

Clarion Magazine

Clarion News

- » [Clarion Data Mapper 1.70](#)
- » [SealSoft 20% Discount](#)
- » [CoolButtons 1.01 Beta](#)
- » [SimDatesClass Templates Count Working Days](#)
- » [CoolFrames 1.12 Beta](#)
- » [Evolution Report Export Tutorial](#)
- » [Ingasoftplus 20% Discount](#)
- » [SetupBuilder 6.5 Patch 1847 for Clarion 3rd Party Developers](#)
- » [PD Browse Button Lookup Updated](#)
- » [Clarion Desktop 3.72](#)
- » [J-Fax 1.50](#)
- » [Free Blowfish Code](#)
- » [J-Skype 1.30](#)
- » [PrintWindow 1.10](#)
- » [Evolution Report Export 1.12](#)
- » [BST 4.23](#)
- » [BoTran 2.4](#)
- » [CoolFrames 1.11](#)
- » [CHM4Clarion Released](#)
- » [Class Anatomy 1.6.0](#)
- » [Clarion FreeImage 3.9.3.2](#)
- » [Clarion Desktop 3.60 Beta](#)
- » [FullRecord 1.82](#)
- » [CapeSoft WorldTour 2007 - Around the World in 18 Days](#)
- » [Simsoft Dates Class and Templates Demo](#)
- » [FileTuner 0.20.](#)
- » [amazingGUI Orange Pack](#)
- » [Free PDF-XChange Viewer 1.016](#)
- » [April Whitemarsh Announcement](#)
- » [CHM4Clarion](#)
- » [UK CUG News](#)
- » [Sticky Notes Updated](#)
- » [amazingGUI 1.0.6](#)
- » [J-Spell 1.25](#)
- » [J-Fax 1.45](#)
- » [EasyOpenOffice 1.04](#)
- » [Autodetecting Clarion Under Vista/Longhorn](#)
- » [Australian QuickBooks/MYOB Accounting Interface Wanted](#)

[\[More news\]](#)

Save up to **50% off ebooks.**
Subscription has its rewards.



Latest Subscriber Content

Creating SQL From XML With XSLT: Calling The Code From Clarion

Having demonstrated how to create SQL INSERT statements from an XML file, Bernie Groperrin shows how to call a C# program to do this task, using a Clarion application

Posted Monday, April 30, 2007

Coping With Vista - There's A Manifest In Your Destiny, Part 2

Vista's new security requirements present a number of problems for Clarion developers. As Jane Fleming explains, the only way to maintain some order and sanity, short of disabling UAC on all your customers' machines, is to sign your applications and include a Vista manifest.

Posted Thursday, April 26, 2007

Sending Messages With NOTIFY And NOTIFICATION

In the first three installments in this series Larry Sand covered the basics of interprocess communication, sending messages, receiving messages, and establishing communication between processes. He also discussed sending user messages between processes. Now Larry expands this capability with NOTIFY and NOTIFICATION.

Posted Wednesday, April 25, 2007

Search Engine Bug Fixed

I've fixed a bug with the search feature which caused some search results to be incomplete. If you've tried a search recently and weren't happy with the results, please search again. Please also note the Search Tips section at the bottom of the [search page](#).

Posted Monday, April 23, 2007

Coping With Vista - There's A Manifest In Your Destiny, Part 1

Jane Fleming explains security tokens, the mechanisms behind security elevation, and the need for application manifests. Part 1 of 2.

Posted Friday, April 20, 2007

Creating SQL From XML With XSLT

You've heard all about XML, and you're probably using it. But do you know about XSLT? Often thought of as a style sheet for XML files, XSLT is far more: it's really a template language for converting XML files into other formats. Bernie Groperrin shows how to create SQL statements directly from XML files using the power of XSLT. Part 1 of 2.

Posted Thursday, April 19, 2007

Coping With Windows Vista - How to live with User Account Control Security

Vista's User Account Control Security has been widely derided as a useless annoyance. But UAC really can be a good thing once you know how to handle it. Jane Fleming explains how to do away with annoying popup messages during routine GUI and command-line tasks, and supplies a "back to the future" set of DOS-style tricks.

Posted Friday, April 13, 2007

Sending User Messages Between Processes

Larry Sand continues his series on interprocess communication by passing user-defined messages between processes.

Posted Thursday, April 12, 2007

Source Code Library 2007.03.31 Available

The Clarion Magazine Source Code Library has been updated to include the March source. Source code subscribers can download the January-March 2007 update from the [My ClarionMag](#) page.

Posted Wednesday, April 04, 2007

[\[Last 10 articles\]](#) [\[Last 25 articles\]](#) [\[All content\]](#)

Source Code

[\[More Clarion 101\]](#)

Latest Free Content

- » [Search Engine Bug Fixed](#)
- » [Source Code Library 2007.03.31 Available](#)

[\[More free articles\]](#)

Clarion Sites

Clarion Blogs

- » [Pimp My Clarion](#)

The ClarionMag Source Code Library

Clarion Magazine is more than just a great place to learn about Clarion development techniques, it's also home to a massive collection of Clarion source code. Clarion subscribers already know this, but now we've made it easier for subscribers and non-subscribers alike to find the code they need.

The Clarion Magazine Source Library is a single point download of all article source code, complete with an article cross-reference.

[More info](#) • [Subscribe now](#)

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our [e-books](#), you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

Printed Books

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:

- » [Clarion 6 Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8](#)
- » [Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X](#)
- » [Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5](#)
- » [Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3](#)
- » [Clarion Databases & SQL - ISBN: 0-9689553-3-9](#)



We also publish Russ Eggen's widely-acclaimed [Programming Objects in Clarion](#), an introduction to OOP and ABC.

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your [subscription](#) not only gets you premium content in the form of new articles, it also includes all the back issues. Our [search engine](#) lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than [pay for itself](#) - you have my personal guarantee.

Dave Harms

ISSN

Clarion Magazine's ISSN

Clarion Magazine's [International Standard Serial Number \(ISSN\)](#) is 1718-9942.

About ISSN

The ISSN is the standardized international code which allows the identification of any serial publication, including electronic serials, independently of its country of publication, of its language or alphabet, of its frequency, medium, etc.

Clarion Magazine

Clarion News

[Search the news archive](#)

Clarion 6.3 Build 9056 Pre-Release

The latest release of Clarion 6.3 has been released to 3rd party vendors to allow them time to prepare their products for the general release. There are many bug fixes and new support for generating a Vista Manifest with UAC attributes, along with the option to automatically add the generated manifest into the project so that it's linked into the executable or DLL. As well, the linker will now report all missing/inaccessible files in the first attempt at compilation. If you're missing icons or other resources you'll get the complete list of all missing files without having to repeatedly attempt to compile the application.

Posted Wednesday, May 02, 2007

Evolution Browse Export Tutorial

There is available a new tutorial video available for Evolution Browse Export. Press F11 for Full Screen mode.

Posted Wednesday, May 02, 2007

Clarion Data Mapper 1.70

Gary James has released Clarion Data Mapper 1.70. The dictionary diagram preview control has been completely rewritten. You can now zoom, set a border around the diagram, set "Shrink to fit", and a couple other new things. There is still a free "light" version of this tool, and Clarion Desktop subscribers get the Pro version at no extra charge.

Posted Friday, April 27, 2007

SealSoft 20% Discount

SealSoft is offering a 20.00% relative discount on all 22 main products.

Posted Friday, April 27, 2007

CoolButtons 1.01 Beta

CoolButtons 1.01 beta is now available. Changes include: Fixed GDI handle and memory leak; Added support for button icon replacement; Added support for selectively colouring buttons; Added support for user graphics as button images; Updated documentation; Updated FAQs; Reduced memory usage.

Posted Friday, April 27, 2007

SimDatesClass Templates Count Working Days

The SimDatesClass templates now have a function to count working days. The function lets you specify how many and which days to count during the period. For example you could count how many Mondays in a period, how many Mondays, Wednesdays and Fridays in a period or any other combination of days (Mon to Fri, Mon to Sat etc). Demo available.

Posted Friday, April 27, 2007

CoolFrames 1.12 Beta

CoolFrames 1.12 Beta is now available. Changes include: Fixed bug where restoring and MDI Frame from iconized left the client area unpainted; Fixed bug where MDI child windows sometimes opened with a little of the old titlebar showing below the CoolFrames titlebar.

Posted Friday, April 27, 2007

Evolution Report Export Tutorial

A new Evolution Report Export tutorial video is now available. Press F11 for Full Screen mode. Evolution Report Export is now on sale for US\$89 at ClarionShop.

Posted Friday, April 27, 2007

Ingasoftplus 20% Discount

Ingasoftplus is offering 20% discounts on all products from April 25, 2007 till May 31, 2007.

Posted Friday, April 27, 2007

SetupBuilder 6.5 Patch 1847 for Clarion 3rd Party Developers

SetupBuilder 6.5 Patch 1847 for Clarion 3rd Party developers is now available. There was a showstopper in the new "Clarion Environment Detection" feature (compatible with Clarion 7). This bug only affected Clarion 3rd Party developers; the patch will update your sbkernel.dll and sbkernel.lib to the latest version (build 1847).

Posted Friday, April 27, 2007

PD Browse Button Lookup Updated

An update to PD Browse Button Lookup is now available. This update adds CtrlUP and CtrlDown as additional keys for scrolling to the next and previous records while in the entry field. This is especially useful when you are using the Enter Key instead of Tab key to navigate fields as the SV class that does this also overrides the use of the up/down arrows. The update also fixes a bug where the lookup locator was incorrectly set when calling the browse after using the up/down arrow. This update also includes the Add/Edit control template that creates a button to call the a browses form directly. This allows you to quickly add a record if one does not exist for to access additional information that might be on the record.

Posted Monday, April 23, 2007

Clarion Desktop 3.72

Clarion Desktop 3.72 has been released.

Posted Monday, April 23, 2007

J-Fax 1.50

J-Fax 1.50 has been released. New in this release: C7 support; Vista support (Business edition); You can now turn off the "MS Fax does not appear to be installed" message; If MS Fax is not installed, J-Fax will now remove the "Fax" option from your reports; Cover page size increased to 5000 characters.

Posted Monday, April 23, 2007

Free Blowfish Code

Strategy Online's Blowfish code is still free. Now C7 and Vista compatible. Includes an improved example EXE, as well as rewritten documentation.

Posted Monday, April 23, 2007

J-Skype 1.30

J-Skype 1.30 has been released. New in this release: The AppToApp example now supports Blowfish encryption; The Skype Robot example has been moved into the main example APP; Vista compatibility; C7 compatibility.

Posted Monday, April 23, 2007

PrintWindow 1.10

New in PrintWindow 1.10: Global settings for automatically replacing window background graphics with high definition report background graphics; Small Indentation added to "entry" field contents; Some debug code removed; Installer compatible with Clarion Desktop; Now you can change the installation folder in the installer; Installation message verifying the registration of the template in the clarion Registry.

Posted Monday, April 23, 2007

Evolution Report Export 1.12

Evolution Report Export 1.12 is now available. New in this release: Integration with Tintools, Icetips Previewer; Option to apply Formats (Text Color - Background - Font Name - Font Size) to the Titles/Headers/Details for export to Excel; Fix for compiling Legacy apps in Local Mode. Available at ClarionShop for US\$89.

Posted Monday, April 23, 2007

BST 4.23

BST 4.23 is now available for download.

Posted Friday, April 20, 2007

BoTran 2.4

BoTran 2.4 is now available. This release adds direct picture assignment for entry controls and support for C6 List header sort and LM stuff.

Posted Friday, April 20, 2007

CoolFrames 1.11

CoolFrames 1.11 beta is now available. This release fixes a bug where CoolFrames broke text field mouse actions on MDI windows.

Posted Wednesday, April 18, 2007

CHM4Clarion Released

CHM4Clarion makes it easy to change from HLP to CHM with any 32-bit version of Clarion versions 2.0 through 7.0. Implementation requires just four lines of code added to your EXE APP (no changes to DLL Apps). The choice between HLP and CHM can be made at runtime which makes it easy work on the conversion to CHM while customers still use HLP help. To assist in the conversion to CHM includes a FindHlp utility to search your source code and find all your HLP context strings, then verify they exist in the CHM topics list. This insures no dead help links. Also included is the SlideShowCHM utility to let you view a list of context strings in a slide show that opens them in the CHM. Carl will be on vacation 4/19 - 4/26 so tech support will probably not be available.

Posted Tuesday, April 17, 2007

Class Anatomy 1.6.0

Class Anatomy 1.6.0 contains a fix to a where any word containing 'END' would be interpreted as the end of whatever was being parsed (class/queue/group). This could result in classes simply not being read.

Posted Tuesday, April 17, 2007

Clarion FreeImage 3.9.3.2

Clarion FreeImage project version 3.9.3.2 is now available. This update corrects an error in the file dialog class's SetDialogFilterNulls method. This error could cause a GPF or other unexpected behavior when displaying the file dialog. This build also corrects the Name attribute on three prototypes for the FreeImage DLL.

Posted Tuesday, April 17, 2007

Clarion Desktop 3.60 Beta

Clarion Desktop 3.60 Beta is now available. As always, there is a 100% free version available as well. New features include: Clarion Desktop can now display links to your 3rdParty accessory documentation; Accessory installers that come in ZIP format are now unzipped for you, and the

unzipped installer is then launched - Clarion Desktop Pro Edition can even unzip password protected ZIP files for you; You can now set where the Download Manager stores and archives your 3rdParty downloads.

Posted Tuesday, April 17, 2007

FullRecord 1.82

Changes in FullRecord 1.82 include: New Record Inspect feature (inspects changes for a particular record from a button, or Hot key, on a Browse or Form); New local disable of code generation for calling Audit Inspect Browse; New version field on Audit structure, on preparation for incoming feature; Field inspection window now uses the field picture to show the data; Global "Variables" moved from General tab to their own tab; On Update procedure for upgrading record structure inside Audit file, proper support for audit record stored in blob field, with or without compression. Note: On multi-DLL systems a full recompilation is needed.

Posted Monday, April 16, 2007

CapeSoft WorldTour 2007 - Around the World in 18 Days

Registrations for the Johannesburg leg (3-5 May) close on Wednesday 18 April. The registrations for Sydney are closed, but you can still register for Las Vegas and Cambridge - but those will need to close soon, so don't delay registering for those venues.

Posted Monday, April 16, 2007

Simsoft Dates Class and Templates Demo

There is a new Simsoft Dates demo that demonstrates how you can use these templates.

SimDatesClass templates are available from www.clarionshop.com.

Posted Monday, April 16, 2007

FileTuner 0.20.

New in FileTuner 0.20: Support for "Test" Btrieve files; Partial support for "Fix broken" Btrieve files (Method 1); Part of the core code moved from procedures to classes; Select files to Tune from local extension list. Changes and improvements include: Get Password window now identifies the file (to which the password is asked); The template now detects if you use variables and you don't load their values; Added _DoNotImport to common procedures names (Main, Welcome, etc.) so in case of accidentally import these into your app they won't overwrite your own procedures; Global variables no longer needed (moved to classes); FT.VerifyAccessToFile changed to open the file in binary mode (instead of native mode) to avoid GPF in corrupt file; ICO resources were not automatically included in the app project (fixed); "FT.CopyFile" procedure on some Btrieve files was making incorrect copies (fixed).

Posted Monday, April 16, 2007

amazingGUI Orange Pack

The amazingGUI Orange Pack includes two new themes to be used with amazingGUI: Orange and OrangeFlat.

Posted Monday, April 16, 2007

Free PDF-XChange Viewer 1.016

Build 1.016 adds full Find/Advanced Search, Typewriter mode and other features.

Posted Monday, April 16, 2007

Clarion Magazine

Creating SQL From XML With XSLT: Calling The Code From Clarion

by Bernard Groperrin

Published 2007-04-30

In the [first article](#) in this series I explained how XSLT works, and I showed how to convert XML into SQL INSERT statements using a stylesheet. In this concluding article I'll show how to integrate that code into a Clarion application.

But first, to show how simply an XSLT Transformation can be executed from within a program I made a small C# program which is included in the downloadable source zip. There is no error checking, as this is really for demo purpose, and not production, but it demonstrates how to make the code smaller and simpler, much easier to see what's going on. The relevant part is as follows:

```
void TransformButtonClick(object sender, EventArgs e)
{
    // Load the XML Document
    XPathDocument myXPathDoc =
        new XPathDocument(XMLFileName.Text);
    // Load the XSL
    XslCompiledTransform myXslTrans =
        new XslCompiledTransform();
    myXslTrans.Load(XSLFileName.Text);
    // Create the output stream
    XmlTextWriter myWriter =
        new XmlTextWriter("SQLScript.sql", null);
    // Do the actual transformation
    myXslTrans.Transform(myXPathDoc,null,myWriter);
    myWriter.Close();
    // Open the document in the text box
    TransformationResult.LoadFile("SQLScript.sql",
        RichTextBoxStreamType.PlainText );
}
```

This C# program makes use of the .NET XSLT libraries. I mentioned at the start of this article that I needed to download the XML file, but the example uses a local filename. That's not a problem - the constructor to `XpathDocument` also accepts URLs, so you can just as easily use a file on another server as input. A Clarion.NET version of the program would look very similar to this C# example.

Figure 1 is a screenshot of the C# program in action. I've blurred out the data for security reasons.

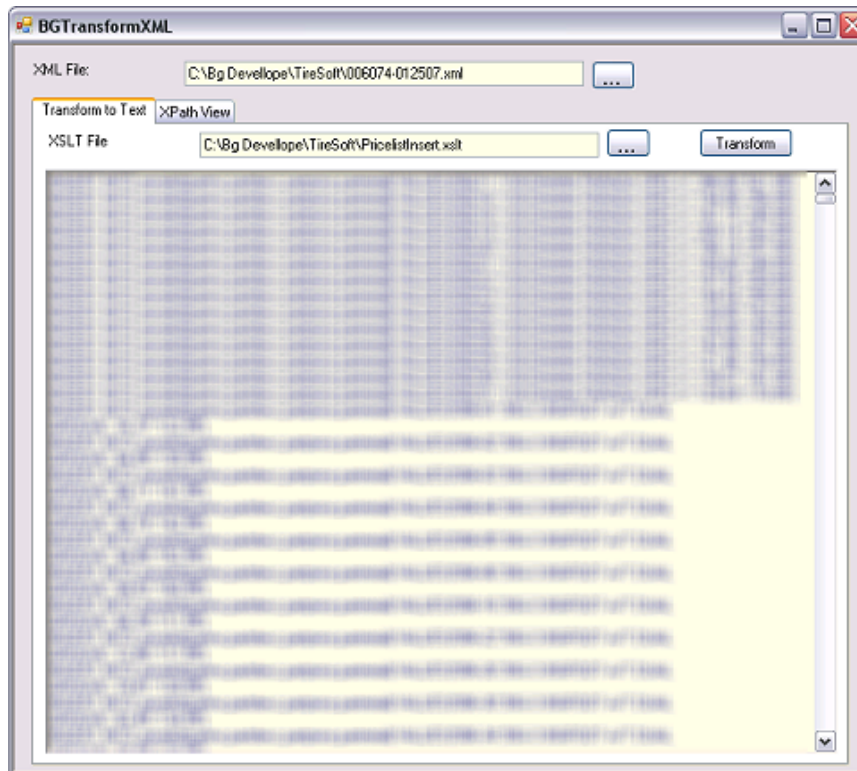


Figure 1. Transformed data (blurred for security reasons) in a C# application.

Another option is to use MS XML COM object directly from Clarion. This works beautifully, but I am not going to explain it here, as it's a bigger topic and well outside the scope of this article.

Both [XMLFuse](#) and [EasyCOM2INC](#) will allow you to use the MS XML COM objects more easily from within a Clarion application.

Clarion's own XML classes are *not* an option, as they don't validate against a DTD or Schema, nor do they allow for URL instead of a path, which is a very important feature in transforming XSLT (as I will explain later). As soon Clarion's XML classes do validation, they will be an option to consider (although they're unlikely to accept URLs as easily as local files).

Transforming the easy way

To make transforming XSLT as easy as possible I made a small OLE object, which you can call from Clarion with a single line of code. At first I wanted to make a DLL, using the inverse `P/Invoke` method. Although I found an [example](#) which works beautifully, I have been unable to get it to work with my own XSLT code. The DLL is created properly but I get an initialization error when starting my

Clarion program. It might be due to the fact that I have three (yes, three) .Net frameworks on my machine, 1.1, 2.0 and 3.0. I will experiment more with inverse p/Invoke, but this is more because of personal interest than absolute necessity.

Creating an OLE object in C#

Since I couldn't resolve the DLL loading error I reverted to an OLE object, following [Wade Hatler's excellent articles](#). I believe we all will certainly continue to create Win32 applications for a while, but .Net Framework offers a vast range of built-in functionality. And while OLE is not the best and fastest method to "consume" .Net from Clarion, it's very simple to do. Read and re-read Wade's series on .Net in ClarionMag to learn about making C# and Clarion work together.

I used [SharpDevelop](#) to create the OLE object, thereby killing two birds with one stone: learning C#, and learning how to use Clarion future IDE.

The very first thing I did was to write a little C# program to test the idea of a single line call to transform XSLT, and to validate the syntax. This is a Windows Forms program, nothing sophisticated at all.

You will find the project and files in the code to download with this paper. The downloadable source includes randomly generated data.

Once I'd verified my code worked as expected, it was time to create the OLE component to use from Clarion. Following Wade's example, I created a second Project in my solution using the Class Library type instead of a Windows Application type, as seen in Figure 2.

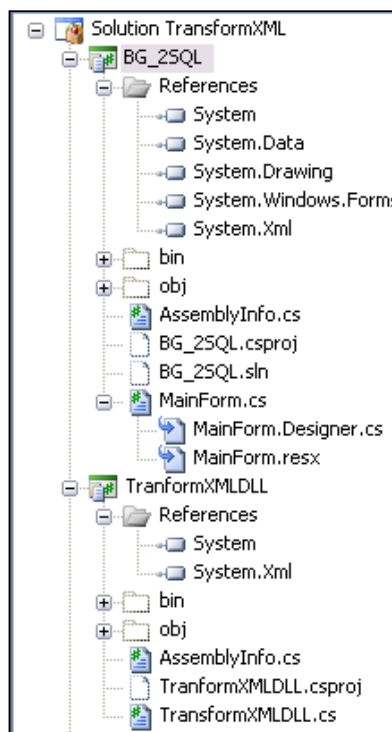


Figure 2. The C# project tree

The only real difference between the test application version and the class library version is that I wrote a method accepting three strings as parameters, for each of the files involved: the XML file, the XSLT file, and a file to store the result of the transformation:

```
public static void BGTransform(string pXMLFile, string pXSLTFile,
                               string pSQLOutput)
{
    XPathDocument myXPathDoc = new XPathDocument(pXMLFile);
    XslCompiledTransform myXSLTrans = new XslCompiledTransform();
    myXSLTrans.Load(pXSLTFile);
    XmlTextWriter myWriter = new XmlTextWriter(pSQLOutput,null);
    myXSLTrans.Transform(myXPathDoc, myWriter);
    myWriter.Close();
}
```

But first, I have a strong warning about this code and the OLE object accompanying this article: Don't use this in production! This is an example only, and you will need to add your own error checking code. For instance, if the files are not respectively XML and XSLT, or are not well formed, the transformation will not happen but you will not know why! For production work, you should validate the XML file against a DTD or schema (even if you have to create the schema yourself). Without validation any changes to the XML or XSLT formats will not be intercepted, and either your transformation will not work or it will give unexpected results. Also, for production work, you may need to pass identifications parameters to the web site where you are grabbing the XML file. The .Net Framework (like the COM object) has the properties and methods to do this, but I wanted to keep this example as simple as possible, as the focus on this article is XML transformation, not writing a production ready OLE control in C#!

