Over the shoulder elevation



CodeGear™
The Developer Company

Windows Vista

- UAC is Enabled by Default
- All Subsequent User Accounts are Created as Standard Users
- Elevation Prompts are Displayed on the Secure Desktop by Default
- Elevation Prompts for Background Applications are Minimized to the Taskbar
- Elevations are blocked in the User's Logon Path
- Built-in Administrator Account is Disabled by Default on New Installations
- New Default Access Control List (ACL) Settings

Standard User

- All processes are started as Standard User as default
- A Standard User can not
  - Change files in **Program Files** folders
  - Change files in **Windows** or **System32** folders
  - Change registry under **HKLM\Software**
  - Change the local machines **date and time**
  - Install or uninstall **Services**
  - …
- Earlier strong Recommendations are now enforced!

CodeGear™
The Developer Company

New Technologies for Windows Vista

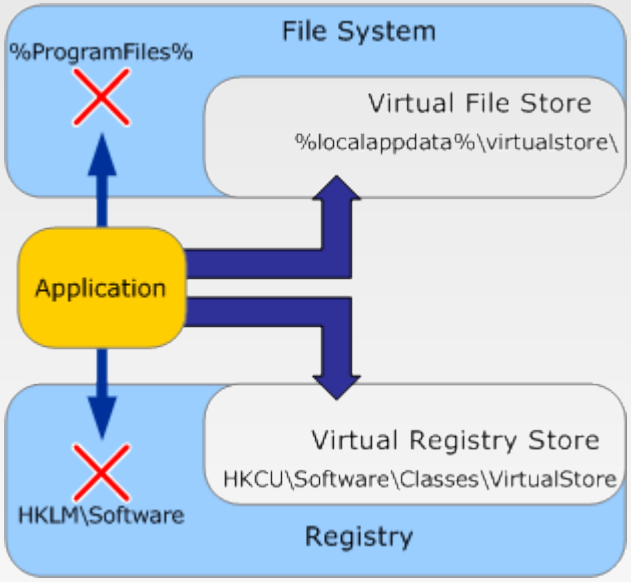
- Installer Detection
- User Interface Privilege Isolation
- Virtualization
- Access Token Split during login
- Secure Desktop

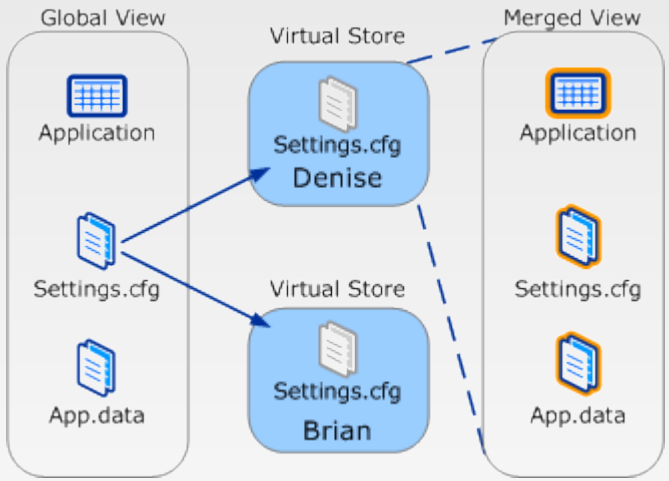
User Interface Privilege Isolation

- General guideline – “lower” can not access “higher”
- A lower privilege process cannot:
  - Perform a window handle validation
  - SendMessage or PostMessage
  - Use thread hooks to attach
  - Use Journal hooks to monitor
  - Perform dynamic link-library (DLL) injection
- Some resources are still shared between processes
  - Desktop window, which actually owns the screen surface
  - Desktop heap read-only shared memory
  - Global atom table
  - Clipboard