Now that I have my method, I need to make this class library (compiled into a DLL), an OLE control usable from Clarion. Wade's article covers this subject thoroughly, using Visual Studio 2005; I'll explain only the differences in doing the same task in SharpDevelop 2.1 beta.

First, I need to add one line to my list of using declarations:

```
using System.Runtime.InteropServices;
```

Unlike Wade, I found the Assembly key file was required if I wanted to be able to generate the COM object. In SharpDevelop, if you right-click on the project name, as seen in Figure 3, the last choice in the drop-down menu is Properties. The third tab, Signing, is where you indicate you want to sign the assembly. Specify the key file created with the sn.exe utility (as explained by Wade).

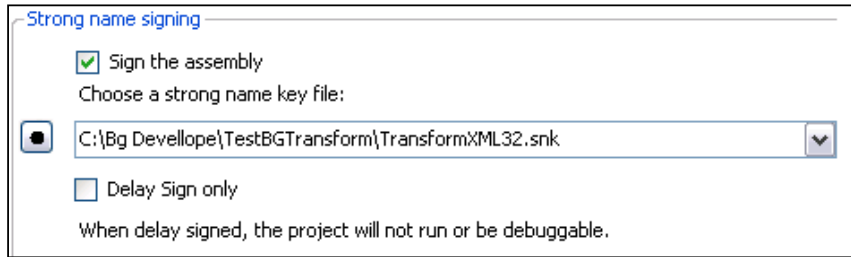


Figure 3. Signing a C# assembly in Visual Studio

As I want to have my DLL/OLE object in the same directory than the Clarion application, I indicate the correct output path from the Project properties Compiling tab, shown in Figure 4.

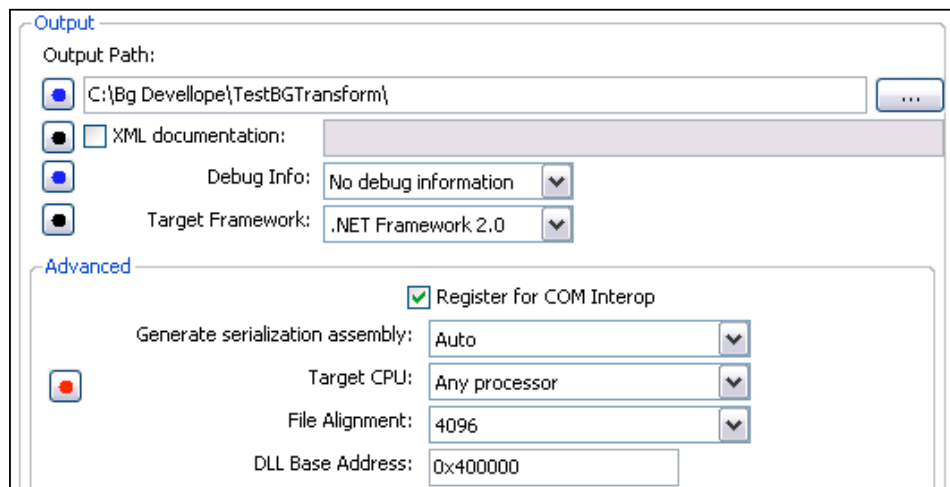


Figure 4. Setting the output path

It's also there that I check the Register for COM Interop box, which will make this .Net assembly visible as an OLE object. The very last thing to do is to indicate which class I want to expose to COM. As there is only one, the code is short, as you can see:

```
namespace TransformXML
{
    ///
    /// Description of MyClass.
    ///
    [ClassInterface(ClassInterfaceType.AutoDual)]
    public class TransformXMLclass
    {
        public void BGTransform(string pXMLFile,
            string pXSLTFile, string pSQLoutput)
    }
}
```

Now I can build the project and get a DLL which is a OLE/COM object.

The last step is to register my brand new assembly for COM. I have not found a built-in automated way to do this from inside the SharpDevelop IDE, but the Project Properties, on the Build Events tab, has provision for a Post-build event command line, where I can add the command line for Regasm.exe.

I would take this step for a complex project, but in this case I simply used the .Net Framework SDK 2.0 command prompt to run regasm and register my object. In the source zip you will find a TransformXML32.reg file, also generated by regasm, which will allow you to do this on your machine very simply.

Now, it's time to verify that all this worked, so I run MS OLE/COM Object viewer (Figure 5) to see if I can find my shiny new object.

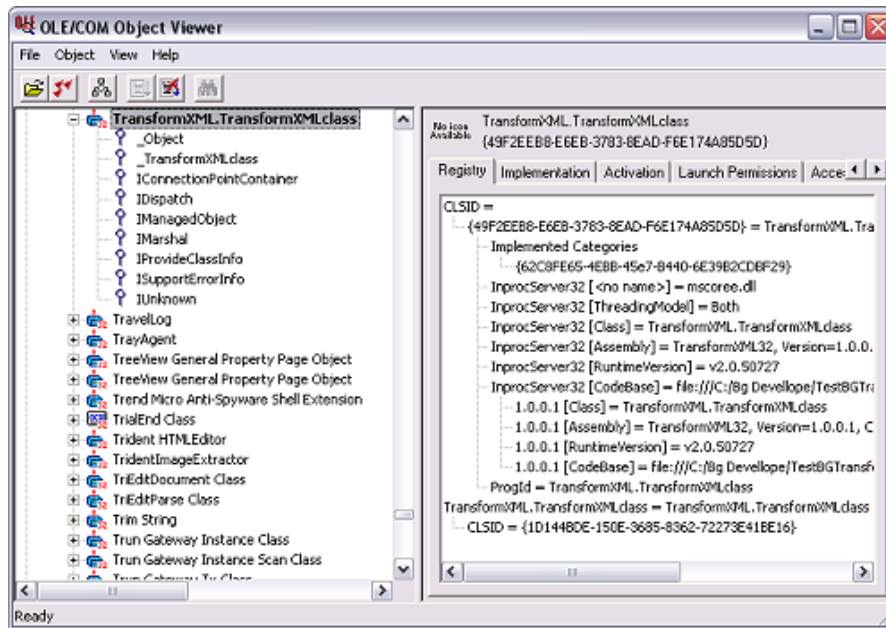


Figure 5. Finding the new COM object with the viewer

Finally, Clarion!

Finally it's time to go back to Clarion to run the transformation. As this involves only a simple method call and no user interface, the code is quite simple:

```
Transform = CREATE(0, CREATE:OLE)
Transform{prop:create} = 'TransformXML.TransformXMLclass';
```

I created an application, without a dictionary at first, to test the control. I created a window procedure and added three entry fields from local variables. To each entry field I added a DOS Lookup control template. I also added an RTF control (which could also have been a text control) and an Exit button to close the window. Then, to initialize the control, I added the above two lines in the ThisWindows.Init embed.

Transform is a long declared in the procedure data. The transformation process is in a routine, and takes only one line of code:

```

Transform{'BGTransform(' & XMLFileName |
& ',' & XSLTFileName |
& ',' & SQLFileName & ')'}

```

This routine is called from the FileDialog button for the result file, which loads the file in the RTF control after the transformation is done.

The Accepted event for the button has two lines of code:

```

DO TransformXML
RTFControl5.Load(SQLFileName)

```

As you can see, it only takes a few lines of Clarion code to do the actual transformation.

One more thing: You may recall that I said at the beginning of this article that my goal was to get those data into a SQL table. My relational database engine of choice is PostgreSQL, so I will show how to do this with PostgreSQL, but most of the SQL servers allow bulk import, so you should be able to adapt this technique to your specific case.

You could, as I have done at first, use XSLT to generate INSERT INTO statements, which you can then process in a loop with PROP:SQL. And it might be your only choice if your database does not have a batch import feature, or if you have to do this from a station not connected directly to the server, but only via TCP/IP.

If you are doing this import from a station on the LAN, or even better, from the server itself, what follows is what I believe to be the best way to do a batch import.

PostgreSQL has a COPY command allowing a bulk import from a text file. The caveat is that this is executed on the server, not the workstation, so the file has to be available to the server, with a path from the server point of view. Which means either I have to copy the file on the server, or I have to use a path meaning something to the server; the file has to be on a shared local drive, and I need to give to the server the UNC path. I could also map my local drive to a drive letter on the server, but this can quickly be a nightmare if I have to help my users to do this. It also won't work if my server is Linux, although the UNC method should be fine.

Another issue is that in PostgreSQL the backslash is a special character, so if you want to include backslash in a string, you have to escape it. I found much simpler to replace backslashes with forward slashes, which work as well in a UNC path.

If you want to experiment with this method with PostgreSQL you need to create a database, then create a single table (see the script provided in accompanying download), then import that table in a dictionary and set that dictionary in your application global properties (assuming you followed my example and created the initial application without a dictionary).

Once that is done you can create a browse box below the text control to display just-imported records in the SQL table.

Again, this requires very little manual code:

```
ConvertUNCtoSlash ROUTINE
```

```
! For postgresql, \ is a special character which would need
! to be escaped. I do the conversion here, which is simpler
! than escaping backslashes
```

```
LOOP I# = 1 to len(SQLFileName)
```

```
  IF sqlFilename[I#] = \'
```

```
    sqlFileName[I#] = '/'
```

```
  END
```

```
END
```

```
LoadIntoDB Routine
```

```
DO ConvertUNCtoSlash
```

```
setcursor(cursor:wait)
```

```
Start# = Clock()
```

```
! a single line of code to insert the records, VERY quickly.....
```

```
SEND(Pricelist,'COPY pricelist FROM "' & SQLFileName & "'')
```

```
IF FileErrorCode()
```

```
  Message('Error: ' & FileError() & '<13,10>' & 'SQL: ' |
```

```
    & Pricelist{prop:SQL} )
```

```
End
```

```
BRW7.resetfromFile()
```

```
SELECT(?list,1)
```

```
End# = Clock()
```

```
endclock = Clock()
```

```
Total# = End# - Start#
```

```
?Button5{prop:text} = RECORDS(Pricelist) |
```

```
  & ' records processed in ' & Total# & '/100th of a second....'
```

```
Setcursor()
```

About half of the code in LoadIntoDB routine is there to evaluate how long it takes to process the records. Inserting the records takes only one line of code.

I changed my initial XSLT style sheet to have tabs between the fields, as this is the default separator for PostgreSQL, and removed the SQL statements. Both style sheets are provided in the accompanying code.

Figure 6 shows the final result: 81/100th of a second to both parse the XML and insert 2648 records in the database on the LAN. As inserting the records took 0.61 seconds, parsing took 0.20 seconds.

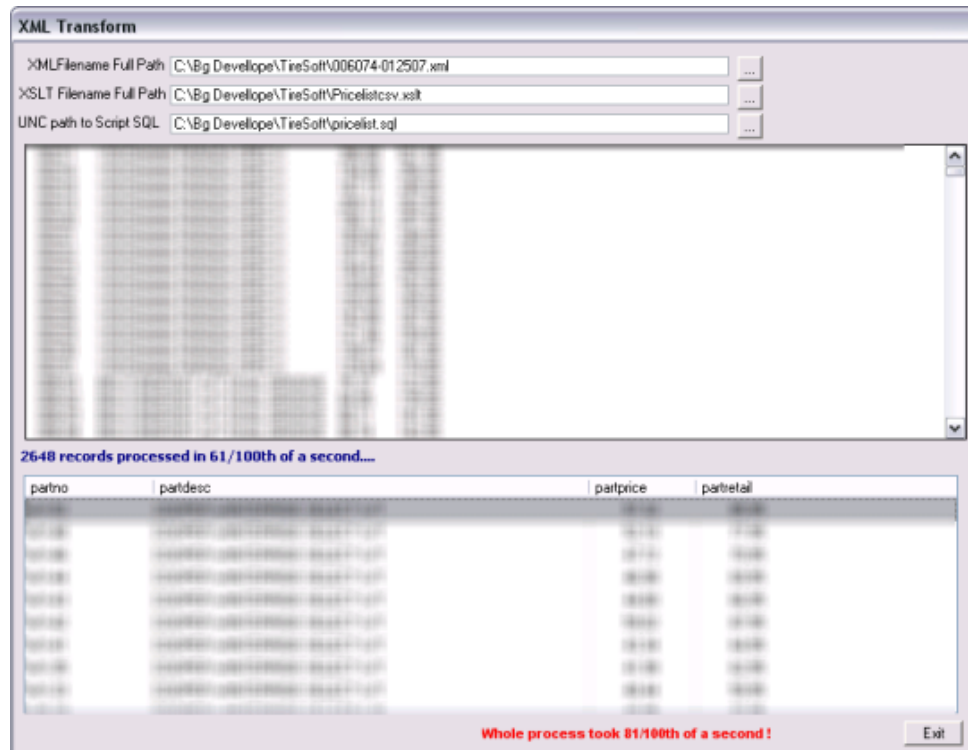


Figure 6. The Clarion program using the .NET library

Where to next?

As noted above, one advantage of this method, is that your files (XML and XSLT) do not need to be local, as this technique will work just as well with URLs as with local files.

In the accompanying source code you will find a modified C# example with an hard coded URL to a style sheet on my server, allowing you to explore the structure of *any* XML file.

There are at least two advantages to using the .NET classes as demonstrated in this article:

- You don't need additional code to connect to the Internet and download the file.
- If your XSLT file is on your own server, rather than shipped to clients, you can make changes as needed without having to ship or to recompile anything.

I hope this article has inspired you to learn more about XSLT. For further information I encourage to to explore the following resources. And feel free to email me if you have any questions.

[Download the source](#)

Resources:

- [SharpDevelop \(#D\)](#)
- [Wade Hatler's paper](#)

- [XMLSpy](#)
- [EditiX](#)
- [XSL Transformations](#)
- [W3Schools](#)
- [MSDN](#)
- [O'Reilly](#)
- [XMLFuse](#)
- [EasyCOM2INC](#)
- [PostgreSQL](#)

[Bernard Groperrin](#) is a native of France, and has been a big fan of everything American since his teens. He began visiting the US in 1996, and moved to San Antonio in 1998, where he discovered his love of Mexican food. He and his lovely wife Gloria, who he met in Tulsa, now live in California. Bernard has been programming and designing software and databases for 14 years, primarily with Clarion. His hobbies include flying radio-controlled airplanes and riding his motorcycle. He loves aircraft of all kinds, and can't miss an opportunity to fly, whether it's in a glider, a World War II trainer, or a general aviation Piper or Cessna.

Reader Comments

[Add a comment](#)

Clarion Magazine

Coping With Vista - There's A Manifest In Your Destiny, Part 2

by Jane Fleming

Published 2007-04-26

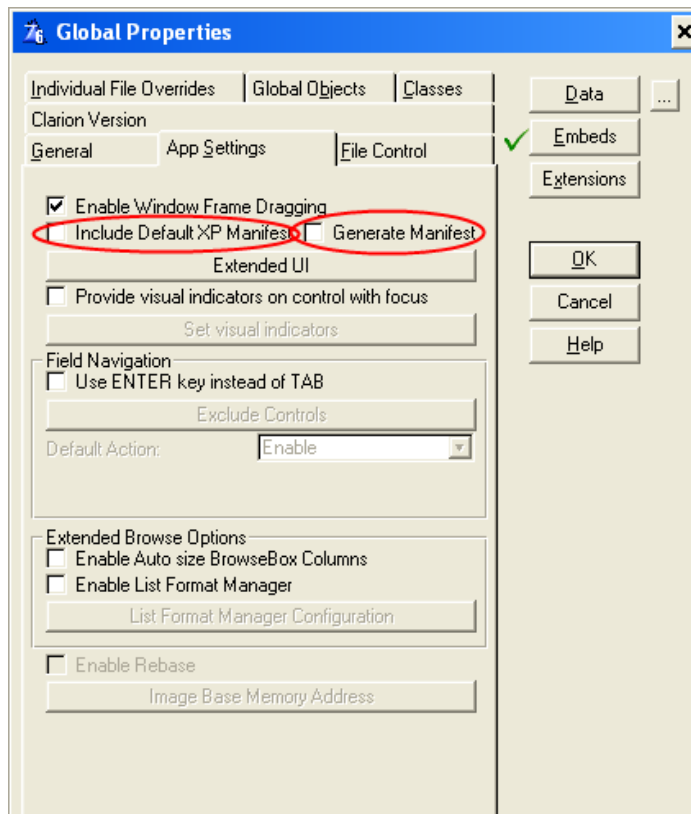
In the [previous installment](#) in this series I covered Vista's use of security identifiers, and described the process by which Vista decides when an application needs to have its security level elevated. But rather than trust Vista for this choice, you should be controlling your application's security levels with Vista manifests.

Manifests

Clearly, if you do nothing to your applications you're trusting the user to know what rights your program needs, and to know how to give your program those rights, or you're trusting Vista's "voodoo" to make the right determination. If nothing else, I hope the above convinces you that the only way to maintain some order and sanity is:

1. Sign your applications
2. Include a manifest so Vista knows what your intentions are.

A manifest is a specially-formed XML file. There's nothing magic about the manifest, but some of the text is case-sensitive. Clarion 6.x can link in an XP-compatible manifest, or generate one as a standalone file. But I'm going to suggest you leave both of those boxes unticked and link in your own manifest (Figure 9).



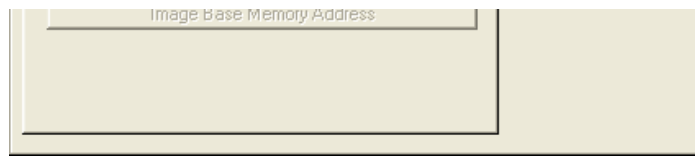


Figure 9. Start leaving both of these boxes unticked

I keep four basic manifests available: two for when I want an XP look and two for when I don't. For each set of two, one runs the app as a regular user, and one asks for administrator privileges.

The Vista portion of a manifest can specify one of three execution levels:

1. asInvoker - The application runs with the same access token as the parent process
2. highestAvailable - I haven't personally found a use for this yet in my own development. An example would be an application that requires certain privileges above those of an ordinary user, but not full administrator rights. One possibility would be a backup program, where the user needs Backup Operator privileges.
3. requireAdministrator - The application must be launched with the full access token of an administrator.

I've included all four of my manifests in the download with this article, but I'll only explain one in detail.

This is my bothadmin.manifest - it does both Vista and XP stuff and requires the administrator token on a Vista box:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
  version="1.0.0.0"
  processorArchitecture="X86"
  name="BeachBunnySoftware.ForScore"
  type="win32" />
<description>ForScore match scoring software</description>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
<security>
<requestedPrivileges>
<requestedExecutionLevel
level="requireAdministrator"
uiAccess="false" />
</requestedPrivileges>
</security>
</trustInfo>
<dependency>
<dependentAssembly>
<assemblyIdentity
  type="win32"
  name="Microsoft.Windows.Common-Controls"
  version="6.0.0.0"
```

```

processorArchitecture="X86"
publicKeyToken="6595b64144ccf1df"
language="*"
/>
</dependentAssembly>
</dependency>
</assembly>

```

The text in bold can be changed to whatever you want.

The Vista-specific section is contained between the <trust Info...> and </trustinfo> tags. The rest of the manifest is exactly what Clarion 6.x would generate if you tick the Generate Manifest box (Figure 9).

The difference between my bothadmin.manifest and both.manifest file is only one line. Both.manifest have level="asInvoker" rather than level="requireAdministrator".

If you don't want the XP look, you can use vista.manifest or vistaadmin.manifest.

Adding A Manifest To Your App Using C6.x

The easiest way to add a manifest is to have Clarion 6.x link in the manifest file when you compile your app (Figure 10). I'm saying Clarion 6.x because while the Clarion 5.5 IDE will let you put a manifest file in that section of the project tree, it does not link in correctly. How can I tell? Well, for one thing when I run the app on Vista, Vista doesn't pick it up. For another, my resource editor shows the manifest is linked in as an IMAGE rather than a manifest.

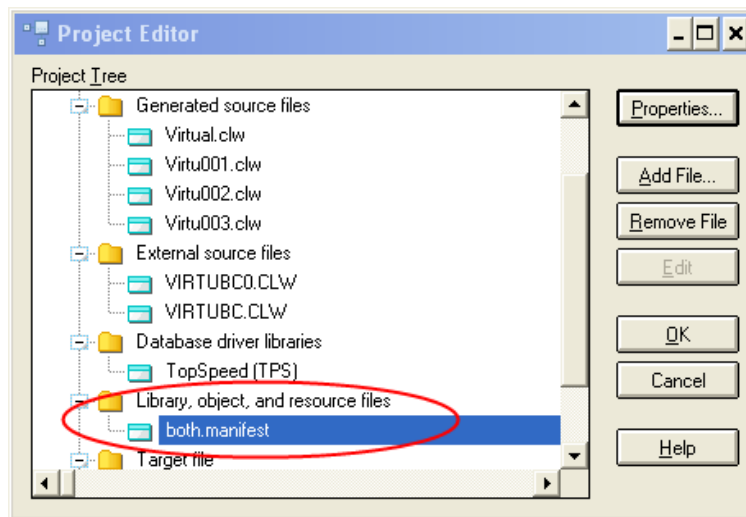


Figure 10. Linking in a manifest using Clarion 6.x

Manifests and resource editors

What's a resource editor? As you might guess, it's something that lets you view and modify resources in an application (or DLL, OCX, etc.) A really excellent *free* resource editor is Colin Wilson's XN Resource Editor. You can download it [here](#) or Google it by name.

Modifying an existing manifest is easy with a resource editor. Inserting a new one is a little trickier. The following screen shots show the process of inserting a manifest into an app created with C55H-PE. I haven't used Clarion 5.5 in quite a while, so this is provided as an illustration; it's not something I've tested thoroughly. But this app does show the XP look in XP, and both the XP look and an elevation prompt in Vista.

As a sample program, I compiled the appdemoa.app in the C55\Examples\HTMLhelp folder. Then started the XN Resource Editor, and used File | Open to open the compiled .exe file.

I'll let the captions and graphics describe the process of adding a manifest file with this tool.

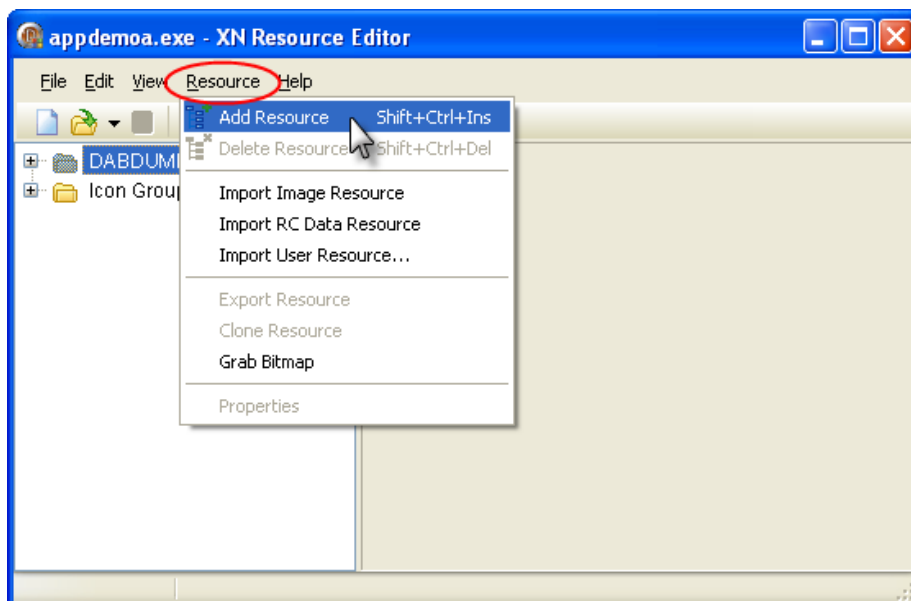


Figure 11. After opening the appdemoa.exe file in the XN Resource Editor, I click Resource, then Add Resource

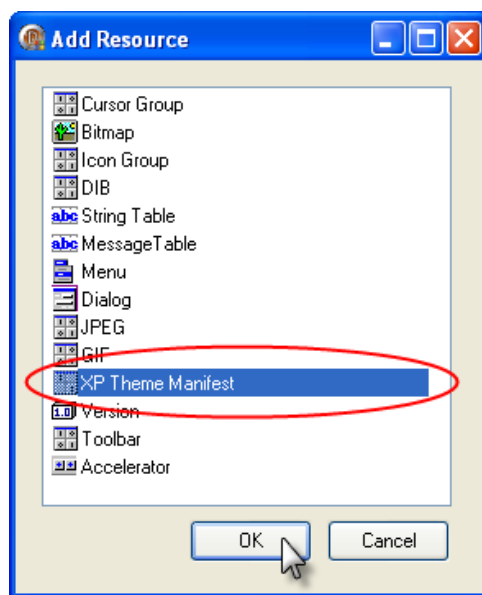


Figure 12. I Select XP Theme Manifest, then click OK

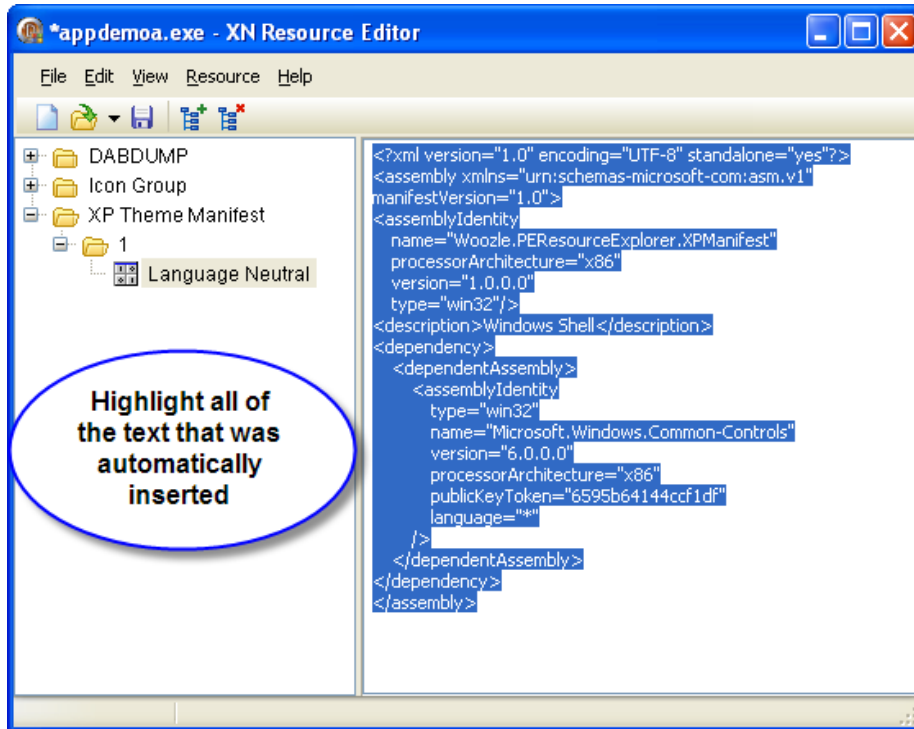


Figure 13. I'm going to replace his default manifest text with my own

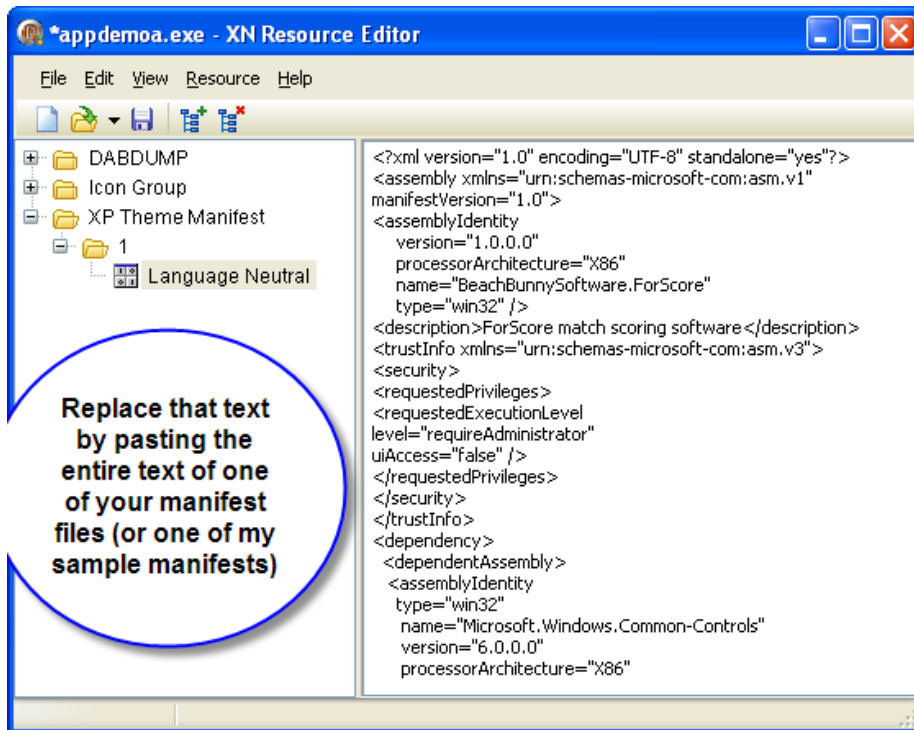


Figure 14. I've replaced his default text with the text from my bothadmin.manifest file. Some of the manifest has scrolled down and is not visible in this window shot.

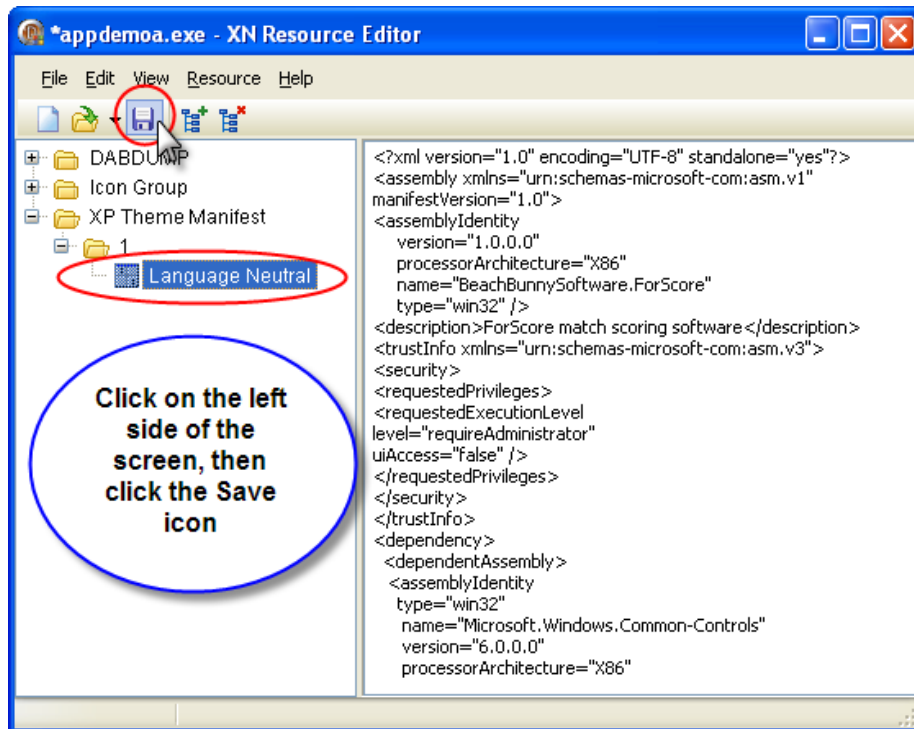


Figure 15. I save my changes and exit the resource editor

One caveat: you must not add a manifest to an app that's already signed (I'll explain why in a few moments.)

Which manifest?

So how do you decide whether to use an asInvoker or a requireAdministrator manifest? I think the best idea is to separate out functions that need administrator privileges and to put them in a separate application.

Then your user isn't bombarded by UAC windows each time he runs your app. That's slightly easier said than done, though.

As you'll recall from the first article in this series, the token that a process is going to use is determined when it is launched. That's why you can't get an elevated Windows Explorer window (unless you use the trick of launching the Explorer window as a separate process, which I described in that article.)

You can easily create an admin app with its own shortcut, but things get trickier if you want to launch that application from within another app running as a regular user. I immediately encountered one situation where I wanted to do that. I needed certain global information written to the HKEY_LOCAL_MACHINE section of the registry when running under Vista so my app can find some folder locations.

If the user installs the app using my SetupBuilder installer, those entries can be created automatically at that time because the installation program runs under an elevated manifest. If, on the other hand, he copies the program folder to a Vista machine or runs the app from a thumb drive, I need an elevated process to create those entries the first time the app runs.

I first tried to use Clarion's RUN or CHAIN commands on Vista to launch an elevated setup utility, and the results were befuddling. I got the Vista whirling-wait-cursor for a moment. Then nothing. I couldn't find a way to get RUN or CHAIN to work the way I wanted

ShellExecute to the rescue.

The source code with this article includes two little programs – normalap.prj and elevated.prj. Compile both of them. If you're signing your apps now, be sure to sign them.

If you run normal.exe on an XP machine, everything works fine. The Run, Chain, and ShellExecute buttons all launch the elevated app. On a Vista box, however, both RUN and CHAIN fail silently, while ShellExecute succeeds.

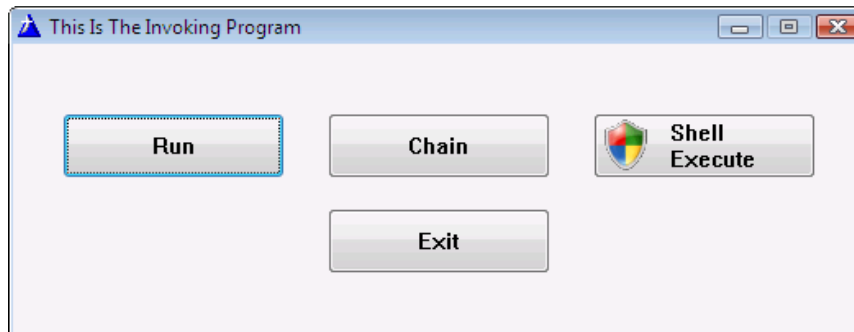


Figure 16. The invoking program. I've used the elevation shield icon on the Shell Execute button

The shield

Note the use of the shield icon on the ShellExecute button (Figure 16). It's considered good form to let users know when elevation is going to be requested, although Microsoft certainly isn't consistent about displaying the shield on buttons within Vista itself. I manually added that icon to the Shell Execute button for this app. Vista has an API call to add the shield to a button, but obviously that call wouldn't work on prior Windows versions.

If your manifest tells Vista that an application is going to need elevation, Vista will indicate that fact in the user interface. Figure 17 is a shot of a Vista directory listing, with Views set to Tiles. Note that two icons have the elevation shield in response to Vista's realizing that those two apps have requireAdministrator manifests; I didn't do anything manually to the icons. The apps that say Beach Bunny Software are ones I've signed. The other information on many of the apps comes from version resource data. MakeMatch.exe was compiled without version resource information and is unsigned..

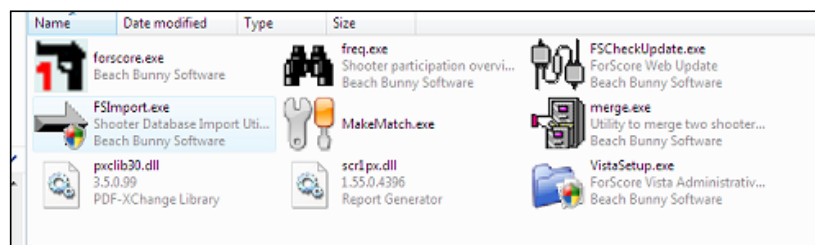


Figure 17. 'Tiles' files listing, showing shield icons, signed publisher, and other info (view full size image)

The order of things

In my [previous articles on code-signing](#) I mentioned that one benefit of a digital signature is that it verifies that the file hasn't been tampered with since it was signed. So obviously you don't want to be modifying

your file after you've signed it. That's why I said earlier that you don't want to add a manifest to an app that's already signed.

In Figure 18 I used the resource editor to change one ASCII character in my program's manifest after it was signed. As you can see, Vista does not approve (Figure 18). The same error will also show if you verify the signature in XP.

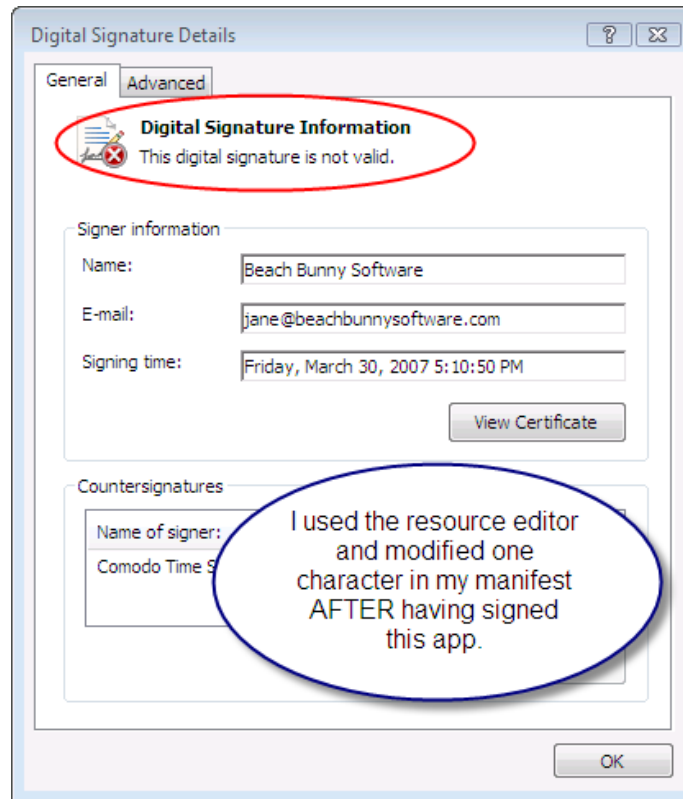


Figure 18. Modifying the manifest after the app is signed is detected as corruption

Armadillo

Those of you who use [Armadillo](#) for copy protection realize that it rather extensively modifies your app. So obviously you need to wrap your app with Armadillo prior to signing it.

The sequence you should followed is:

1. Compile the program.
2. Add the manifest (preferably done as part of the compile/link process).
3. Protect the program with Armadillo, if you use it.
4. Sign the program.

Note that this is *not* the sequence you'll be using if you use Lindersoft's amazing [SetupBuilder](#) program to embed a manifest in the programs you're distributing, unless you use a batch file within your SetupBuilder script to run Armadillo and protect your app *between adding the manifest and signing the file*. What you should do if you want SetupBuilder to code-sign your files is to embed the manifest first. Then protect with Armadillo if applicable. Then let SetupBuilder take care of the code-signing. If you use SB's default sequence and embed a manifest into a program that's already protected by Armadillo,

my experience has been that when you try to run your app either Windows will say the .exe is an invalid file or Armadillo itself will crash the file with an unpacking error message..

Personally, I prefer to handle Armadillo and code signing at compile time. In my code-signing article I showed using a batch file to sign a program or DLL. What I've wound up doing is writing a specific batch file for each EXE and DLL similar to the one I used as an example in that article. For apps that are to be protected by Armadillo, I add one line to the batch file. Such a batch file would be something like the following (three lines, with breaks added for readability):

```
"D:\program files\softwarepassport\armadillo.exe"
"D:\program files\softwarepassport\MyProject.arm" /P
e:\development\signtool sign /f e:\development\Jane.pfx
/p MyPassword /t http://timestamp.comodoca.com/authenticode
/d "My Program Description" /du http://www.beachbunnysoftware.com/
/v e:\development\MyApp.exe
pause
```

The pause lets me see if the screen has any error messages.

Then I insert the batch file into the project. (Figure 19 shows both the batch file and the manifest). I personally prefer to do this even on files that are not protected by Armadillo rather than having SetupBuilder insert the manifest and/or sign the files, as this way the file dates reflect when they were actually compiled rather than when the installer was built.

I realize that [Charles Edmonds](#) has been talking about an interface he's creating between Armadillo and SetupBuilder to incorporate Armadillo protection into the installer compilation process. That will be great. But until that happens, be aware that it's important to sign a program after wrapping it with Armadillo, but bad to insert or modify a manifest after the program has been protected.

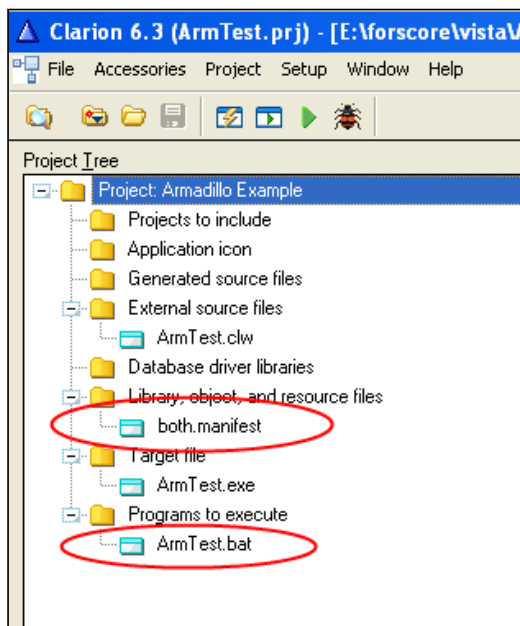


Figure 19. Add manifest, then use batch file to wrap with Armadillo and sign

Conclusion

When users log on to a Windows system, they receive a token which includes their own SID and the SIDs of any groups to which they belong. Unlike past versions of Windows, Vista gives administrators a split token. An administrator normally operates under Vista with the part of the token that gives him basic user rights. When he tries to do something that requires administrator powers, the User Account Control consent window opens to require his approval. Once a process is started with the rights of a given token, it cannot be further elevated.

Vista can try on its own to determine which applications require elevation, but it's much better to take control of the decision process by using manifests properly. And code-signing has become even more important so that users aren't confronted by scary orange "An unidentified program wants access to your computer" elevation screens.

Coming up

Some really weird things can happen to applications that don't use manifests properly. Join me next time for a look at *Vista's Virtual Reality*. Things are not what (or where) they seem. And the pills that Vista gives you don't do anything at allâ€.

[Download the source](#)

Additional Reading

- [Administering Windows Vista Security: The Big Surprises](#): Mark Minasi's book is only 255 pages, and developers will learn a whole lot by reading a mere 150 of them or so. Do yourself a favor.
- [Windows Vista Application Development Requirements for User Account Control Compatibility](#): An extremely valuable resource from Microsoft. If this link doesn't work, go to the main [Microsoft web page](#) and search for **msdn windowsvistauacdevreqs.doc**

[Jane Fleming](#) is a college dropout who subsequently lived four years in Europe, a year and a half in Mexico, and three years in India, and later taught yoga for a living in California (she's been vegetarian since 1970). She developed circuits and wrote assembly code for several embedded microcontroller projects during the 1980s. She began using Clarion Professional Developer for in-house projects back when Clarion was running display ads in InfoWorld and has used it very intermittently since. She is a former Microsoft Certified Trainer and taught Microsoft and Novell network administration at a business college for four years. Now widowed ten years, Jane plays [classical piano](#) and has found her m tier as a semi-retired NRA-certified pistol instructor.



Reader Comments

Posted on Wednesday, May 02, 2007 by Carl Barnes

Using Clarion RUN on a program requiring elevation will return an error 1040 which I could find no information about.

Using WinExec() will also fail and GetLastError will return the correct error 740 "The requested operation requires elevation."

The reason RUN fails is it uses CreateProcess which cannot elevate. This is all documented somewhere on MSDN, sorry can't find it right now.

This article "The Windows Vista Developer Story: Application Compatibility Cookbook" is good reading:

<http://msdn2.microsoft.com/en-us/library/aa480152.aspx>

If the user fails to authorize ShellExecute will return 5 and GetLastError of 1223 "The operation was canceled by the user". So you want to spot that error and remind the user that he must press ALLOW.

If you need a RUN(,1) that waits for completion you can do that with ShellExecuteEx() and it's option to return the handle to the newly created process SEE_MASK_NOCLOSEPROCESS. You will have WaitForSingleObject() on the hProcess, the don't forget to CloseHandle().

[Add a comment](#)

Clarion Magazine

Sending Messages With NOTIFY And NOTIFICATION

by Larry Sand

Published 2007-04-25

In the first three installments in this series I've looked at the basics of interprocess communication, sending messages, receiving messages, and establishing communication between processes. I've also discussed sending user messages between processes. Now it's time to look at expanding this capability with NOTIFY and NOTIFICATION.

Prior to Clarion 6 you had two options to effect change in another thread: SetTarget and POST. Using SetTarget in Clarion 6 is not thread safe but both POST and NOTIFY are. Both of these functions eventually call the Windows PostMessage function internally, so the real difference is that NOTIFY can pass a long integer along with its notification code. Remember that NOTIFY accepts three parameters, a notification code, an optional thread number, and an optional long parameter. The notification code and parameter are sent as the wParam and lParam of the windows message. The message it sends is the constant EVENT:Notify. On the other hand, POST is only capable of sending a constant message.

Deriving the OnUserMessage method

The following hand coded program shows how you can derive the OnUserMessage method to use NOTIFY and NOTIFICATION to process your Interprocess message. This example is based on the previous simple example program. The source code for this example is located in ipcB.clw and ipcB.prj, it also relies on the class declared and implemented in ipcACI.inc and ipcACI.clw.