Virtualization / Redirection

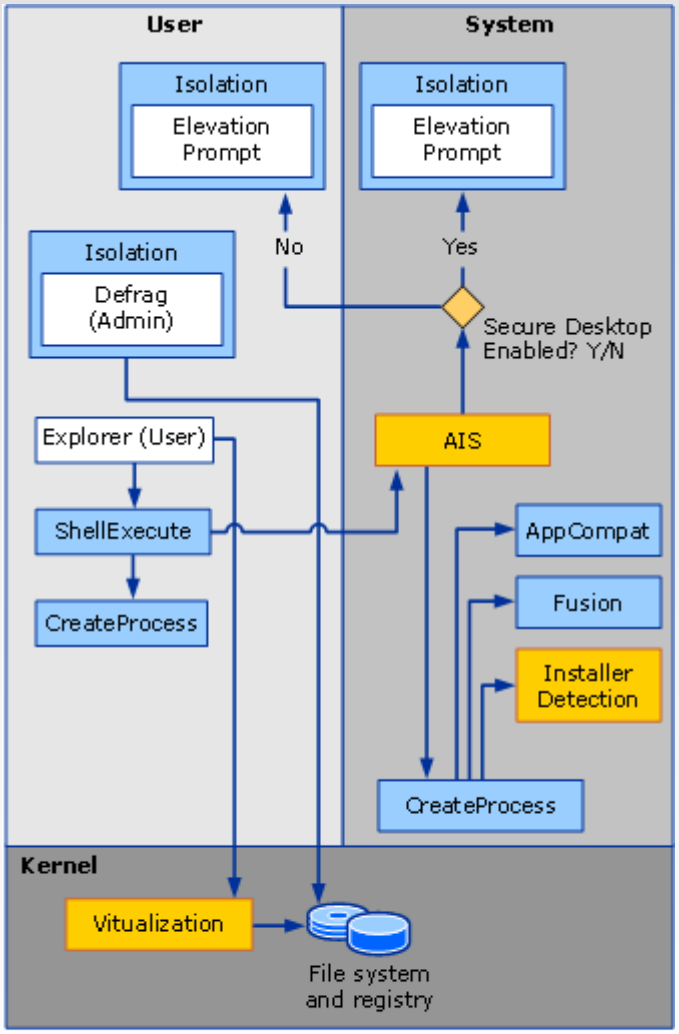
- Virtualization is for compatibility – not a feature
- Disabled for executables with UAC info in manifest!