```

Program
  Include('ipcACI.inc'),Once
  Map
  End
W  WINDOW('Interprocess communication, derived OnUserMessage')
  ,AT(,,300,200),SYSTEM,GRAY
  BUTTON('Send User message'),AT(30,33,83,14),USE(?SendUserMsg)
END

```

```

ipc Class(InterprocessComs)
OnUserMessage Procedure(UNSIGNED nUserMsg, Long lParam)|
  ,Virtual,Derived

```

Larry
Sand
is
an

End

nCode UNSIGNED,Auto

nlParam Long,Auto

aUserMessage Equate(CMAG_WM_APP + 100)

Code

Open(W)

If ipc.Init(W, |

'27FF0CD5-DA68-4841-AA53-EDC40DB1B0EA')

ipc.RegisterUserMessage(|

'72A9E86F-59D8-4ca0-A970-C7458AAF05FE')

End

Accept

Case Field()

Of ?SendUserMsg

If EVENT() = EVENT:Accepted

ipc.SendUserMessage(aUserMessage, Clock())

End

End

Case EVENT()

Of EVENT:Notify

If NOTIFICATION(nCode, ,nlParam)

Case nCode

Of aUserMessage

Message('Received notification sent at: '|

&Format(nlParam,@t4))

End

End

End

End

Close(W)

Return

ipc.OnUserMessage Procedure(UNSIGNED nUserMsg, |

Long lParam)!,Virtual,Derived

Code

NOTIFY(nUserMsg, Self.nThreadNumber, lParam)

Return

The main differences between this and the previous example are highlighted in bold text. Starting at the top, you'll notice that the ipc object inherits the InterprocessComs class and then the OnUserMessage has a prototype identical to the parent's. See the Clarion documentation for a complete description of classes and methods. Using this declaration, the compiler expects to find the implementation of ipc.

OnUserMessage in the same source code module. Here it's implemented after the Return statement of the main program code.

Two variables, nCode and nIParam are declared to use with the NOTIFICATION function. Two new GUIDs are used to initialize the ipc object and register a user message. To prevent collisions with other programs registering messages, it's important that you always use guidgen.exe to create new GUIDs for use in your own programs.

When SendUserMessage is called, the return value from Clock is passed as the IParam. This example shows how to receive the value passed in IParam in your partner process. Skip to the implementation of ipc.OnUserMessage. This is a virtual method and you have the option of calling up the inheritance chain using Parent.OnUserMessage(...). You're probably used to seeing references to "before" and "after" the parent call. If you were to place your code after the parent call it would look like this:

Code

```
Parent.OnUserMessage(nUserMsg, IParam)
Notify(nUserMsg,Self.nThreadNumber, IParam)
Return
```

However, the base class method forwards the user defined message using the Clarion POST function and you don't want that to happen. The message would be sent twice, once by POST and again by NOTIFY. This implementation only calls the NOTIFY function to forward the message and long parameter (in this case the value returned by the Clarion Clock function when the message was sent) to the Clarion ACCEPT loop.

When your program, the one that's listening for a message receives it via its subclassed window procedure, the derived OnUserMessage method forwards it to the ACCEPT loop where you catch it with the EVENT () function. NOTIFY generates an EVENT:Notify message and upon its receipt you call the NOTIFICATION function to get the notification code and long parameter. Once you have the notification code (your user defined message) and long parameter, you can do something useful. In this example, the messagebox displays the time that the message was sent from the other program.

To make this example as simple as possible, the derived method uses the default thread number (zero) in the nThreadNumber property for the optional thread parameter. In a more complex application you would probably have the InterprocessComs object attached to an Application frame. Then the derived OnUserMessage would forward the message to a MDI window running on another thread. In that case you'd have a mechanism in place to track the Clarion thread number and set it with the ThreadNumber method before receiving the message. And that's exactly what I'll demonstrate next.

Refreshing a browse in the partner application

This next example shows you how to pass messages between two ABC applications. The sending application notifies the partner application with a registered user defined message that it should refresh

its browse. When the receiving application receives the message, it passes it on to the browse thread telling the browse to refresh itself. Both applications operate on the same data, and any change to the table data by one user causes the other user's browse to update. The examples make use of the NOTIFY and NOTIFICATION functions to process the inter-thread messages.

The sending application is located in the ipcBrwS.app file and the receiving application located in the ipcBrwR.app file. Both applications require the dictionary ipc.dct and the InterprocessComs class files, ipcACI.inc and ipcACI.clw.

Before you can attempt to use the class you must include its header file in the global includes. Open the global embeds and locate the After Global Includes embed point. Create a source embed and include the class header with this code:

```
Include('ipcACI.inc'),Once
```

In the same embed point add two message constants:

```
WMU_IPC_NAMES_UPDATED Equate(CMAG_WM_APP + 10)
WMU_IPC_REFRESH_NAMES Equate(CMAG_WM_APP + 11)
```

The first message constant is used by the browse to notify the application frame window that it was updated. Then the second message is sent to the receiving application to tell it to notify its browse to refresh itself.

Next, one global variable is declared and used to store the thread number of the main application frame.

```
FrameThreadNumber Long
```

The earlier example projects initialized the InterprocessComs object after the window was opened. You'll do the same thing in this application by adding the following code to the Window Manager, Init method after the Open the window (priority 8001) embed point.

```
FrameThreadNumber = Thread()
If ipc.Init(AppFrame, |
    '5E16F92F-B03C-4645-9116-C8462968AA41')
    ipc.RegisterUserMessage(|
        'EB55DB36-5D44-4000-9E5A-2A55E501747C')
End
```

Your browse will send messages to this thread and will need to know the thread number of the main frame. This thread number is written only once so there's no need to synchronize access to this global variable. The two GUIDs were generated with guidgen.exe and then used to initialize the object and register a user message. This is identical to what you've done in the previous examples.

The ABC framework's WindowManager object has a handy TakeNotify method that you'll use to receive notify messages from the browse and forward to the other application. Locate the Window Manager, TakeNotify method and create a source embed after the Parent call at priority 5001. Place this code in the embed:

```

Case NotifyCode
Of WMU_IPC_NAMES_UPDATED
  ipc.SendUserMessage(WMU_IPC_REFRESH_NAMES, Parameter)
End

```

TakeNotify calls the NOTIFICATION function for you and passes the value of the notification code and parameter in the NotifyCode and Parameter arguments respectively. This code listens for the message that the browse sends to the main application frame (more about that in a moment) and upon its receipt, the WMU_IPC_REFRESH_NAMES user-defined Interprocess message is sent to the partner application.

The browse window notifies the main application frame window that its browse was updated by sending it the WMU_IPC_NAMES_UPDATED message using the NOTIFY function. This message should be sent whenever a record is added, changed, or deleted. You'll do this by checking the return value from the WindowManager's Run method. Locate the Window Manager, Run method in the embed list and create a source embed after the Parent call at priority 8500. Place this code in the embed:

```

If ReturnValue = RequestCompleted
  NOTIFY(WMU_IPC_NAMES_UPDATED, FrameThreadNumber)
End

```

ReturnValue is equal to RequestCompleted when the user saved their changes or confirmed the deletion of the record. The notification message is sent to the thread number of the main application frame that you saved in the global variable FrameThreadNumber.

So what happens when this program runs and you change a record in the names browse?

1. The browse notifies the main application frame that a record was changed with the WMU_IPC_NAMES_UPDATED message.
2. The main frame receives the WMU_IPC_NAMES_UPDATED message in the TakeNotify method of the WindowManager.
3. The ipc object's SendUserMessage method is called to send the WMU_IPC_REFRESH_NAMES message to the receiving application (ipcBrwR.exe)

That's all there is to sending the message.

Receiving the message

For this to be useful you need to write a little code to receive and act on the message. What's necessary to receive the browse refresh message?

In your receiving application (example code in ipcBrwR.app), open the global embeds and locate the After Global Includes embed point. Create a source embed and include the class header and messages as you did in the sending application:

```

Include('ipcACL.inc'),Once

WMU_IPC_NAMES_UPDATED  Equate(CMAG_WM_APP + 10)

```

```
WMU_IPC_REFRESH_NAMES Equate(CMAG_WM_APP + 1)
```

Then add a new message constant for use in communicating the browse thread number to the ipc object in the main frame:

```
WMU_BRW_SETTHREADNUMBER Equate(CMAG_WM_App + 1)
```

Declare a global variable to store the main frame thread number:

```
FrameThreadNumber Long
```

As in the sending application, add the following code to the WindowManager.Init method after the Open the window> (priority 8001) embed point.

```
FrameThreadNumber = Thread()
If ipc.Init(AppFrame, |
    '5E16F92F-B03C-4645-9116-C8462968AA41')
    ipc.RegisterUserMessage(|
        'EB55DB36-5D44-4000-9E5A-2A55E501747C')
End
```

Note that the GUIDs are identical to those in the sending application. Remember to create your own GUIDs for your applications.

Next you'll need to derive the InterprocessComs class to derive the OnUserMessage method. Start by declaring the derived class. Locate the Local Data, Other Declarations and add a source embed with this code:

```
ipc Class(InterprocessComs)
OnUserMessage Procedure(UNSIGNED nUserMsg, |
    Long lParam),Virtual,Derived
End
```

OnUserMessage will use NOTIFY to send your refresh message to the browses' thread using the nThreadNumber property. To implement your ipc object, create a source embed in the Local Procedures section with this code:

```
ipc.OnUserMessage Procedure(UNSIGNED nUserMsg, |
    Long lParam)!,Virtual,Derived
Code
Case nUserMsg
Of WMU_IPC_REFRESH_NAMES
If Self.nThreadNumber <> 0
    NOTIFY(WMU_IPC_REFRESH_NAMES, |
        Self.nThreadNumber, lParam)
End
End
```

Return

This is similar to the code presented before implementing a derived OnUserMessage method. This one listens for the WMU_IPC_REFRESH_NAMES message and forwards it to the browse thread if the nThreadNumber property is non-zero. If you sent it to thread zero, the message would end up in the main frame's message queue. How does the nThreadNumber property get set? The browse notifies the main frame with the new WMU_BRW_SETTHREADNUMBER constant message that you declared in the global embeds. Here's how the message is received in the main frame.

Locate the Window Manager, TakeNotify method embed and create a source embed after the Parent call, priority 5001 with this code:

```
Case NotifyCode
Of WMU_BRW_SETTHREADNUMBER
  ipc.nThreadNumber = Parameter
End
```

When the TakeNotify method receives the WMU_BRW_SETTHREADNUMBER message you set the nThreadNumber property of the ipc object to the value sent in Parameter. The browse thread sends that to the main frame when its window is opened and again when the window closes.

To send the WMU_BRWSETTHREADNUMBER message locate the Window Manager, Init in your browse procedure and create a source code embed at priority 8005 (after Open Window) with the following code:

```
NOTIFY(WMU_BRW_SETTHREADNUMBER, |
  FrameThreadNumber, Thread())
```

With this code the browse sends its thread number to the ipc object's nThreadNumber property through the TakeNotify method in the main frame.

Note: If you allow multiple instances of the browse thread then you'll need to derive or overload the OnUserMessage and ThreadNumber methods. In them you will write code to store, remove and POST the message to more than one thread.

To set the property to zero when the browse window closes, add this code to the Window Manager, Kill method at priority 9800 (after Leave Procedure Scope):

```
NOTIFY(WMU_BRW_SETTHREADNUMBER, |
  FrameThreadNumber, 0)
```

Now you need to add the code to listen for the message that the main frame will send when the browse must refresh its list box from the file. Locate the Window Manager, TakeNotify method and create a source embed at priority 5001 (after the Parent call):

```
Case NotifyCode
Of WMU_IPC_REFRESH_NAMES
  namBrowse.ResetFromFile()
```

End

Note: substitute your browse class object's label for `namBrowse` in your program. The template defaults to naming browse objects in the form `BRWnn`. When the main frame sends the `WMU_IPC_REFRESH_NAMES` message, you tell the browse to update itself from its associated file.

Here's what happens after the partner application sends the refresh message:

1. The `WMU_IPC_REFRESH_NAMES` message is received by the `ipc` object's `OnUserMessage` method.
2. `OnUserMessage` forwards the message to the browses' thread using `NOTIFY`.
3. The browse receives the forwarded message in `TakeNotify` and updates itself from the file.

The browse is responsible for sending the notification message to the main frame with its thread number when it opens and zero when it closes.

Try compiling and running both `ipcBrwS.app` and `ipcBrwR.app`. Open the browse window in both applications. Then add some records to the `ipcBrwS.app` and you'll see them appear in the `ipcBrwR`.app's browse. If you want bi-directional communication, you'll need to implement the sending and receiving bits in each application. Or you could implement this in one application, and simply run two instances.

So far in this series I've sent interprocess messages using the asynchronous `PostMessage` function. These messages are sent without regard for what happens at the other end. But if you need a return value from the receiving window procedure you must use synchronous messaging functions. I'll cover that topic next time.

[Download the source](#)

independent software developer who began programming with Clarion in 1987. In addition to normal database development, he specializes in connecting Clarion to external devices like SCUBA diving computers, kilns, and satellite transceivers used in medical helicopters. In other lives, he sailed Lake Superior as the owner/operator of shipwreck SCUBA diving tours and later as a Master for the Vista Fleet. When Larry is not programming you'll find him messing about in boats, or with boats.

Reader Comments

[Add a comment](#)

Clarion Magazine

Coping With Vista - There's A Manifest In Your Destiny, Part 1

by Jane Fleming

Published 2007-04-20

Once upon a time, not so many years ago, the world was a kinder, gentler place. There was very little risk online – Usenet was much more prevalent than the web and viruses were rare. You pretty much had to insert a floppy infected with a boot sector virus into your machine or run an infected program in order to suffer any damage. Web browsing was a mostly innocuous pastime, whether you were using Mosaic or that new browser from upstart Netscape. Microsoft, confident of its vision of the future, eschewed web development in favor of the cheerful new interface that was going to revolutionize everybody's computing experience – Microsoft Bob.

Novell ruled the network, and at first nobody took Microsoft's excursions into business computer networking very seriously. Microsoft tried to ease people into networking. "Hey, you can play Solitaire on our server!" When you created folders or shares on NT networks, default permissions were Full Control for Everyone. Kinder. Gentler. Kumbaya.

Fast forward. Now IT administrators have a number of new data protection laws to contend with, some of which carry financial and/or criminal penalties. People keep losing laptops containing customers' personal and financial information. Server 2003 was much more locked-down out-of-the-box than was 2000. Vista is a whole lot more locked-down than any previous Microsoft OS (check out Vista's new [BitLocker](#) option to address the stolen computer issue).

In the first article in this series I briefly gave my reasons for thinking Vista's User Account Control (UAC) is a good thing, and I outlined several strategies for minimizing UAC annoyances in day-to-day computing/developing. I also argued that since many users will be running UAC, we developers should do likewise.

This article will go under the hood a bit more and look at Vista UAC's impact on application development. I'll explain how to use manifests and I'll begin to show why they're important. In the last article in this series, which deals with Vista's virtual reality, I'll provide a more dramatic illustration of the importance of manifests. But first I'll explain how Vista goes about keeping track of who's doing what.

Howdy, SID

It's time to say hello to SID. SID is Microsoft's abbreviation for Security Identifier. SIDs are much like Globally Unique Identifiers, or GUIDs. Every object, such as a user account, that participates in network security has a SID. (While I'll be using the term *network*, SIDs are also used on standalone computers) In concept, a SID is analogous to the primary key field that uniquely identifies each row of data in a table – it's a datum that's normally invisible to the user, but it guarantees the program's ability to identify records uniquely. Many of us have started using GUIDs in our database structures for this purpose.

SIDs are not only for user accounts. Computers also have SIDs (which administrators need to take into account when ghosting drives onto workstations on a network, as no two SIDs may be the same).

Permissions on network file objects are granted in Discretionary Access Control Lists (DACLS), which keep track of the permissions given to different SIDs. For the Windows file system, these permissions are stored in the volume information on a specific hard drive. So if a user has permissions on various folders and shares on ten different servers on a network, for example, his permissions are stored in the NTFS file system catalogs on those hard drives and the individual computers' share systems.

Basing security on SIDs confers the same benefits as basing record identification on an unchanging primary key value. Obviously, when Betty Jones gets married and changes her name to Betty Smith, it would be an enormous chore to chase down all her permissions around a network and change them to reflect her new name. But because the permissions are given based on a SID rather than on her name, her name change causes no problems.

Conversely, if you accidentally delete Betty's user account and then recreate it identically, all her permissions will be gone because the new user account will be created with a new SID.

Not all SIDs are user-defined. The built-in administrator account in all versions of Windows, for example, has had a SID ending in "-500". Hackers over the years have found ways of enumerating SIDs to try to find the Administrator account (even it has been renamed) to attack it. As mentioned in the first article, Vista disables the built-in Administrator account by default.

To make assigning permissions more manageable, Microsoft networks (and Novell, etc.) provide for different types of user groups. The idea is that you can give a group certain permissions, and anyone assigned to that group will automatically gain the group's permissions. In Windows, each user group also has its own SID. Microsoft has long urged administrators to assign permissions to groups rather than to individual accounts. That way, if you did delete and recreate Betty's account, all you'd need to do would be to make her a member of the groups needed for her job. Likewise, if your company expanded and you needed to hire a second person whose job required her to have the same access that Betty does, life is easy – just put the new hire into those groups.

Tokens

When a user logs on to a computer or a network, a *token* is created. This token consists of the user's own SID and also the SIDs of all the groups to which he belongs (which is why if you change somebody's group membership, he needs to log off and then log on again before he'll experience the change).

When a user wants access to a resource, Windows presents his token to the system. Tokens are evaluated both for *allow* and *deny* conditions. It's like somebody going up to the door of a private club. The doorman, Bruno The Bouncer, asks "What's in your wallet?" (let me see all the SIDs in your token). Then he looks down his list. Your user account's SID may or may not be on his list, but if one or more of your group SIDs is there, you'll get the combined privileges that those SIDs authorize. On the other hand, if *any* of your SIDs are on Bruno's deny-access list, you're bounced. Deny settings override any authorized permissions.

Vista introduces one significant difference in how tokens are interpreted. Elevated users (administrators or others who have various user rights and permissions that Microsoft considers sufficiently sensitive) now have a *split token*. Split tokens are really two tokens. If your Vista user account is a member of the Administrators group, you'll log in and get a standard user token and an administrator token, but

Vista won't use the rights embodied in the administrator token without asking you first. (By the way, Microsoft does differentiate between rights and permissions, but that's not particularly germane to this discussion so I may be a bit sloppy in use of the terms.)

All right. Show of hands.

How many of you log on with an administrator account all the time every day?

I thought so. It's just too hard to get anything done otherwise. The downside is that if you inadvertently step on a virus, trojan, rootkit, or any other kind of malware, it will run with your full authority on your computer or on your network.

For years, we've been told to do as much of our work as possible under an ordinary user account so as to minimize the damage we might inflict. Windows 2000 was the first Microsoft system that really pushed this, by adding a right-click option (or Shift+right click) for Run As. The idea was that you'd log on as a regular user, and just use your administrator power when you specifically needed it. But not a lot of people used Run As. I confess to being like everyone else. I never could figure out how to make the Run As command work to configure network and IP settings, for example.

But under Vista with UAC, we're *all* running as ordinary users. That only changes when those of us who have a split token specifically tell Vista that we need to do something that involves one of those sensitive user rights. My analogy is that UAC is like a firewall for administrators – intended to minimize the damage we might inadvertently do.

Looking at elevation

If you've played with Vista, I'm sure by now you've experienced the elevation prompt. The screen dims and an "are you sure?" window pops.

Actually, there are two separate processes happening here, and they're configurable individually in your computer's Local Security Policy applet.

The dimming of the screen is part of what's called Secure Desktop. While the screen is dimmed only code-signed parts of the Vista operating system itself can interact with the user. The purpose of this is to foil some type of malware that might, for example, create its own image of the elevation prompt to trick a user into clicking Continue. In his Vista security book (referenced at the end of this article), Mark Minasi speculates on the possibility that somebody could create a custom mouse pointer that shows the pointer 40 pixels to the left of the actual pointing location, so when a user thinks he's clicking Cancel he's actually clicking Continue. The world is full of devious people with time on their hands....

After the Secure Desktop pops up, the user is greeted by one of four different color-coded elevation prompts.

What's the color code, Kenneth?

There are four major categories of consent screens.

1. The process requesting elevation is signed as being an integral part of Vista itself. The bar across the top is teal. The shield is multi-colored (Figure 1).

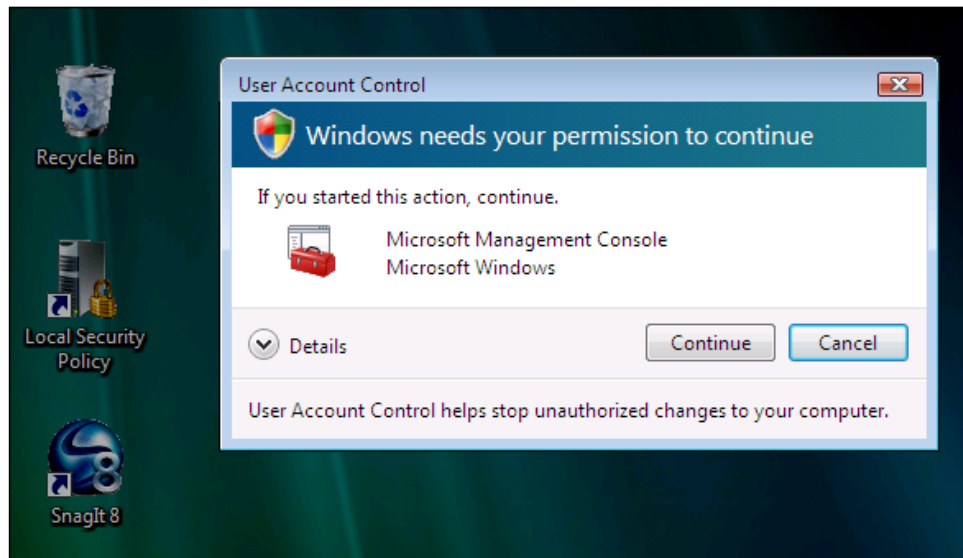


Figure 1. A signed component of Vista requesting elevation (note dimmed background)

2. The process requesting elevation is an application signed by a recognized certificate but the process is not a component of Vista (I explained certificate hierarchies in a previous article on code-signing). The bar across the top of this window is grey and the shield is orange(Figure 2.)

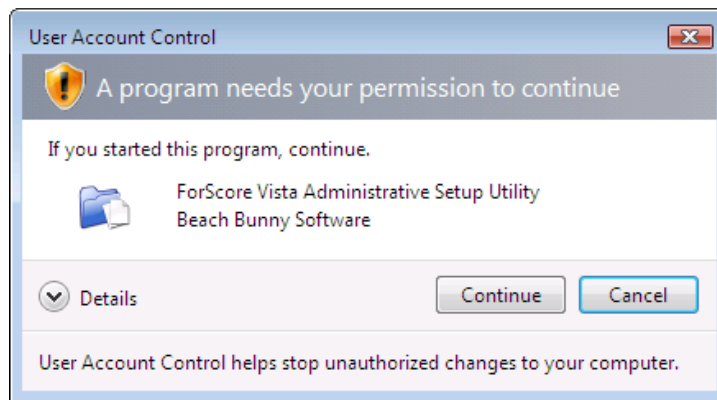


Figure 2. A signed program that isn't a native part of Vista

3. A program that isn't signed is requesting elevation. The bar and the shield are orange. (Figure 3) (Now aren't you glad you bought that [signing certificate](#)?) Note the new-style Vista "buttons". The blue-outlined box around Cancel and its explanation is the cancel button. If you move your mouse down to Allow, the box moves there.

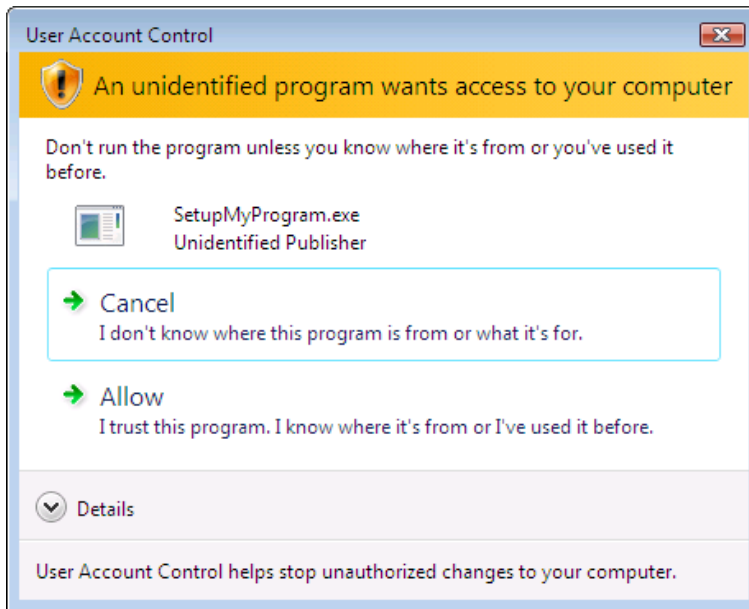


Figure 3. Unsigned program requesting elevation

Under some circumstances with an unsigned app you may see a different screen. If the User Account Control: Only elevate executables that are signed and validated policy is set, anybody trying to run an unsigned app at the elevated level (administrator or ordinary user) will see the rather cryptic screen shown in Figure 4.

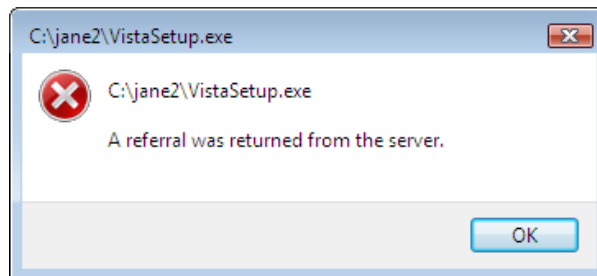


Figure 4. Unclear error message when unsigned program requests elevation.

Figures 1 through 3 assume that you're logged on as an administrator, so Vista just needs to ask your permission to elevate.

An ordinary user gets a modified screen asking for an administrator name and password. Figure 5 show this requested by a program that has a valid digital signature.

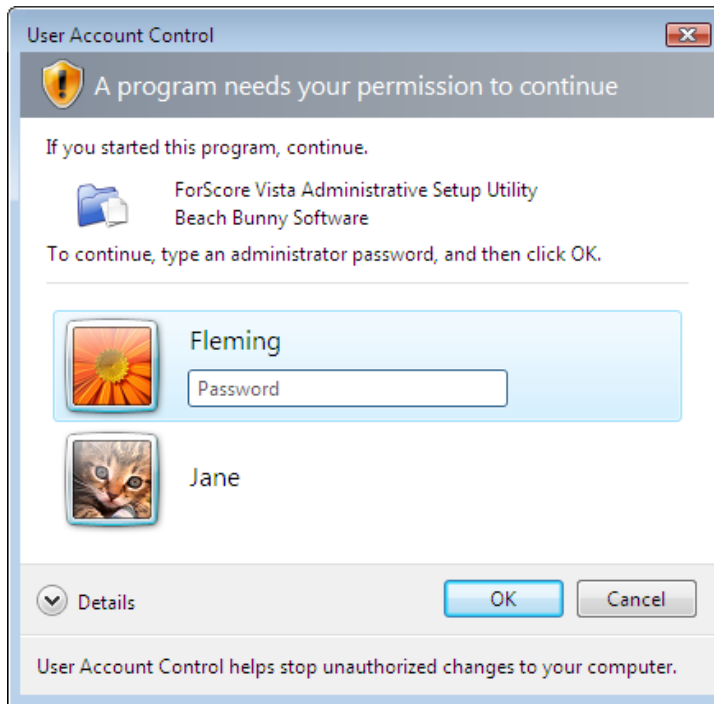


Figure 5. Ordinary user needs to supply an administrator password

4. The fourth category of consent screen is a dead-end window that's opened for a program that's denied, or a computer where ordinary users are forbidden to give credentials to elevate. The window has a red X rather than a shield, and only a "click OK and die" button (Figure 6).

Several settings might trigger the window in Figure 6. If the computer's User Account Control: Behavior of the elevation prompt for standard users is set to Automatically deny elevation requests, a user who is not an administrator will see the dead end window with no ability to enter an administrator account and password. Or if a program is blocked by a security policy, anybody (regular user or administrator) will encounter that screen as well. Such a policy can block the program by name, by hash, or block all programs signed by a specific certificate.

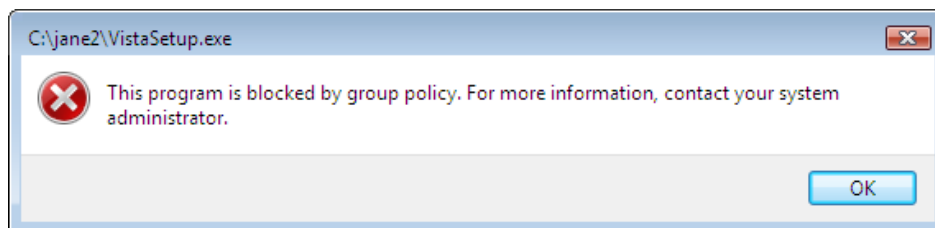


Figure 6. Policy has been set denying ordinary users the ability to elevate

How Does Vista Know To Elevate?

The elevation mechanism is a bit complicated. Mark Minasi explains it in *Administering Windows Vista Security: The Big Surprises* so I won't go into the details here.

There are several ways that Vista decides you need to see the "are you sure" elevation prompt.

1. The Program Compatibility Assistant decides that your program needs elevation.

2. You're running a setup program
3. You're running a program with a manifest that tells Vista this program needs elevated privileges.
4. You've right-clicked a program or shortcut and selected Run As Administrator
5. You've specified on the program's Compatibility tab that it must be run as an administrator, or done so on its shortcut (as I did with the elevated command prompt in the previous article in this series.)

Program Compatibility Assistant (a.k.a Voodoo)

I haven't explored the Program Compatibility Assistant, as manifests are a lot easier to understand than voodoo. You can get some information [here](#). Or Google for "Program Compatibility Assistant" if you're interested.

Setup programs

What's in a name? Compile any Clarion program, and *do not include a Vista manifest*. Copy the program to a Vista machine. It runs normally. Now rename it setup.exe or install.exe or some variation like SetupMyProgram.exe. Run it and you'll get the elevation prompt.

Manifests

You're probably familiar with manifests from applying the "XP look" to your Clarion 6.x programs. You have the option of linking in a manifest or including a manifest in the same folder with your program. To do the latter, the manifest file needs a specific name format. If my application is called MyApp.exe, the manifest file needs to be MyApp.exe.manifest.

I'll go into more detail about manifests in a moment.

Run As Administrator

If you right-click and specify Run As Administrator, you're telling Windows in the most direct way possible that you want to elevate the program's rights.

Specified for the program or shortcut

In the previous article in this series I modified a shortcut to create a command prompt that always runs elevated, so I won't go over that again here. But I was careful to modify just one shortcut, not the cmd.exe program itself.

If you're trying to track down why a program insists on running elevated, you'll want to remember the Show settings for all users button. In Figure 7, I've ticked the Run this program as an administrator box on the Compatibility tab. Whether I invoke the program from a shortcut or from the command line, henceforth it will require elevation. But this check box only affects me, not other users on this computer.

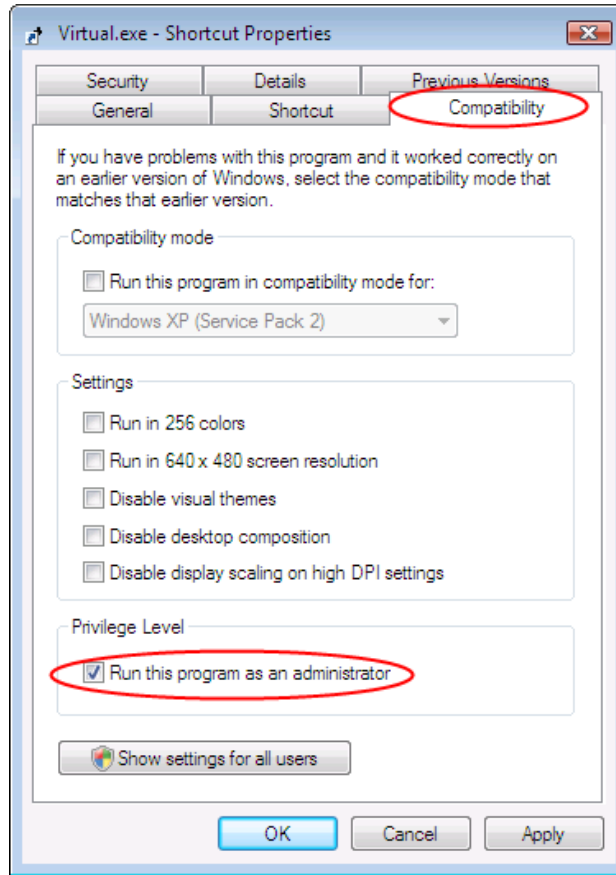
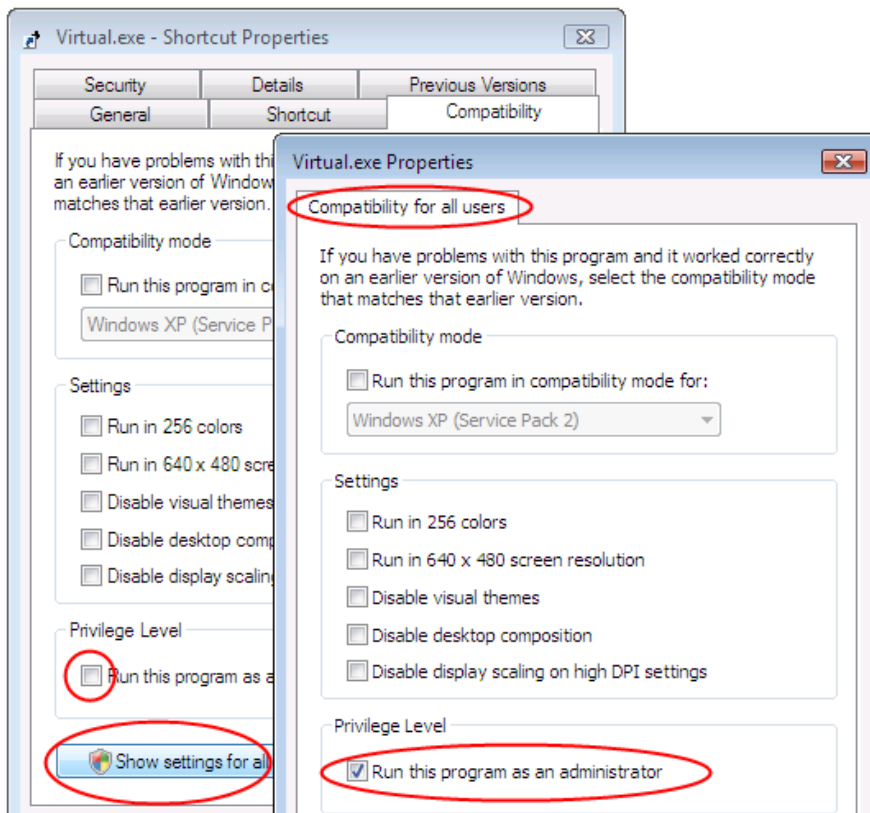


Figure 7. Run program as an administrator - but for this user only

In Figure 8, I've clicked the Show settings for all users button. If I set Run As Administrator at this level, it affects all users. But from looking at the Compatibility tab, it may not be obvious that this policy is set for this program.



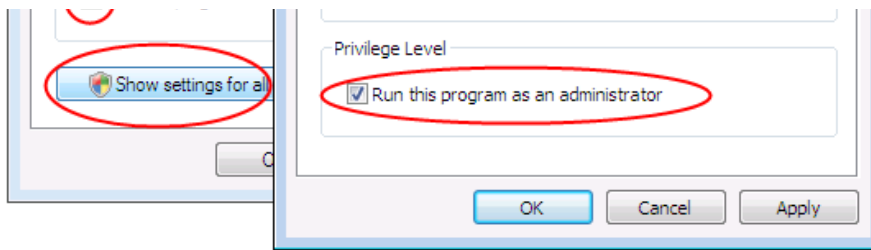


Figure 8. The "all users" setting can override the current user's setting

Clearly, if you do nothing to your applications you're trusting the user to know what rights your program needs, and to know how to give your program those rights, or you're trusting Vista's "voodoo" to make the right determination. It's much better to take control of this process by creating a Vista manifest. That's the subject of the [next article](#) in this series.

Additional Reading

- [Administering Windows Vista Security: The Big Surprises](#): Mark Minasi's book is only 255 pages, and developers will learn a whole lot by reading a mere 150 of them or so. Do yourself a favor.
- [Windows Vista Application Development Requirements for User Account Control Compatibility](#): An extremely valuable resource from Microsoft. If this link doesn't work, go to the main [Microsoft web page](#) and search for **msdn windowsvistauacdevreqs.doc**

[Jane Fleming](#) is a college dropout who subsequently lived four years in Europe, a year and a half in Mexico, and three years in India, and later taught yoga for a living in California (she's been vegetarian since 1970). She developed circuits and wrote assembly code for several embedded microcontroller projects during the 1980s. She began using Clarion Professional Developer for in-house projects back when Clarion was running display ads in InfoWorld and has used it very intermittently since. She is a former Microsoft Certified Trainer and taught Microsoft and Novell network administration at a business college for four years. Now widowed ten years, Jane plays [classical piano](#) and has found her métier as a semi-retired NRA-certified pistol instructor.



Reader Comments

[Add a comment](#)

Clarion Magazine

Creating SQL From XML With XSLT

by Bernard Groperrin

Published 2007-04-19

You've heard all about XML, and you may be even using it in some ways. Whether or not you use XML, you may be wondering what all the fuss is about. After all, it's just another ASCII file with a flexible set of markup tags, right?

Well, yes, that's right. But if that's all you know and want to know about XML, you are missing 90% of it!

And the main part you are missing is XSLT.

XSLT stands for eXtensible Stylesheet Language Transformation. The problem with the XSLT acronym, as I see it, is the S, which is for Style. This leads many people to think of XSLT as a style sheet for XML files, the way CSS files are style sheets for HTML. But the important part of the acronym is the T, which stands for *Transformation*. It is this aspect that makes XSLT so special and useful.

In this article I will attempt to show that XSLT is really much more a Transformation Language than a Style Sheet, and hopefully I will demonstrate that XML is worth a second look because of XSLT.

An XML template language

XSLT is a template language. Yes, templates, as in Clarion templates. That is, XSLT allow you to generate about anything you want from an XML file, which contains the "data".

Consider the problem of downloading an XML file, in this case a price list, and storing its contents in a SQL database. The "normal" way to do this is to:

1. Connect to the Internet
2. Download the file
3. Parse it, one way or another
4. Process the data to insert in a file

There are Clarion tools, classes, templates, for each one of those steps, and with the right tools this process work well; there is nothing wrong with this approach. But what if I tell you that you can have a single tiny procedure, which will be truly re-usable with *any* XML file, and which allows you to do all of the above in a few lines of code, really quickly? That's what this paper is all about!

Transformation

As I said above, XSLT allow you to transform XML in any text format you want. Most of the time XSLT is used to generate HTML, or XHTML, but there's no reason you can't generate an ASCII, Comma delimited file, or an SQL script, or whatever else you like. In the following example I will generate a text file (CSV) to Insert data in an SQL table. I will make this work for PostgreSQL, but you can adapt this method to the SQL database of your choice. (Although this example only creates the INSERT statements, you could also quite easily modify the XSLT to generate the CREATE TABLE script as well.)

Tools

What tools do you need to create and test an XSLT file? First, since XSLT, like XML, is stored in simple text files, Windows Notepad will do. You can test the result of your transformation by viewing the file in Internet explorer. This approach really does work, and it is a zero cost solution (assuming you're already running Windows, of course). There are many XSLT tools which will enhance your productivity. The "Rolls Royce" is XMLSpy, by Altova: <http://www.altova.com/simpliedownload2.html>

If you don't work a lot with XML-XSLT, you may need fewer features and more simplicity than XMLSpy. SharpDevelop, the IDE upon which Clarion 7 and .Net is built, has a decent XML/XSLT editor, just like Visual Studio, and SharpDevelop will also do the transformation. My guess is that the new Clarion IDE will have that same editor built in.

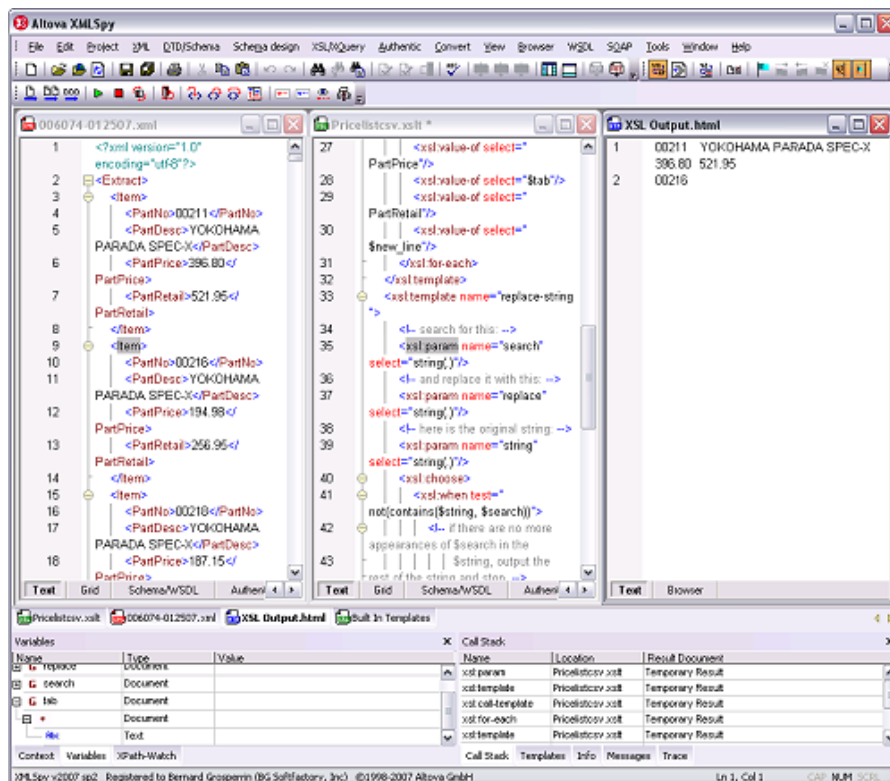


Figure 1. Altova XML Spy (view full size image)

A debugger will allow you to follow the transformation step by step, which is an excellent learning tool, as

you can see on Figure 1. XMLSpy has a debugger, as does the excellent shareware [EditiX](#) application. If you do more than just a couple of tests because of this article, I suggest you pay the license, as this is an excellent tool to work with XML/XSLT and even XSL:FO (The FO stands for "formatting object XSL:FO does not transform exclusively to a text file, but to a document containing sequences for printer , allowing, for example, to transform directly to a PDF file, on the fly. XSL:FO is beyond the scope of this article.)

Where to start?

The example XML file I will use is an extract, with some changes, of a real, proprietary XML document, which is used to update prices. I changed the names and the prices, and use only a few records instead on the 2500 something the original file contains. The size reduction is only for convenience; even with about 2500 records the processing shown below is lightning fast. On my machine it takes approximately 0.6 seconds to parse the XML and import the data.

Here is a short excerpt of the XML file:

```
<Extract>
  <Item>
    <PartNo>00211</PartNo>
    <PartDesc>SCHMURZ GRAND_SPORT</PartDesc>
    <PartPrice>396.80</PartPrice>
    <PartRetail>521.95</PartRetail>
  </Item>
  <Item>
    <PartNo>00216</PartNo>
    <PartDesc>SCHMURZ GRAND TOURISMO</PartDesc>
    <PartPrice>194.98</PartPrice>
    <PartRetail>256.95</PartRetail>
  </Item>
  <Item>
    <PartNo>00218</PartNo>
    <PartDesc>MICHELIN M+S</PartDesc>
    <PartPrice>187.15</PartPrice>
    <PartRetail>245.95</PartRetail>
  </Item>
</Extract>
```

To transform an XML file into some other format I need an XSLT file. If I open EditiX and create a new XSLT file (choose XSL 2:0 for XML), and replace `xsl:output method="xml"` by `method="text"` (see

the bold text below), I'll have the basic structure I need to start writing my XSLT file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions"
  xmlns:xdt="http://www.w3.org/2005/xpath-datatypes"
  xmlns:err="http://www.w3.org/2005/xqt-errors"
  exclude-result-prefixes="xs xdt err fn">
  <xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">
  </xsl:template>
</xsl:stylesheet>
```

As you can see, an XSLT file is itself an XML file where each tag name has an xsl: prefix.

There are books, tutorials, and web sites explaining everything you may ever want to know about XSL so I won't go into all the details. For more information see the Resources section at the end of the article.