CodeGear™
The Developer Company

UAC Architecture

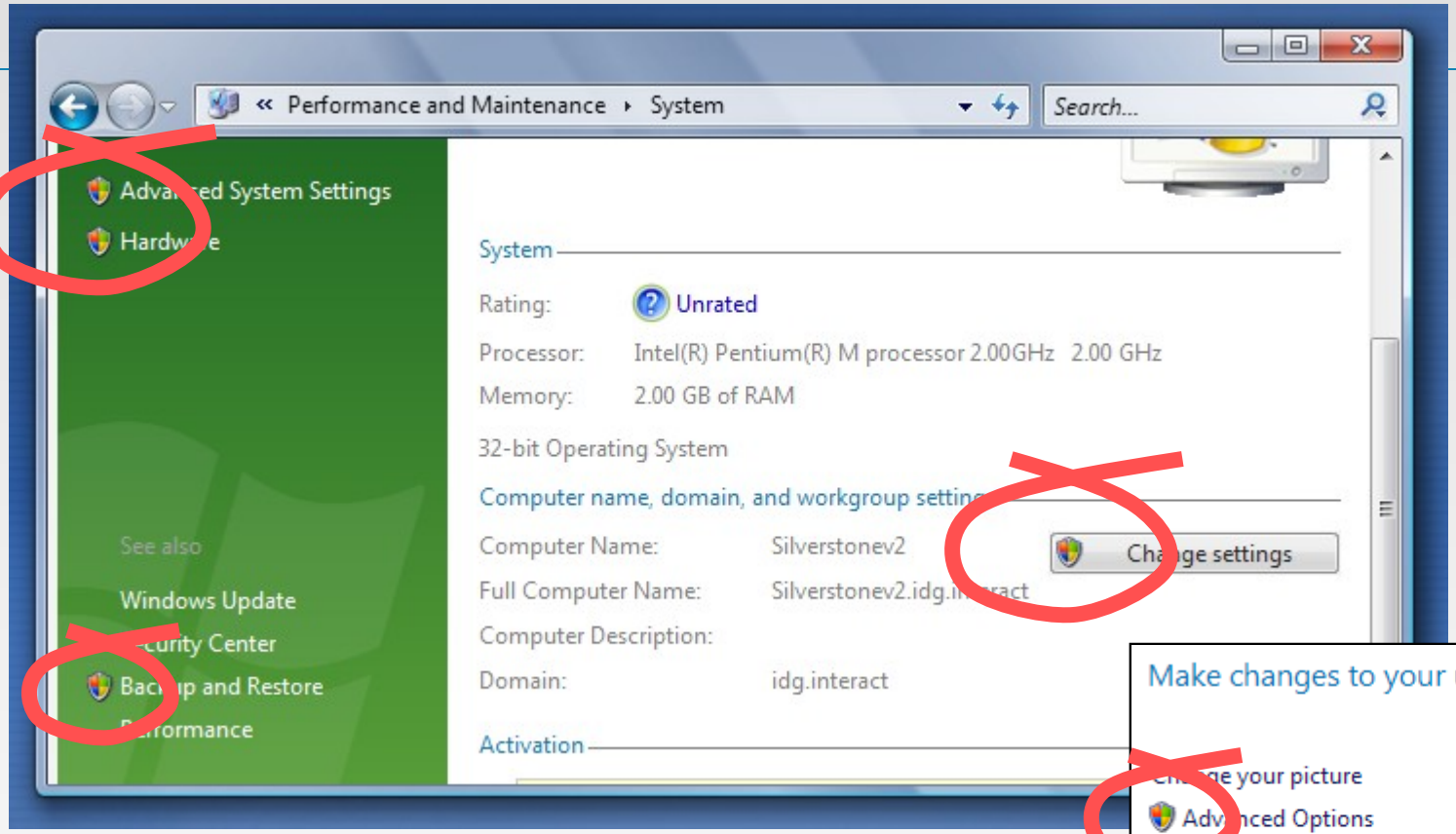


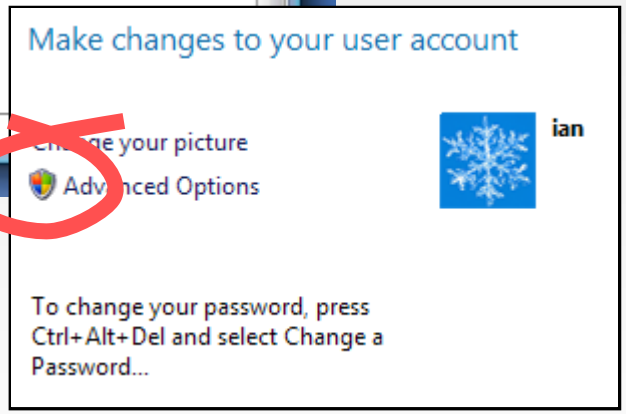
CodeGear™
The Developer Company

The Shield

- Attached to controls which, if clicked, will require elevation as the next step
- Has only one state (I.e. no hover, disabled etc.)
- Does not remember elevated state
  - *Not* an unlock operation