The work really starts at <xsl:template>, as this element will contain the set of rules to apply to a specified node. As I want to work on the whole file and the root element is Extract I change this tag to

```
<xsl:template match="Extract">
</xsl:template>
```

There is nothing inside this element yet; if you were to try a transformation now (that is, run the XSLT file), you would just get an empty file. But if you change the tag to this:

```
<xsl:template match="Extract">
  here is my first transformation!
</xsl:template>
```

a transformation will have the sentence "here is my first transformation!" inside your result file.

OK, it's nothing spectacular, or useful. Yet.

Take a closer look at the XML file above. Inside the root element Extract there are a number of <Item> elements, and each <Item> element contains four elements, <PartNo>, <PartDesc>, <PartPrice> and <PartRetail>. The XSLT file needs to be able to loop through the XML file, and for each <Item> element get the values of each of the four elements it contains. I'll use the <xsl:for-each> tag for the loop:

```
<xsl:template match="Extract">
  <xsl:for-each select="Item">
```

```

</xsl:for-each>
</xsl:template>

```

The xsl:for-each tag defines this XSLT file's first rule: do something for each <Item> element.. And the something is to write down the value of the sub-elements with the <xsl:value-of> tag:

```

<xsl:template match="Extract">
  <xsl:for-each select="Item">
    <xsl:value-of select="PartNo"/>
    <xsl:value-of select="PartDesc" />
    <xsl:value-of select="PartPrice"/>
    <xsl:value-of select="PartRetail"/>
  </xsl:for-each>
</xsl:template>

```

If I transform my XML file with this XSLT, I obtain this:

```

00211SCHMURZ GRAND_SPORT396.80521.9500216SCHMURZ
GRAND TOURISMO194.98256.9500218MICHELIN M+S187.15245.95

```

The output is simply a long string, without any space, containing data extracted from the XML file. The XSLT output is unusable and ugly, but the transformation works. Clearly the output could benefit from some line breaks. Declaring a linefeed variable in this XSLT script:

```

<xsl:variable name="new_line" select="&#xA;" />

```

The variable is new-line, and its value #xA; which is the HTML equivalent of a linefeed + carriage return or, in Clarion terms, '<13><10>'. You won't always need this particular technique. For instance, XSLT is normally used for HTML where carriage returns are a matter of convenience only and are not required, but this XSLT file is generating a pure text file so I need a way to mark the end of a record. I could use the #xA; value directly, but the variable makes the code a little more readable.

Here's the full script so far:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions"
  xmlns:xdt="http://www.w3.org/2005/xpath-datatypes"
  xmlns:err="http://www.w3.org/2005/xqt-errors"
  exclude-result-prefixes="xs xdt err fn">
  <xsl:output method="html"/>

```

```

<xsl:variable name="new_line" select="'&#xA;'" />
<xsl:template match="Extract">
  <xsl:for-each select="Item">
    <xsl:value-of select="PartNo"/>
    <xsl:value-of select="PartDesc" />
    <xsl:value-of select="PartPrice"/>
    <xsl:value-of select="PartRetail"/>
    <xsl:value-of select="$new_line" />
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

The result looks a little better:

```

00211SCHMURZ GRAND_SPORT396.80521.95
00216SCHMURZ GRAND TOURISMO194.98256.95
00218MICHELIN M+S187.15245.95

```

To create a comma delimited file, simply add commas and quotes as needed:

```

<xsl:for-each select="Item">
  <xsl:value-of select="PartNo"/>
  <xsl:text>,"</xsl:text>
  <xsl:value-of select="PartDesc" />
  <xsl:text>",</xsl:text>
  <xsl:value-of select="PartPrice"/>
  <xsl:text>,</xsl:text>
  <xsl:value-of select="PartRetail"/>
  <xsl:value-of select="$new_line" />
</xsl:for-each>

```

This XSLT code gives the result:

```

00211,"SCHMURZ GRAND_SPORT",396.80,521.95
00216,"SCHMURZ GRAND TOURISMO",194.98,256.95
00218,"MICHELIN M+S",187.15,245.95

```

But I said I wanted to generate an SQL script to directly insert this data into a table. Let's do it, then:

```

<xsl:for-each select="Item">

```

```

<xsl:text>INSERT INTO pricelist(partno,partdesc,partprice,
    partretail) VALUES('</xsl:text>
<xsl:value-of select="PartNo"/>
<xsl:text>','</xsl:text>
<xsl:value-of select="PartDesc" />
<xsl:text>','</xsl:text>
<xsl:value-of select="PartPrice"/>
<xsl:text>','</xsl:text>
<xsl:value-of select="PartRetail"/>
<xsl:text>');</xsl:text>
<xsl:value-of select="$new_line" />
</xsl:for-each>

```

Run the XSLT again and you get the following output, which is a valid set of PostgreSQL INSERT statements:

```

INSERT INTO pricelist(partno,partdesc,partprice,partretail)
VALUES('00211','SCHMURZ GRAND_SPORT','396.80','521.95');
INSERT INTO pricelist(partno,partdesc,partprice,partretail)
VALUES('00216','SCHMURZ GRAND TOURISMO','194.98','256.95');
INSERT INTO pricelist(partno,partdesc,partprice,partretail)
VALUES('00218','MICHELIN M+S','187.15','245.95');

```

This is not the whole story, though, and does not fully show the power of XSLT.

In fact, in many places in my original XML file the description contains quotes: instead of, say, MICHELIN M+S, the file may contain something like MICHELIN 'M+S'. And the insert will fail because in PostgreSQL quotes are special characters needing to be escaped.

One way to solve this problem is to replace, inside a description, a single quote with a double quote. PostgreSQL is happy with this solution and I am too, as long as I am consistent through the whole database.

Now, I know how to do this search and replace in Clarion, but I am using XSLT here, and there is no way to include a little piece of Clarion code....

Did I say XSLT was a language? I did! So there should be some way to do a search and replace, right? In fact there are a number of ways, but I will demonstrate just one.

I will write a "kind of" recursive function which splits a description into what is *before* what I search, and what is *after*, so that I can replace as many single quotes as needed in a description; this code then calls itself for each *after* portion until no more single quotes are found.

First, I need to declare two variables, search and replace:

```
<xsl:variable name="search"><xsl:text>&apos;</xsl:text></xsl:variable>
<xsl:variable name="replace"><xsl:text>&quot;</xsl:text></xsl:variable>
```

The ' text is for the single quote ('), and " is for the double quote (").

Then, I need to write my "function" using an <xsl:template> tag:

```
<xsl:template name="replace-string">
</xsl:template>
```

Yes, I am writing a second template, inside my "stylesheet". This one does not "match" anything, but has a name, which will allow me to call it from inside the previous template. If you're familiar with the Clarion template language, you can think of this as being similar to a #GROUP statement

I need to be able to pass parameters, so I declare them like this:

```
<xsl:template name="replace-string">
  <!-- search for this: -->
  <xsl:param name="search" select="string(.)"/>
  <!-- and replace it with this: -->
  <xsl:param name="replace" select="string(.)"/>
  <!-- here is the original string: -->
  <xsl:param name="string" select="string(.)"/>
```

each parameter has a name and a select attribute that points to "self", or the value passed with the name.

Now, I need something like Clarion's CASE structure, or an IF/THEN/ELSE to handle multiple test conditions. In XSLT, xsl:choose does the job:

```
<xsl:choose>
  <xsl:when test="not(contains($string, $search))">
    <!-- if there are no more appearances of $search in the
    $string, output the rest of the string and stop. -->
    <xsl:value-of select="$string"/>
  </xsl:when>
```

The <xsl:otherwise> tag is an ELSE for when \$search is found in \$string:

```
<xsl:otherwise>
  <!-- output the part of the $string that is before the
  first appearance of $search. -->
  <xsl:value-of select="substring-before($string, $search)"/>
```

```

<!-- output the replacement $replace. -->
<xsl:value-of select="$replace"/>

```

Finally this template calls itself again, using the part of \$string that come *after* the first \$search:

```

<xsl:call-template name="replace-string">
  <xsl:with-param name="search" select="$search"/>
  <xsl:with-param name="replace" select="$replace"/>
  <xsl:with-param name="string" select="substring-after($string, $search)"/>
</xsl:call-template>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

```

That's the generic search and replace function. It receives three parameters: the original string, the search for string, and the replace with string. You can follow step by step in an XSLT debugger what is happening; the steps are as follows:

First call:

\$string = MICHELIN 'X+S'

\$search = '

\$replace = “

First Iteration Output= MICHELIN “

Second call:

\$string= MICHELIN “X+S'

\$search='

\$replace=”

Second Iteration Output=MICHELIN “X+S”

Third call:

\$string = MICHELIN “X+S”

\$search='

\$replace=”

Exit immediately with the value of \$string

Finally, I need to modify the main template to call the search and replace function:

```

<xsl:call-template name="replace-string">
  <xsl:with-param name="search" select="$search"/>
  <xsl:with-param name="replace" select="$replace"/>
  <xsl:with-param name="string" select="PartDesc" />

```



```
</xsl:call-template>
```

Here's the whole script so far:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions">
  <xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:variable name="new_line" select=""
"/>
  <xsl:variable name="search"><xsl:text>&apos;</xsl:text></xsl:variable>
  <xsl:variable name="replace"><xsl:text>"</xsl:text></xsl:variable>
  <xsl:template match="Extract">
  <xsl:for-each select="Item">
  <xsl:text>INSERT INTO pricelist(partno,partdesc,partprice,partretail) VALUES('</xsl:text>
  <xsl:value-of select="PartNo"/>
  <xsl:text>', '</xsl:text>
  <xsl:call-template name="replace-string">
    <xsl:with-param name="search" select="$search"/>
    <xsl:with-param name="replace" select="$replace"/>
    <xsl:with-param name="string" select="PartDesc" />
  </xsl:call-template>
  <xsl:text>', '</xsl:text>
  <xsl:value-of select="PartPrice"/>
  <xsl:text>', '</xsl:text>
  <xsl:value-of select="PartRetail"/>
  <xsl:text>');</xsl:text>
  <xsl:value-of select="$new_line" />
  </xsl:for-each>
  </xsl:template>

  <xsl:template name="replace-string">
  <!-- search for this: -->
  <xsl:param name="search" select="string(.)"/>
  <!-- and replace it with this: -->
```

```

<xsl:param name="replace" select="string(.)"/>
<!-- here is the original string: -->
<xsl:param name="string" select="string(.)"/>

<xsl:choose>
  <xsl:when test="not(contains($string, $search))">
    <!-- if there are no more appearances of $search in the
    $string, output the rest of the string and stop. -->
    <xsl:value-of select="$string"/>
  </xsl:when>
  <xsl:otherwise>
    <!-- output the part of the $string that is before the
    first appearance of $search. -->
    <xsl:value-of select="substring-before($string, $search)"/>

    <!-- output the replacement $replace. -->
    <xsl:value-of select="$replace"/>
    <!-- repeat the process, using the part of $string that
    comes after the first appearance of $search. -->
    <xsl:call-template name="replace-string">
      <xsl:with-param name="search" select="$search"/>
      <xsl:with-param name="replace" select="$replace"/>
      <xsl:with-param name="string" select="substring-after($string, $search)"/>
    </xsl:call-template>
  </xsl:otherwise>
</xsl:choose>
</xsl:template>
</xsl:stylesheet>

```

What about Clarion?

I have an XSLT script, and I know it works because I tested it inside EditiX with my XML file, and I get a nice SQL script as output, but this has nothing to do with Clarion, at least so far. I'll explain how to integrate this code into a Clarion app [next time](#).

[Bernard Grosperin](#) is a native of France, and has been a big fan of everything American since his teens. He began visiting the US in 1996, and moved to San Antonio in 1998, where he discovered his love

of Mexican food. He and his lovely wife Gloria, who he met in Tulsa, now live in California. Bernard has been programming and designing software and databases for 14 years, primarily with Clarion. His hobbies include flying radio-controlled airplanes and riding his motorcycle. He loves aircraft of all kinds, and can't miss an opportunity to fly, whether it's in a glider, a World War II trainer, or a general aviation Piper or Cessna.

Reader Comments

Posted on Friday, April 27, 2007 by Didier Le Duc

Nice work, I have been using XML since 1998 and have always been surprised that the Clarion community was not following the hype. XSLT is the next step for serious work and needed to be presented. Thank you Bernard

[Add a comment](#)

Clarion Magazine

Coping With Windows Vista - How to live with User Account Control Security

by Jane Fleming

Published 2007-04-13

This started out as an article on using manifests in Windows Vista. These manifests provide information to Vista's User Account Control (UAC) security system. Without a correctly configured manifest, applications will trigger Vista's security measures.

But perhaps like many of you, my first exposure to Vista's incessant popup "Are You Sure" window was **make it stop!!** And from chatter I see on various forums, the first thing a lot of people seem to want to learn how to do within Vista is to turn off the UAC security that's behind those popups.

Mark Minasi's really fantastic book [Administering Windows Vista Security: The Big Surprises](#) was instrumental in changing my mind. I've converted to the opinion that UAC is actually A Good Thing (although it wouldn't have hurt my feelings too much if Microsoft hadn't made it so much resemble a four year-old who can't stop saying, "Are we there yet?"). As I'll explain in this article, there are some techniques you can use to make life with UAC a whole lot more bearable. Many of these, ironically, involve going back to the command line interpreter. But that isn't necessarily a bad thing.

In the second of these articles, "There's A Manifest In Your Destiny", I'll go into more explanation as to what's happening when those nag screens pop up. Briefly, the problem occurs because Vista doesn't run most tasks with administrator privileges. I'd wager that if you're like most Clarion developers, most of the time you log onto your computers as an administrator. Unfortunately, that means if you inadvertently launch any kind of virus or malware, it also runs with administrator credentials. Vista, on the other hand, doesn't really run you as an administrator until after one of those nag screens has popped up, and then it does so only as long as the specific application you've approved runs.

As a loose analogy, UAC is a firewall for administrators. A firewall in its original, literal sense is a construction feature that contains and minimizes damage to a building. Computer firewalls are an extension of that concept. It's rather rash not to have a firewall on your computers nowadays.

In his book, Mark Minasi explains that you can't get completely around the elevation issue without turning off UAC, because the Explorer shell (Windows Explorer, not Internet Explorer) starts when you log on, and a process can't be elevated once it's started. And Windows explorer is the fundamental way you interact with Windows.

In my own experimenting, I found a trick to getting a fully-elevated shell for the duration of one logon, and e-mailed my explanation to Mark. He replied that after his book had gone to press, he'd also found a way to get a fully-elevated Explorer window.

For those of you who can't live without a mouse, I'll explain those two tricks at the end of this article. But first I'm going to encourage you to take a trip on the Wayback Machine... to the land of the command prompt (or DOS as some old folks still think of it.)

I'm one of those old folks. When I log onto a computer for the first time, one of the first things I do is to create a command prompt shortcut on the Quick Launch tool bar. Each morning, when I log onto my development machine I immediately open two command prompt windows. But I realize that a lot of people don't use the command prompt. In this article, I hope to change your mind and give you an idea of how useful a basic comfort level at the prompt can be, especially when it comes to dealing with UAC.

Power begets power

Once you've given Vista the authority to run a process at the elevated, full administrator level, that process can launch other processes. So if you launch a command prompt "as administrator", from that point onward anything you invoke from that prompt will also run elevated without any annoying popups.

Let's play with it a bit.

Open the Start menu. Click All Programs. Click Accessories. Hold the Ctrl key while you drag Command Prompt onto your desktop. Be sure you hold the Ctrl key while you do that. You're just copying the shortcut; you don't want to drag it off the menu altogether. Or you can right-click the Command Prompt menu item on the Accessories menu, then click Send To and click Desktop (create shortcut).

Right-click the shortcut on your desktop and rename it Admin. Now do the Ctrl-drag again (or Send To again) and make another copy of the Command Prompt shortcut. This will be your "regular" prompt.

Right-click the Admin shortcut and click Properties. Now click on the Colors tab. Click the dark blue square (Figure 1).

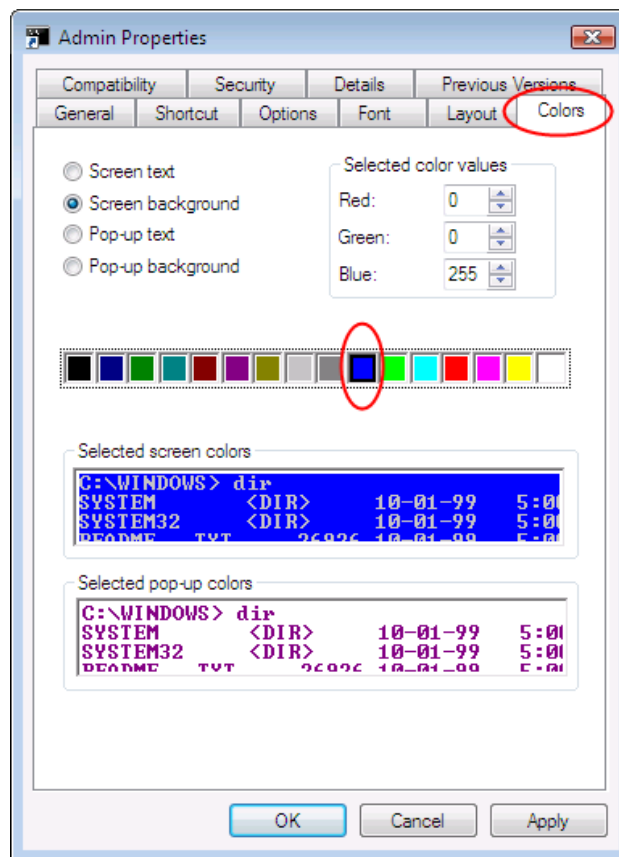


Figure 1. Set the color

Click the Shortcut tab. Click Advanced.... Click Run as administrator (Figure 2).. *Be sure* you're on the Shortcut tab, *not* the Compatibility tab.

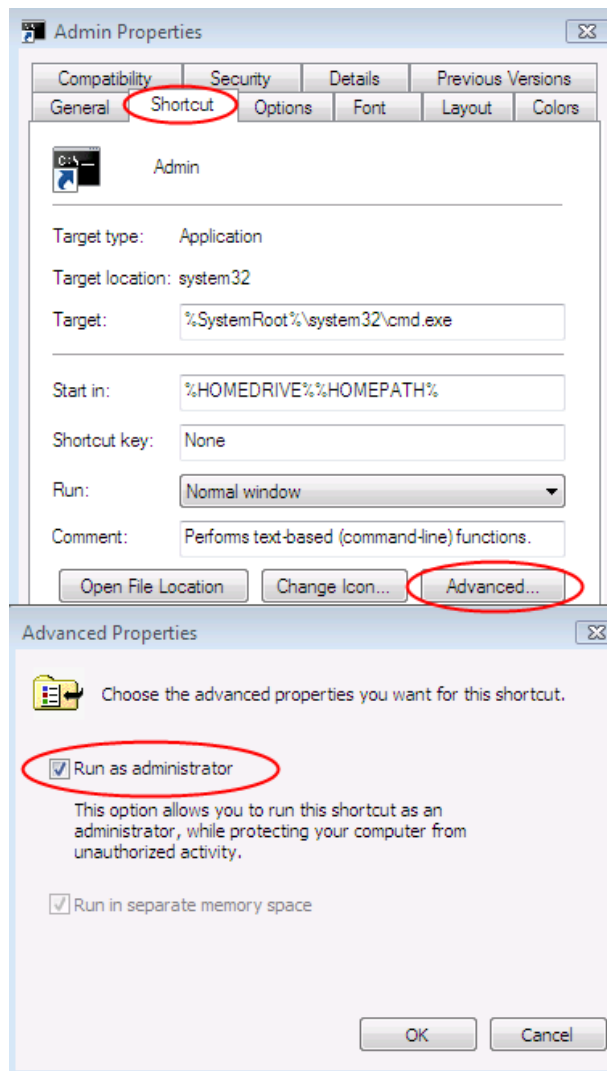


Figure 2. Specify Run as administrator on Shortcut properties

Set a different size font on the Font tab if you wish. Then click OK to get back to the desktop.

Now double-click your shortcut. Vista has put the word 'administrator' in the window title bar, but the blue color will be your most visible ongoing reminder that you've put aside your safety net (Figure 3).

I like to drag a copy of the admin shortcut down onto the Quick Launch section of the Task Bar so it's always handy.

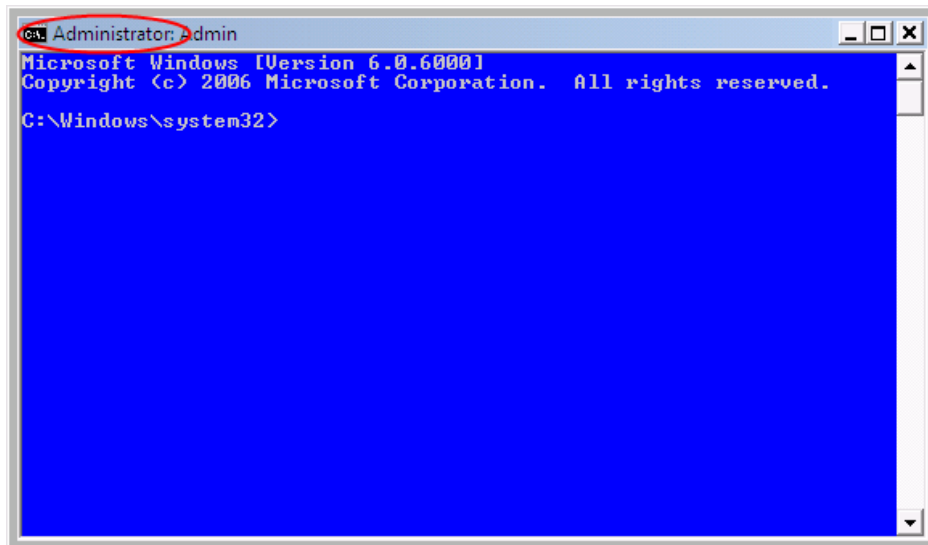


Figure 3. The elevated command prompt

OK. Let's try it out.

Type

Regedit [Enter]

Hey. No popup.

Type

Firewall.cpl [Enter]

With silent grace the firewall administration applet opens.

Type

sysdm.cpl [Enter]

Go ahead. Change your startup parameters or page file size.

Wanna format a disk or configure user accounts or folder shares or a bunch of other stuff??? How about

compmgmt.msc [Enter]

Hey, Buddy. Wanna configure an ODBC source?

odbcad32 [Enter]

Gets pretty addictive, doesn't it?

Your administrator command prompt has probably landed you right in the c:\Windows\System32 folder. As far as running these commands, it doesn't matter where you've navigated. But if you're in that folder, you can get a list of Control Panel applets by typing

dir *.cpl [Enter]

and a list of Management Console snapins by typing

```
dir *.msc [Enter]
```

Google the file names if you can't figure out what any of them are for.

For the truly ain't-got-a-life set, you can make your own [customized Management Console](#) with some of the items you use most, and run it from the prompt. I've included one such console in this article's download. You can create your own custom consoles for prior operating systems as well, but the one I'm including will only run on a Vista machine. To use it, copy it into the C:\Windows\System32 folder and then type

```
my.msc [Enter]
```

Or you could make a shortcut to my.msc. If you run it from a shortcut, you'll need to deal with the "are you sure" popup once, but then as long as you leave it open you have a lot of management control.

But Wait, There's More

Various online crib sheets are available with helpful command-line programs. [Here's one](#) listing more than 100, most of which are applicable to earlier Windows versions as well as to Vista.

A Command-line Primer

By the way, that's pronounced primm'-er, not like a first coat of paint.

Well, if you're with me this far, I'd like to review a few of the other basic commands that are useful when you're working at the prompt. Although I'll use lower-case commands, they are not case sensitive.

cd - change directory

The cd command moves your focus from one folder (directory) to another. The details depend on whether you're going deeper into a folder hierarchy or up.

If you're in C:\Program Files, for example, and you want to move to C:\Program Files\MyProgram, you just need to type the name of the subfolder followed by Enter.

```
cd MyProgramFolder [Enter]
```

If you're in C:\Program Files\MyProgramFolder and want to move to C:\Program Files\OtherProgram, it's slightly more complicated. You need to either specify the new folder absolutely (relative to the root of the drive) or relative to where you are now.

The root of the drive is represented by a backslash character (yes, the opposite of the slash that's used in web URLs). At any point, if you type

```
cd \ [Enter]
```

you'll wind up back at the root of the drive. So I could type cd \Program Files\OtherProgram [Enter] as this would tell the system the full path to the new folder.

Two dots .. represents backing up one level. If I'm in C:\Program Files\MyProgramFolder and type


```
cd .. [Enter]
```

I'll wind up at C:\Program Files. If I'm in C:\Program Files\MyProgramFolder and type

```
cd ..\OtherProgram [Enter]
```

I'll wind up in C:\Program Files\OtherProgram. The two dots told the system to go up one directory, then down to the OtherProgram folder. The drive letter plus colon command moves the focus from one disk to another.

You'd think cd would work for switching from one drive to another, but it doesn't. To switch from c:\Program Files\MyProgram to the D drive on your computer, just type

```
d: [Enter]
```

(that's d and then a colon and then Enter).

dir and wildcards

The dir command does a directory listing. There are many times when I find this much more useful and powerful than the GUI Windows Explorer listings, although at times the GUI listing is more convenient.

But first I'll take a moment to review wild cards, which can be used with many commands.

The star (*) represents any number of letters. The question mark (?) represents a single letter. The final dot in a file name splits the name in two, so a wild card before the dot doesn't imply a wild card after it.

To get a list of all DLLs in a folder, type

```
dir *.dll [Enter]
```

To get a list of all DLLs beginning with win, type

```
dir win*.dll [Enter]
```

To get a list of all DLLs where the second and third letters are in, type

```
dir ?in*.dll [Enter]
```

Now that you've got that down, you can specify other sorting options. Want a list of all DLLs in your release folder sorted by date?

```
dir *.dll /od [Enter]
```

The o stands for *order*, and there are several orders you can use and combine. I find the most useful are by date and by size, /od and /os respectively. To reverse the sort, insert a minus sign. dir /o-s [Enter] will show files largest-to-smallest.

How about a listing of icon files, newest to oldest?

```
dir *.ico /o-d [Enter]
```

If you want to see hidden files as well, use /a For example:

```
dir c:\pagefile.sys /a [Enter]
```

If you want to see 8-and-3 style short file names, use the /x switch:

```
dir *.dll /x [Enter]
```

If you want to see files from subdirectories also, use the /s switch:

```
dir c:\clarion\3rdparty\examples /s [Enter]
```

If you tried that last command and had a gazillion files streaming by, you have a couple of options. Use /p to pause after each page:

```
dir c:\clarion\3rdparty\examples /s /p [Enter]
```

To abort the listing, press Ctrl+C. Or redirect the listing to a text file:

```
dir c:\clarion\3rdparty\examples /s > myfile.txt [Enter]
```

will create a text file of the listing, rather than showing it on the screen. Type myfile.txt [Enter] to have it open in Notepad or whatever you've specified as your default text file editor.

md - make directory

The md command makes a directory (folder). If you use the \ backslash, the directory will be made in the root of the drive. Otherwise it will be created beneath the current folder.

```
md c:\test [Enter]
```

will make a directory called test in the root of the C drive.

If you're creating a directory with a space in its name, enclose the name in quotation marks:

```
md "c:\My Stuff" [Enter]
```

rd - remove directory

The rd command deletes a directory. If the directory has anything in it, rd will refuse to work unless you use the /s switch. Again, use quotation marks around the directory name if it contains any spaces.

```
rd c:\test /s [Enter]
```

If you don't want the "are you sure?" prompt, also use the /q (for quiet) switch.

del - delete files

The del command deletes one or more files. Wildcards are acceptable. Be careful!

xcopy - copying multiple files

For copying files, I rely heavily on xcopy. Vista also includes the robocopy command which was previously included with various Resource Kits. But xcopy is available on pretty much any Windows machine you'll encounter.

The xcopy switches I find most useful (in varying combinations) are /s /e /v /h /d /u /y

I always use the /v switch with xcopy or with copy. It tells the system to verify that the copied file was written correctly. I recall when I was teaching networking that students would try to drag a 100 megabyte service pack from their CDs onto their hard disks using the GUI and it sometimes didn't copy correctly. But they had no indication that the copy was corrupt until they tried to run the service pack. When you use the /v switch, you know.

/h tells xcopy to copy files even if they're hidden.

/y tells xcopy to overwrite an existing file with the same name. But there are ways to use this with a certain amount of precision.

/s means to copy files within any sub folders

/e means to copy the folder structure, including any empty folders.

/u means only to copy files that have the same name as files that already exist in the destination

/d means date. If you use /d without specifying a date, only newer files will be copied.

Here are some samples.

Copy everything (including hidden files) from my development folder and all its subfolders to a backup folder on an external hard drive that my system sees as M: and create a matching subfolder structure on the destination. (NOTE: if I've already navigated to somewhere on the D: drive, I wouldn't need to type the d: part of these command strings.)

```
xcopy d:\development\*.* m:\MyBackups /s /e /v /h [Enter]
```

Copy everything from my development folder to a backup folder on an external drive, but don't overwrite any file unless the source file is newer:

```
xcopy d:\development\*.* m:\MyBackups /s /e /v /h /d /y [Enter]
```

Or let's say I have a folder I'm using for a SetupBuilder project. I have all the files that my installer needs in that folder. But after I've been working, I want to copy anything more recent from my development folder into the installer setup folder.

```
xcopy d:\development\*.* e:\setupfolder /d /u /v /y [Enter]
```

(copy only files that already exist on the destination, only if they're newer, verify the copy, and don't ask for confirmation to overwrite.)

That also makes a great simple batch file to check for any newer Clarion or third party DLLS as well as my own updated modules, and update my setup folder. Here's a three line doup.bat for my machine:

```
xcopy d:\development\*.* e:\setupfolder /d /u /v /y
xcopy c:\clarion6\bin\*.* e:\setupfolder /d /u /v /y
xcopy c:\clarion6\3rdparty\bin\*.* e:\setupfolder /d /u /v /y
```

attrib - file attributes

attrib shows and optionally modifies a file's attributes. The command

```
attrib *.doc [Enter]
```

will show attribute for all files with the doc extension in the current folder. The command

```
attrib *.dll -s -r -h [Enter]
```

will remove the system, hidden, and read-only flags from any DLL files in the current folder that may have them set. A minus sign clears the attribute in question. A plus sign sets it: `attrib myfile.exe +r [Enter]`

Those are the commands I use very day. The reading list at the end of this article contains additional references.

Mapping A Drive

I find it much faster and more convenient to map drives from the prompt than from Windows Explorer. To map from the prompt you use the powerful and versatile net command. I'll also show you how you can use it to add a user account to your computer (or to a domain).

First, though, consider the matter of persistence of mappings. When you use the GUI to map a drive, there's a checkbox labeled Reconnect at logon. If that box is ticked, the mapping is considered persistent, otherwise not.

I prefer that my mappings not be persistent. To set that, type

```
net use /persistent:no [Enter]
```

To see what computers your computer is aware of, type

```
net view [Enter]
```

To see what shared items are available on a computer, type

```
net view \\computername [Enter]
```

(where computername is the name of the computer you want to view)

NOTE: Only visible shares will show in this listing. Any time you create a share, if you use a dollar sign as the last character of its name it will be hidden. Then a user needs to know the name and must type it exactly in order to map a drive, rather than browsing and selecting the share to map.

When you use net to map a drive you can specify the drive letter you want the system to use or let the system select the next drive letter automatically, starting from Z and working backward,

To map a drive and specify that I want to use drive X, I type

```
net use x: \\vista1\jane2 [Enter]
```

(where Vista1 is the name of the computer and jane2 is the name of a share on Vista1). To map a drive and let the operating system select a drive letter, I type

```
net use * \\vista1\jane2 [Enter]
```

To map a drive to a hidden share, I type

```
net use z: \\vista1\MySecretShare$ [Enter]
```

To get a list of my drive mappings, I type:

```
net use [Enter]
```

To remove the mapping I've created for the Y drive, I type

```
net use y: /d [Enter]
```

The Secret Administrative Shares

There's a twist to the hidden share thing that's quite handy to know. By default, Windows creates hidden "administrative" shares for the root of each drive. Their names are the drive letter followed by a dollar sign (which, as mentioned above, makes the share hidden).

So to map a drive to the root of the C drive of a computer on my network, I could type:

```
net use z: \\RemoteComputerName\c$ [Enter]
```

Imagine my dismay when I found that this doesn't work with Vista. At least, not out of the box. On computers that do not belong to a domain, Vista disables the administrative shares. It took some Googling, but I found the registry value that needs to be added.

NOTE: Make registry changes at your own risk. I suggest you back up the registry first.

From your elevated administrator command prompt, type

```
Regedit [Enter]
```

Navigate until you find `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System` and click Edit on the top menu. Select New, then DWORD (32-bit) Value. Name the new value `LocalAccountTokenFilterPolicy`

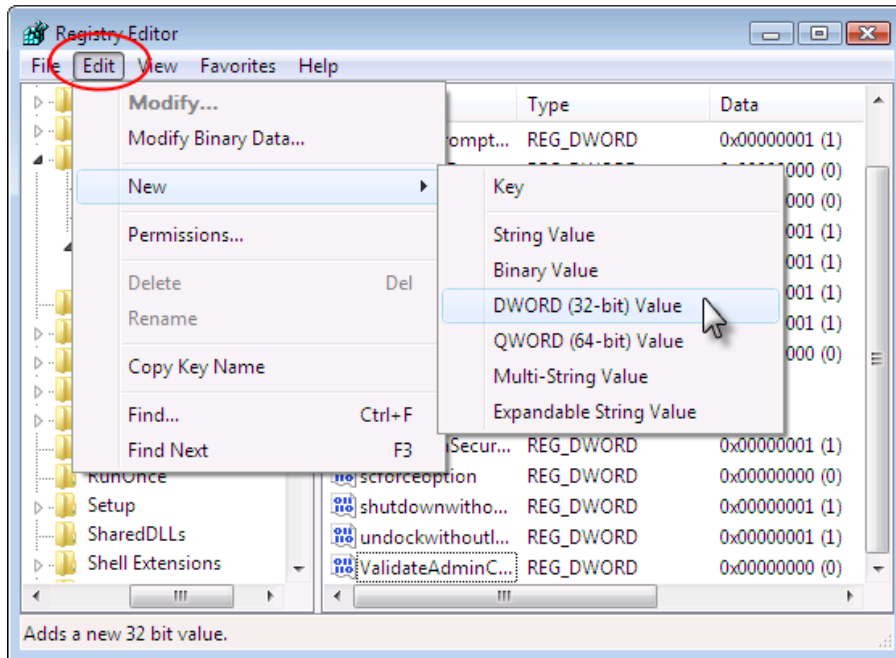


Figure 4. Add a DWORD value called LocalAccountTokenFilterPolicy

Double-click the new entry you created and set its value to 1 (Figure 5).

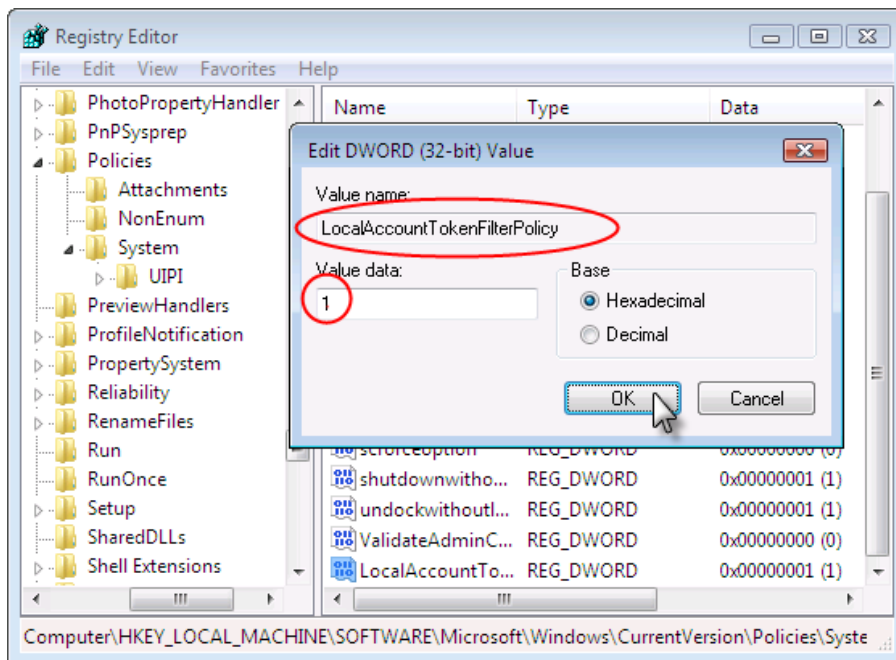


Figure 5. Set the value to 1 and then reboot

You'll need to reboot to see the changes.

One of the other neat things you can do with the hidden administrative shares is to sit at Computer A and create shares on Computer B. (Obviously, for any of this hidden administrative share stuff to work, you'll need administrator credentials on both machines.)

If you right-click My Computer and select Manage, you normally get the local computer Computer Management screen. If you then right-click the Computer Management (Local) bar, you have

the option of connecting to manage a different computer (Figure 6).

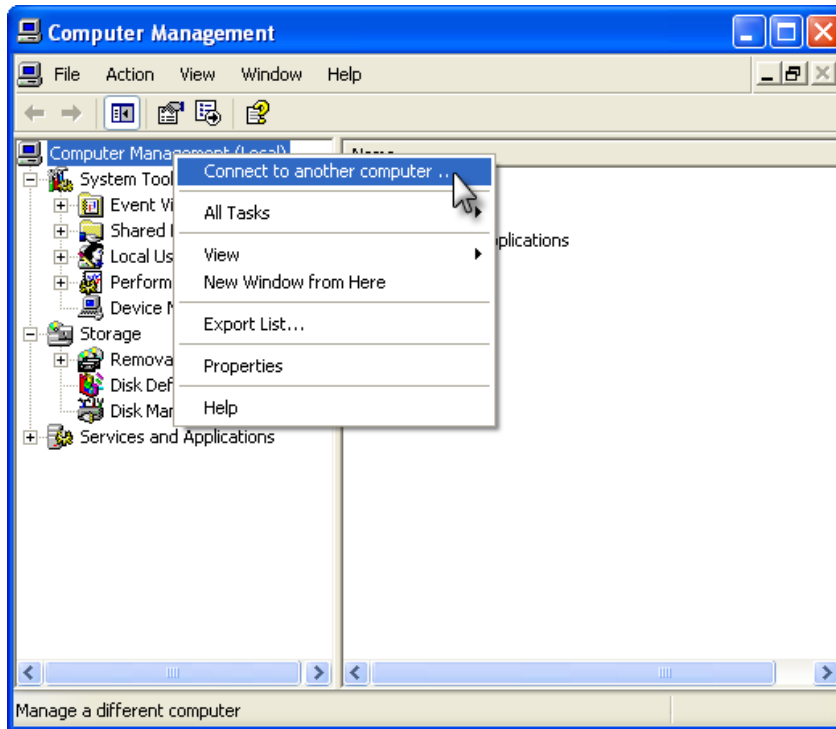


Figure 6. Connecting to manage a different computer

When I try to manage shares on the remote Vista computer without having modified its registry, though, I get an error (Figure 7).

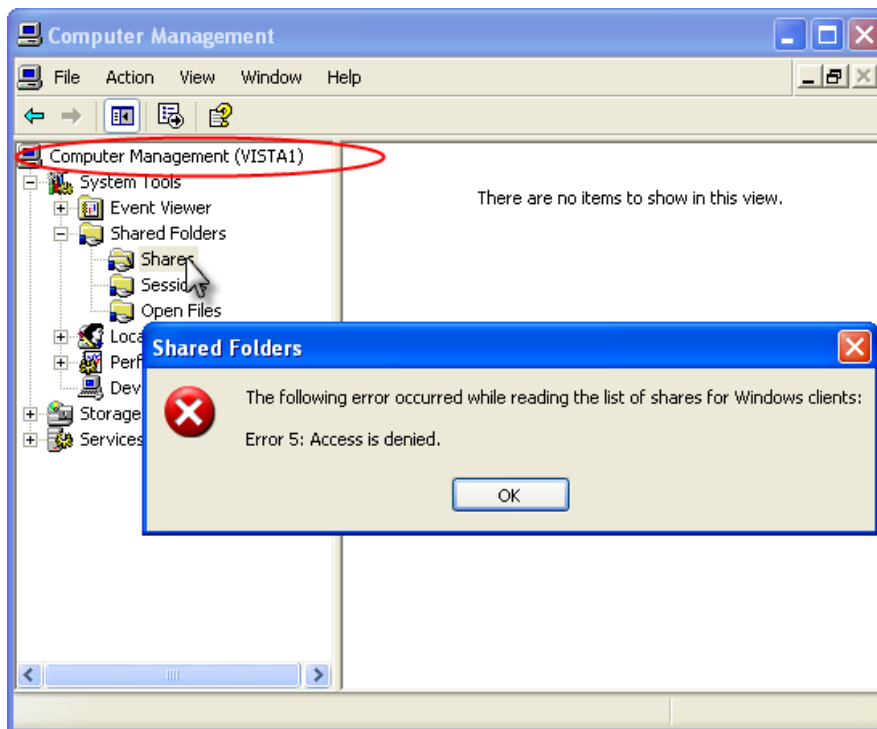


Figure 7. Access to remote share management is denied

After creating that registry entry and rebooting, however, I can see the shares on the remote computer

and create new ones (Figure 8).

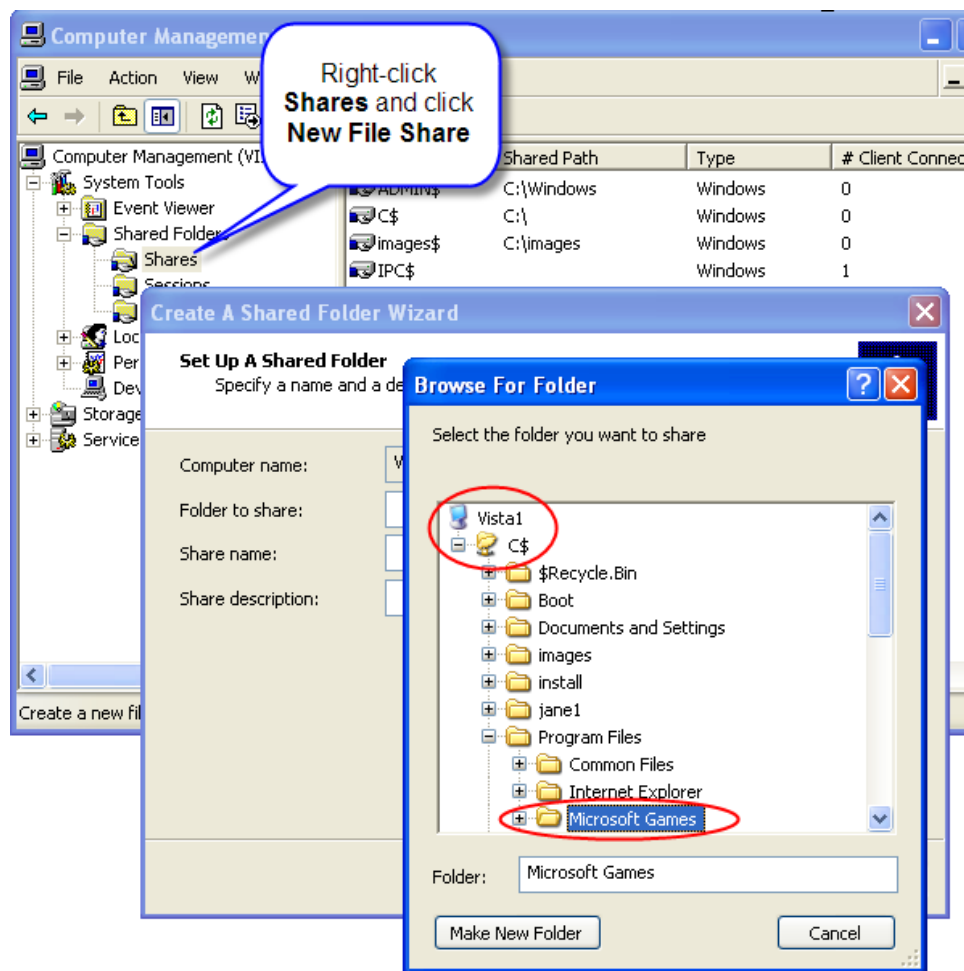


Figure 8. Use the hidden administrative C\$ share to create other shares remotely

More frequently, though, I'll map a local drive letter to the administrative share on the remote computer from the command line. Then I can access whatever I need on the remote drive without creating a bunch of shares. After having added this new value to the registry, that mapping now works fine (Figure 9).

```

C:\> net use z: \\vista1\c$
The command completed successfully.

C:\> net use
New connections will not be remembered.

Status          Local          Remote
-----
OK              Z:             \\vista1\c$
The command completed successfully.

C:\>

```

Figure 9. From an XP machine's command-line, mapping to a hidden administrative share on a Vista machine

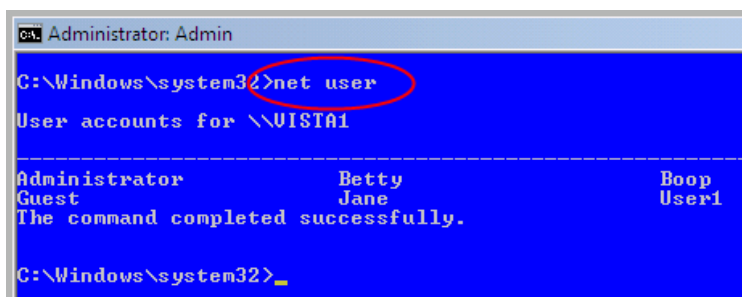
One of the other handy uses of net is for basic maintenance of user accounts. Type

```
net user help [Enter]
```

to get a list of options.