Shield UI Examples





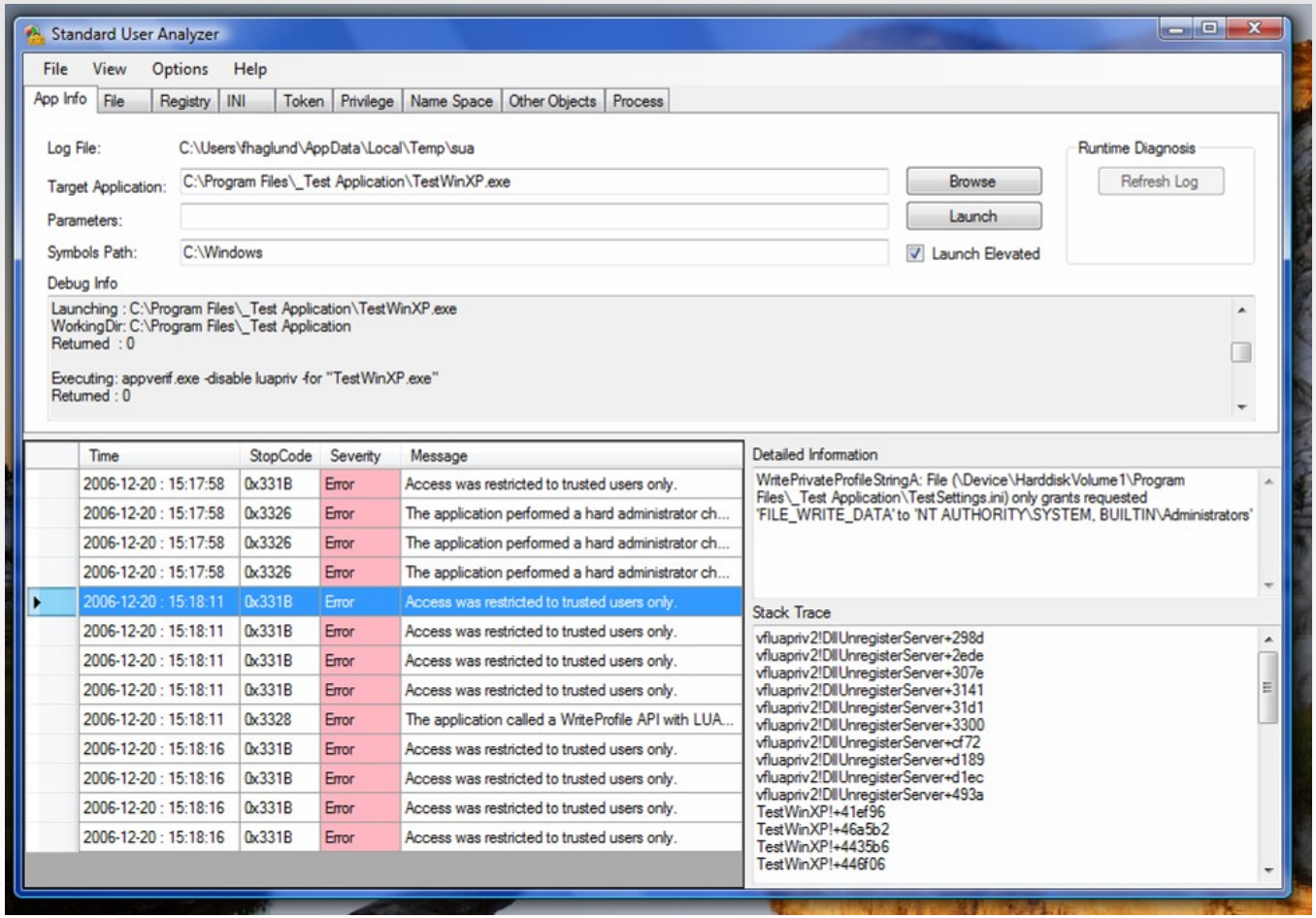
CodeGear™
The Developer Company

Delphi – What you have to do…

- Test your application – identify problems
- Classify your application as Standard User, Admin or Mixed.
- Add application Manifest
- Redesign functionality
  - User apps should write data to correct locations
  - Split out admin stuff into a separate executable
- Redesign user interface
  - Add shield to buttons
- Redesign installer
- Test again
- Optionally sign application (Authenticode)
- Determine whether to pursue the Windows Vista Logo program

Test with Standard User Analyzer Tool

- SUA helps you find what you do that can break application



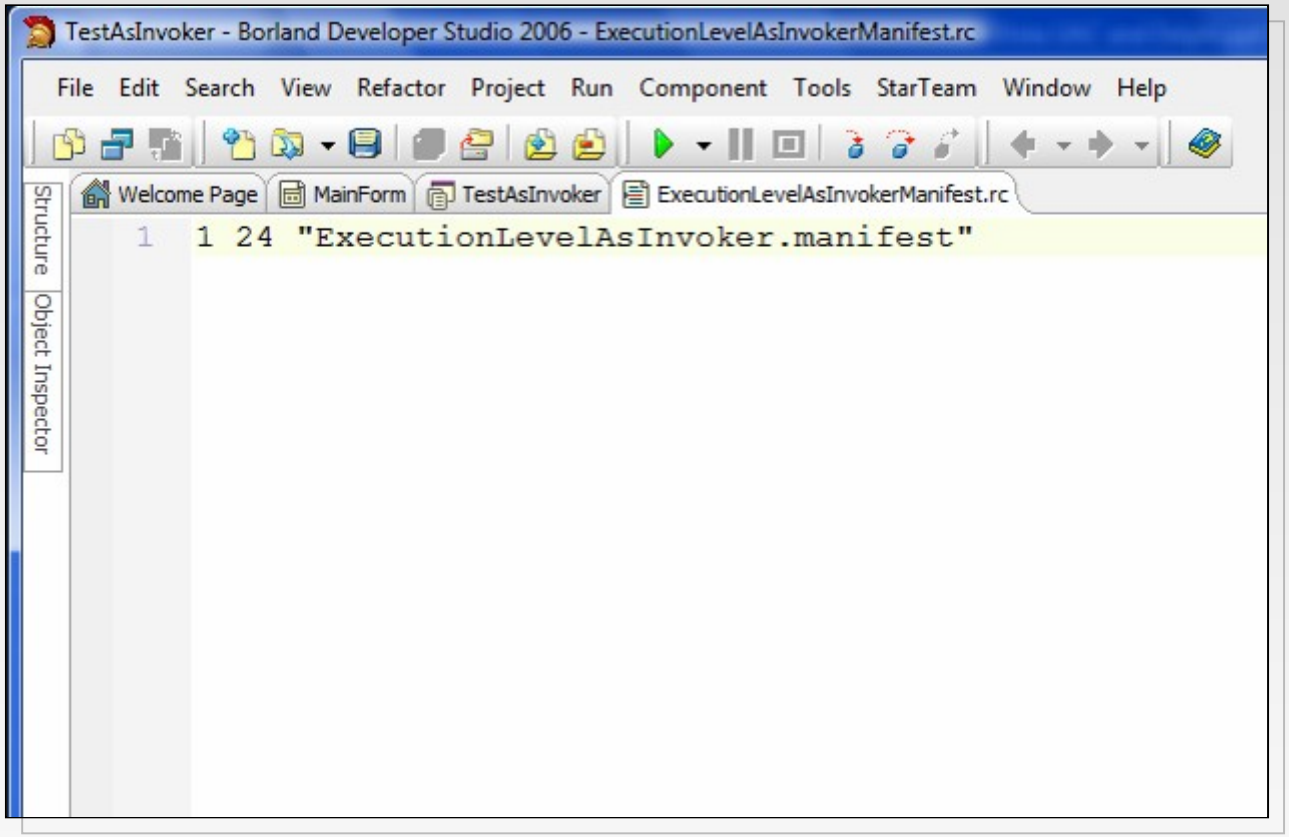
CodeGear™
The Developer Company

Requested Execution Level in Delphi

- NB! Remove all references to XPMan unit from project!!!

```
program TestAsInvoker;

{$R 'ExecutionLevelAsInvokerManifest.res' 'ExecutionLevelAsInvokerManifest.rc'}

uses
  Forms,
  MainForm in '..\Common\MainForm.pas' {Form2};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

RC-file is compiled to RES-file



CodeGear™
The Developer Company

Manifest

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <dependency>
    <dependentAssembly>
      <assemblyIdentity
        type="win32"
        name="Microsoft.Windows.Common-Controls"
        version="6.0.0.0"
        publicKeyToken="6595b64144ccf1df"
        language="*"
        processorArchitecture="x86" />
    </dependentAssembly>
  </dependency>

  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel
          level="asInvoker"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

<requierdExecutionLevel />

- level=”asInvoker”
  - Start process runing with same token as the process creating it.
- level=”highestAvailable”
  - Ask administrators for consent to elevate but start as standard user if user has no administrative privileges
- level=”requireAdministrator”
  - Ask administrators for consent to elevate.
  - Standard user will get login dialog for over the shoulder support
  - Will only start with administrative privileges

Windows XP Warning!

- Incorrect formatting of Manifest can **blue screen** Windows XP
- Read KB921337

Redesign

- Do not open files or registry keys with Write flag
- Save data, log files, etc. in the right location using SHGetFolderPath
  - CSIDL_PERSONAL { My Documents }
  - CSIDL_APPDATA { Application Data, new for NT4 }
  - CSIDL_LOCAL_APPDATA { non roaming, user\Local Settings\Application Data }
  - CSIDL_COMMON_APPDATA { All Users\Application Data }
  - CSIDL_MYPICTURES { My Pictures, new for Win2K }
  - CSIDL_COMMON_DOCUMENTS { All Users\Documents }
  - …

SHGetFolderPath

```
uses
  SHFolder;

function GetFolder(csidl: Integer; ForceFolder: Boolean = False): string;
var
  i: Integer;
begin
  SetLength(Result, MAX_PATH);
  if ForceFolder then
    SHGetFolderPath(0, csidl or CSIDL_FLAG_CREATE, 0, 0, PChar(Result))
  else
    SHGetFolderPath(0, csidl, 0, 0, PChar(Result));
  i:= pos(#0, Result);
  if i > 0 then
    SetLength(Result, Pred(i));
end;

function GetLocalAppDataFolder(ForceFolder: Boolean = False): string;
begin
  Result:= GetFolder(CSIDL_LOCAL_APPDATA, ForceFolder);
end;
```

RunAsAdmin

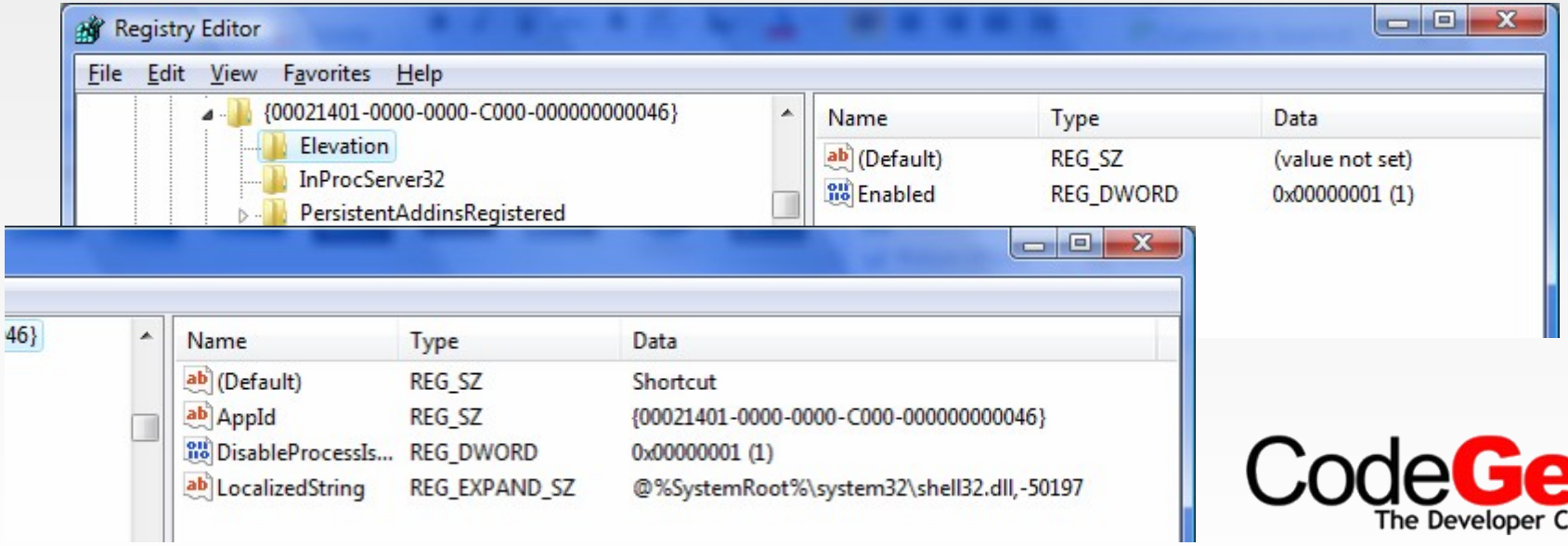
- Launch application running as administrator
- Use Application.Handle to delay elevation if app is minimized.
- No handle always gives direct foreground elevation.

```
// Vista Utilities

procedure RunAsAdmin(hWnd: HWND; aFile: string; aParameters: string);
var
  sei: TShellExecuteInfoA;
begin
  FillChar(sei, SizeOf(sei), 0);
  sei.cbSize := sizeof(sei);
  sei.Wnd := hWnd;
  sei.fMask := SEE_MASK_FLAG_DDEWAIT or SEE_MASK_FLAG_NO_UI;
  sei.lpVerb := 'runas';
  sei.lpFile := PChar(aFile);
  sei.lpParameters := PChar(aParameters);
  sei.nShow := SW_SHOWNORMAL;
  if not ShellExecuteEx(@sei) then
    RaiseLastOSError;
end;
```

CodeGear™
The Developer Company

Using COM class for Admin tasks

- COM Server must be an EXE
- EXE must have requireAdministrator to install COM objects correctly
- Registration of COM Class must
  - add value LocalizedString (and resource string in executable)
  - add key Elevation and value Enabled = 1



CodeGear™
The Developer Company

Elevated COM calls

- Use Moniker to create elevated CoClass from User Process

```
function NewCoGetObject(pazName: PWideChar; pBindOptions: PBindOpts3;
         const iid: TIID; out ppv): HResult; stdcall; external
'ole32.dll'
         name 'CoGetObject';

function CoCreateInstanceAsAdmin(WndHandle: HWND; clsid: TCLSID;
    iid: TIID; out ppv): HRESULT;
var
    Bo      : TBindOpts3;
    Moniker : PWideChar;
begin
    Moniker := PWideChar(WideString('Elevation:Administrator!new:' +
                                    GuidToString(clsid)));
    FillChar(Bo, SizeOf(Bo), #0);
    Bo.hwnd            := WndHandle;
    Bo.cbStruct        := SizeOf(Bo);
    Bo.dwClassContext  := CLSCTX_LOCAL_SERVER;
    Result             := NewCoGetObject(Moniker, @Bo, iid, ppv);
end;
```

CodeGear™
The Developer Company

The Shield - SetElevationRequiredState

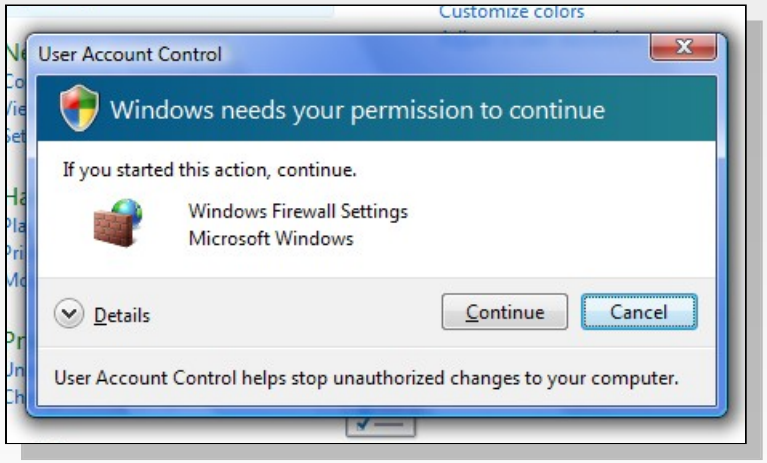
- Call function with Button as parameter to add Shield symbol

```
const
  BCM_FIRST = $1600; // Button control messages
  BCM_SETSHIELD = BCM_FIRST + $000C;

procedure SetElevationRequiredState(aControl: TWinControl; Requiered: Boolean);
var
  lRequired: Integer;
begin
  lRequired := Integer(Requiered);
  SendMessage(aControl.Handle, BCM_SETSHIELD, 0, lRequired);
end;
```

CodeGear™
The Developer Company

Sign with Authenticode

- Get less serious looking consent dialog
- Register at winqual.microsoft.com
- Buy certificate (Verisign, etc.)
- Sign executables (MakeCert, Signtool.exe)
- Register applications at winqual to get access to crash logs



CodeGear™
The Developer Company

Resources

- Document
  - Windows Vista Application Development Requirements for User Account Control Compatibility
- Tool
  - Microsoft Standard User Analyzer
- Windows Vista Logo Program
  - http://microsoft.mrmpslc.com/InnovateOnWindowsVista/

Thank you! ☺

CodeGear™
The Developer Company