```
net user [Enter]
```

shows the user accounts currently on my computer (Figure 10).



```

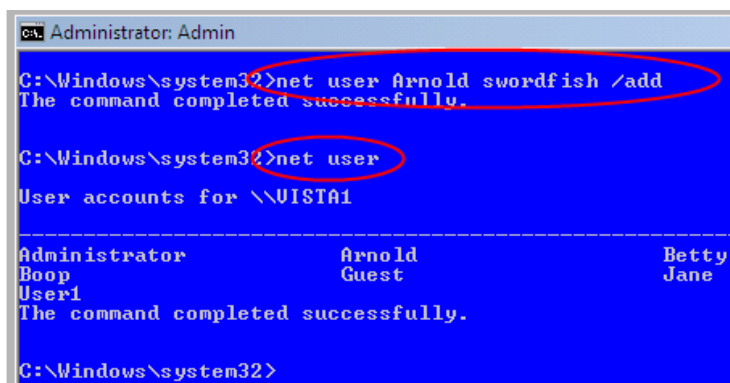
C:\Windows\system32>net user
User accounts for \\VISTA1
-----
Administrator      Betty              Boop
Guest               Jane              User1
The command completed successfully.

C:\Windows\system32>

```

Figure 10. Net use lists a computer's users

It's easy to add a user account (Figure 11). I'm only specifying the user's name and password, but there are many more details you can add. Someone posted a question on one of the news groups lately, by the way, about having SetupBuilder create a user account. I suggested he could use a simple batch file invoking net user, include the batch file as a support file in his SB project, and run it at the appropriate point in his script. .



```

C:\Windows\system32>net user Arnold swordfish /add
The command completed successfully.

C:\Windows\system32>net user
User accounts for \\VISTA1
-----
Administrator      Arnold            Betty
Boop                Guest            Jane
User1
The command completed successfully.

C:\Windows\system32>

```

Figure 11. Adding a new user account

In Figure 11 I added a new user account, then used net user [Enter] by itself to get a new listing. Of course, on Vista I need to be elevated to do so. Because this is the elevated prompt window I created, the command succeeds

And have you noticed the blissfully peaceful lack of "are you sure" screens through all this?

Go Really Retro with Command-line FTP

This almost goes in the "glutton for punishment" category. You can type

```
ftp ftp.somesite.com [Enter]
```

and get a client that epitomizes the essence of clunky. On rare occasions I've found it useful. Wouldya believe that LCD isn't a type of screen, but means "Change the Local Directory"? Pretty ugly...

It's A Clean Machine!

This trick I find very useful (on older machines as well as Vista). When you're packaging a program to distribute, you want to be sure you have all the pieces. While you can do dependency checks, the acid test is usually to install your software on a clean computer where there's no chance that it will be picking up some DLLs from a folder elsewhere.

But there's a trick with the prompt that you can do to simulate this.

You probably know that the system has a search path. If you type a command at the prompt (or in the Run box from the start menu), the system searches through the list of folders in its search path. The first program it finds matching what you've typed is what it will run. Similarly, it searches through its path for DLLs.

The Clarion redirection file is used by the Clarion linker and tools, but not by the operating system, so that's not part of this equation.

You can see your path on an XP machine by right-clicking My Computer, clicking Properties, then on the Advanced tab clicking Environment Variables. On a Vista machine, right-click Computer, click Properties, then click the Advanced system settings link, then Environment Variables.

Hah! "Are you sure" got you, didn't it! OK, from your elevated prompt type

```
sysdm.cpl [Enter]
```

and then select the Advanced tab and click the Environment Variables button. One of the variables you'll find is Path. However, you'll see it's a System variable. If you change it here, you change it for all user accounts and all windows on your computer. Don't want to do that! So close that window. Double-click the other command prompt shortcut you created – the one that is not running as an administrator. Type

```
path [Enter]
```

You'll see your path entries, each separated by a semicolon. The path on my XP machine looks like the top of Figure 12.

```

c:\>path
PATH=C:\WINDOWS\system32;C:\WINDOWS;d:\program files\textpa~1;C:\WINDOWS\System32\Wbem;C:\Program Files\ATI Technologies\ATI Control Panel;d:\cw6\bin;d:\cw6\3rd party\bin;e:\cw6\BIN;;E:\PROGRAM~1\Capesoft\Safe;E:\CW6\3rdParty\Bin;C:\MSSQL7\BIN;C:\Program Files\Microsoft SQL Server\90\Tools\bin\;C:\Program Files\ATI Technologies\ATI.ACE\;C:\Program Files\QuickTime\QTSystem\
c:\>notepad
c:\>path=.
c:\>path
PATH=.
c:\>notepad
'notepad' is not recognized as an internal or external command,
operable program or batch file.
c:\>

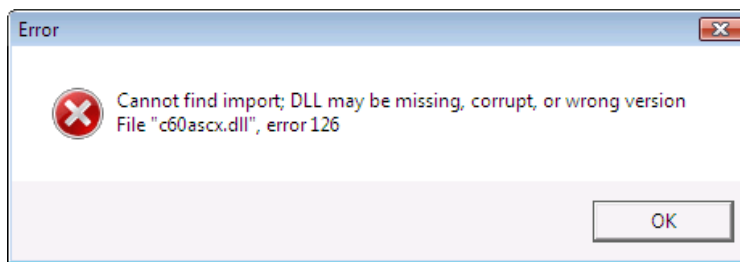
```

Figure 12. Showing the search path and clearing the path ([view full size image](#))

After I typed `path` [Enter] and got a listing of my search path, I typed `notepad` [Enter]. Because Notepad is in `C:\Windows\System32` the operating system found it and ran it.

Then I typed `path=.` [Enter] If you recall, I said that the operating system interprets two dots (`..`) as being one folder level up (which we used with `cd ..`). The interpreter sees a single dot as meaning "the current folder". So my command tells the operating system that the full width and breadth of its search path will henceforth be limited to this folder. Hence, the `notepad` command failed when I tried to run it after clearing the path.

But the neat thing is that this *just affects this particular command window*. The rest of my system works normally. Any other command prompt windows I may have open work normally. Now it's easy to try running my program from that de-pathed prompt and adding DLLs to the setup folder until everything is happy (Figure 13).

**Figure 13. Finding needed DLLs**

By the way, here's another trick with the single dot. On any Windows machine, if you're at a prompt and you've navigated some place where a GUI window would be helpful, type

```
explorer . [Enter]
```

The dot tells Windows Explorer to open focused on *this* folder.

Wanna be ordinary?

If your program is going to be run in an environment where users won't be logged on as administrators, you'll want to see how it behaves when running from an ordinary user account. While I'd suggest you actually log on as such a user to do final tests, you can simulate an ordinary user's experience with the `runas` command.

From Windows 2000 onward there has been an option to run a program as a different user. Right-click on a shortcut (or Shift+right-click in some cases) and that's one of the options on the popup menu. Not so with Vista. The only run-as option you have is Run as administrator.

But you can still run as a different user account from the command prompt. There are a number of options; I'm just going to illustrate the most basic.

In Figure 14, I'm trying to run Notepad using the credentials of an account named Betty. The system prompts me for Betty's password, then opens Notepad. (If I were using any parameters with my command, such as `notepad mytextfile.txt`, I'd enclose the whole thing in quotation marks: `runas /user:betty "notepad mytextfile.txt"`.)

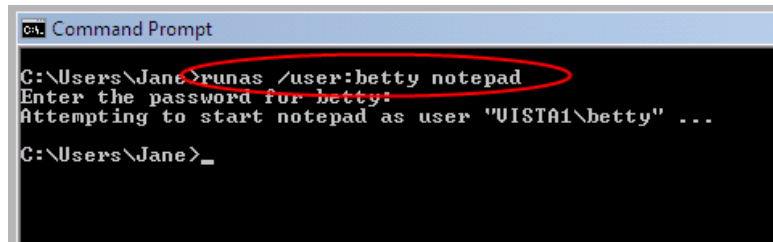


Figure 14. Starting a program as a different user

Then within Notepad, when I save a file you can see that it is respecting Betty's profile (Figure 15).

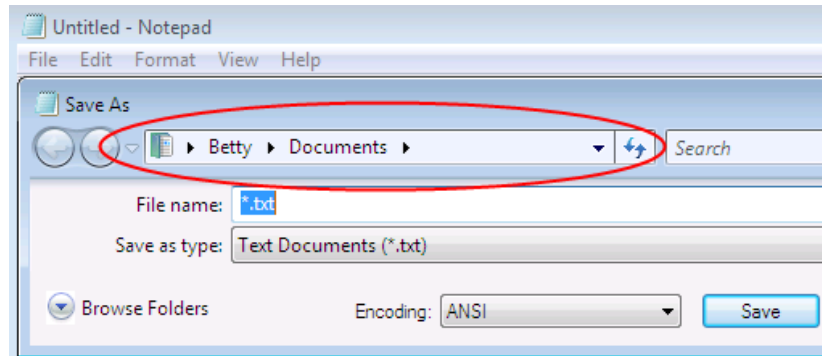


Figure 15. File open/save default to the run-as user's profile settings

Speaking of the Run command

You've probably noticed that Run is no longer one of the options when you click the Start menu button.

You can restore it by right-clicking the Task Bar, clicking Properties, selecting the Start Menu tab, clicking Customize..., scrolling about two thirds down the list, and ticking the Run command check box. I personally don't bother, as I do so much from the prompt. But one place the Run box is particularly handy is to open a remote computer's shares.

Assuming adequate permissions, from the Run box I could type \\Vista1 [Enter] and see a list of visible shares on that machine, any of which I could then right-click and map. Or better yet, type \\Vista1\c\$ [Enter] and have GUI access to all the folders on the remote machine's C drive (assuming I've added the registry value I described earlier.)

So, two quick tricks for a machine that doesn't have the Run box enabled.

1. Click Start. Then in the *Start Search* box at the bottom, type Run and hit [Enter].
2. Hold the Windows key (the one with the Windows wavy logo thing on it) and hit R

(Two other indispensable Windows key shortcuts are Windows+D to minimize everything and show your desktop and Windows+E to open Windows Explorer.)

But I really want my GUI!

All right. I said we'd see a couple of higher-powered GUI strategies that are better than turning off UAC altogether.

In Mark Minasi's book, he explains how a process once launched cannot be elevated. You'll experiment with that in the next article in this series when you try to run an elevated application from an ordinary one using Clarion's RUN and CHAIN commands.

If you open Task Manager, select the Processes tab, tick the Show processes from all users box, and sort alphabetically by Image Name, you'll see one instance of explorer.exe. That's the Windows shell, which is the interface between the user (i.e., you) and the Windows machinery. It starts when you log on, so it does not start elevated.

The Shell Game

Open Start | All Programs | Accessories, hold Ctrl and drag Windows Explorer to make a desktop shortcut (or right-click it, click Send To and then Desktop (create shortcut)).

Now right-click your shortcut and select Run as administrator. UAC will do its "are you sure?" thing, and you'll say Yes. Scroll down the left panel and open up Control Panel. Click Administrative Tools (this will only be visible if you've enabled Show hidden files and folders), then in the right pane double-click Services. Bam! "Are you sure?" strikes again! Or navigate to the C:\Windows\System32 folder. Put your mouse over some white space in the right hand pane, right-click, and click New. There's no option to create anything other than a folder.

When I was running from an elevated command prompt, I really *was* elevated and could invoke those types of utilities without having to answer the elevation window every time. Now, even though you theoretically started Windows Explorer running elevated, you actually didn't.

When you right-click My Computer and click Explore, the shell is actually just opening a new Explorer window. With the Windows Explorer window open, re-sort the processes in Task Manager. You'll still see only one instance of explorer.exe.

All For The Moment

From time to time I've encountered situations where I needed to kill the shell altogether in order to delete some files it's holding onto. After which I restarted it. So I reasoned this might be a way to get an elevated shell. And it is. If you want to try, follow these steps exactly.

1. Close all programs.
2. Open an elevated command prompt.
3. Open Task Manager by right-clicking the Task Bar or by typing taskmgr [Enter] at your elevated prompt.
4. Select the Processes tab in Task Manager. Click on any instance of explorer.exe. Click the End Process button.
5. Then kill Task Manager

Hmmm... Blank screen. No buttons. No taskbar. In your elevated command prompt window, type

```
explorer [Enter]
```

If you hadn't killed the shell, that command would have opened a new window (as it did earlier with explorer . [Enter].) In that you've killed the shell, the command started it from scratch. Since the prompt was elevated, Explorer now inherits its elevation. From now until you log off, Windows won't question your actions. Period. So be careful!

One Window

As I mentioned, Mark Minasi didn't present a means of using an elevated GUI interface in his book (without disabling UAC), so I e-mailed him my solution. He replied that he'd discovered a different solution after the book went to press. Here's Mark's.

Be sure you log off and log on again if you've done the previous demonstration, so Windows Explorer is again running at its default level. Open Windows Explorer. You're going to need to use the menu bar, so if it's not visible, click the Organize button, then click Layout, then click Menu Bar (Figure 16).

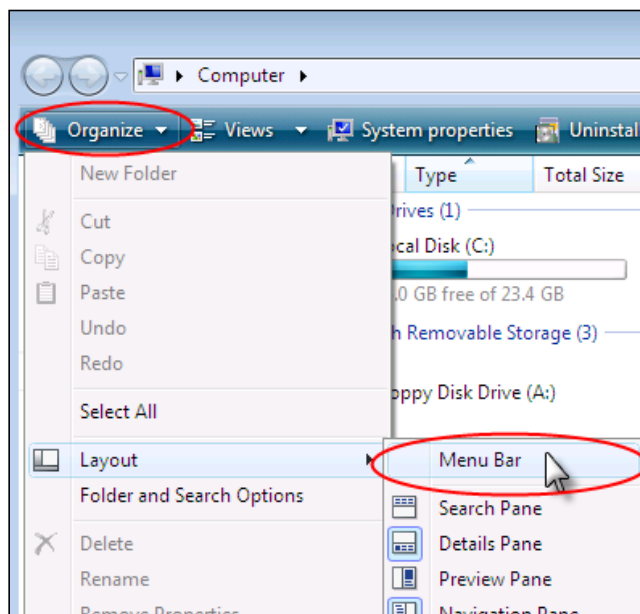


Figure 16. Enabling the Windows Explorer menu bar

Now click Tools, then Folder Options. On the View tab, scroll about half way down and tick the Launch folder windows in a separate process box (Figure 17). (You can also get to Folder Options from Control Panel or simply by clicking Folder and Search Options on the Organize menu itself.). As long as you're there, also select Show hidden files and folders and clear the Hide extensions for known file types boxes.

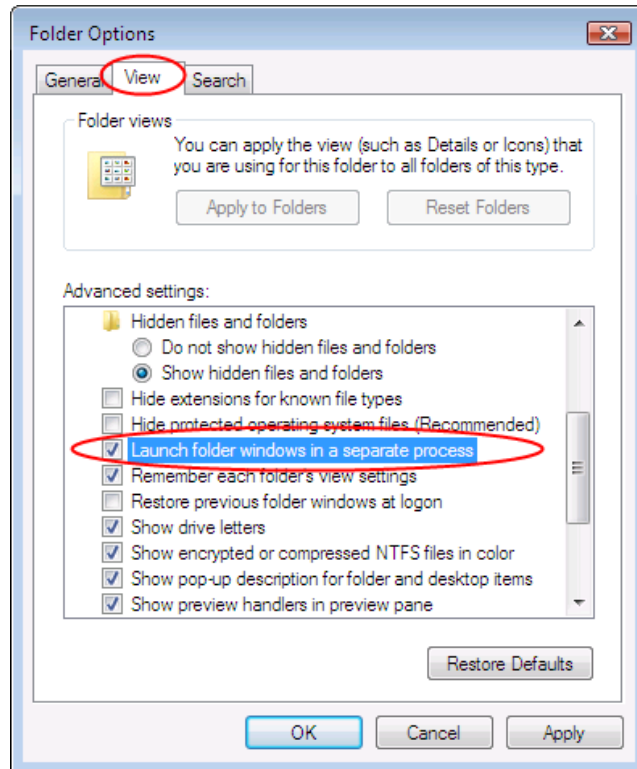


Figure 17. Launch folder windows in a separate process

Click OK. Now, if you right-click your Windows Explorer shortcut and click Run as administrator, you really are. Right-click within C:\Windows\System32 now and select New, and all the options are there, not just Folder.

You can also create an elevated Windows Explorer shortcut by modifying your Windows Explorer desktop shortcut as shown in Figure 2 for the command prompt shortcut.

What About *Him*?

OK. Here's one last option that doesn't involve turning off UAC. The default Administrator account is by default *not* subject to UAC. (That can be changed in a machine's security policy settings). But by default, the local Administrator account is disabled.

The built-in Administrator account has been a hacker target for years. It has a "well-known" Security Identifier (SID. I'll talk more about SIDs in the next article). But if you really want to enable the account anyway, do the following:

1. Right-click Computer and click Manage.
2. Click to expand Local Users and Groups
3. Click on Users
4. Right-click the account named Administrator and click Properties.
5. Clear the Account is disabled check box.
6. Click OK
7. Right-click the account named Administrator and click Set Password. Set it to whatever password you want the account to use.

8. Click through all the OKs.
9. Log off, then log on as Administrator.

Conclusion

It does strike my weird sense of humor that Microsoft's newest operating system may drive some people to becoming comfortable with the command prompt.

UAC is annoying, to be sure. But Microsoft has been the butt of security scorn and jokes for many years, and I'm glad they're really putting resources into trying to address security.

Beyond our own convenience or annoyance, there's the consideration of our users. Whether as software developers or computer gurus, we'll be dealing with many people who are living in the environment Microsoft has designed. I think it's irresponsible to join the chorus of people who are telling everybody just to "turn off that annoying UAC."

I hope some of the strategies I've presented may make living with UAC a bit easier.

In the [next article](#) in this series I'll talk about manifests and how our applications communicate with Vista's UAC. And after that, the magic mushrooms of Vista's virtual reality....

Additional reading:

- [Administering Windows Vista Security: The Big Surprises \(Mark Minasi\)](#)
- [New command-line tools in Vista \(Microsoft document\)](#)
- [Command-line A to Z \(as of XP\) \(Microsoft document\)](#)
- [Another list of command-line commands](#)

[Download the source](#)

[Jane Fleming](#) is a college dropout who subsequently lived four years in Europe, a year and a half in Mexico, and three years in India, and later taught yoga for a living in California (she's been vegetarian since 1970). She developed circuits and wrote assembly code for several embedded microcontroller projects during the 1980s. She began using Clarion Professional Developer for in-house projects back when Clarion was running display ads in InfoWorld and has used it very intermittently since. She is a former Microsoft Certified Trainer and taught Microsoft and Novell network administration at a business college for four years. Now widowed ten years, Jane plays [classical piano](#) and has found her *métier* as a semi-retired NRA-certified pistol instructor.



Reader Comments

Posted on Tuesday, April 17, 2007 by Sean Cameron

Microsoft's implementation of UAC, along with all their other security implementations, are completely useless for one reason:

They ask you the same question more than once.

To illustrate this point the other day I saw Bruce run a shortcut to an application that ran from a shared directory on a server, and of course it popped open the security warning dialog and Bruce clicked through it immediately. I asked him whether he wanted to turn it off and his reply demonstrated why this approach to security is so completely pointless - he didn't even see the dialog box, he always clicks the same button on it without thinking, because otherwise it just gets in the way, and there is no way you can ask a human being to click the same button thousands of times and then expect them to pick a different option the one time they really should. It's actually completely goes against the way that humans learn, and physically it is difficult not to push the same button after repeating the same action so many times, it is an ingrained neural pattern.

And that's with a programmer, with a non technical person it is far, far worse. There is no way my grandmother will ever be able to discern between the allow and disallow options. Even worse, if I have allowed a specific program to perform a specific action, why would I not want that program to always be allowed to perform the action?

Your workaround does disable UAC in effect, it's no different from turning it off - once you have elevated privileges for a process UAC no longer applies. It's using the administrator token, exactly the same as it would be if UAC was off.

Microsoft seriously need implement a trust list that is automatic - allow the user to trust an application and then don't continuously prompt for the same action every single time. There is no way I will turn on UAC and I don't know of anyone else who uses UAC either, which completely invalidates their attempt at security. Besides Trend Internet Security does the same thing, only far, far better,

Posted on Tuesday, April 17, 2007 by Paul Rubiola

Jane,

Thanks for an insight into the workings of the Vista security center.

I'm an old DOS user and I too have remained in touch with my inner nerd. Here is a little command prompt trick I was forced to learn.

I still have some Clarion-Dos programs in use and with the advent of USB printers there was gloom on the faces of many a user.

I found that if I shared the printer, regardless of the fact that there was no network, that I could then do the following command prompt fix:

```
net use LPT1 \ComputerNameMySharedPrinterName /Persistent:Yes
```

and all was right with the world again.

As this is now mostly a GUI world, the need for such tricks is diminishing. Still it is helpful to remember the basics. You never know when Civilization 2.0 will drop back to StoneAge.1

Paul Rubiola

[Add a comment](#)

Clarion Magazine

Sending User Messages Between Processes

by Larry Sand

Published 2007-04-12

In the [first installment](#) in this series I focused on establishing a link between two processes, and I demonstrated a class I've written to manage interprocess communication. If you created a couple of applications that implemented this class, you wouldn't yet have anything particularly interesting. These apps would negotiate the link and nothing more, but the good stuff happens when you start moving user messages between processes

I noted [last time](#) that it's unsafe to directly pass messages in the WM_USER and WM_APP range between processes. However, there's no such restriction when they are wrapped in a registered message and sent to another process using the method described for the link messages. To facilitate this, the class implements four methods that are used to register, send and receive the user defined messages.

This framework defines the user defined message as:

- hWnd - The window handle of the window that receives the message
- uMsg, Self.UserMessage - the return value from the RegisterWindowMessage for the GUID received in the RegisterUserMessage method.
- wParam - Message in the 0400h to 0BFFFh range to send to the other process.
- lParam - User defined value
- Returns zero

As with the link message, the user defined message wrapper must call RegisterWindowMessage to receive a message identifier that's safe to pass between processes. To do this the class implements a RegisterUserMessage method.

```
InterprocessComs.RegisterUserMessage Procedure(String sUserMessageId)
szUserMessageId &CString
bResult      BOOL,Auto
Code
bResult = False
szUserMessageId &= New CString(Len(Clip(sUserMessageId)) +1)
If szUserMessageId &= Null
    Self.TakeError(IPC_ERROR_OUTOFMEMORY, 'Not enough memory')
Return bResult
End
```

Larry
Sand
is
an

```

szUserMessageId = Clip(sUserMessageId)
Self.UserMessage = CMAG_RegisterWindowMessage(szUserMessageId)
If self.UserMessage = 0
    Self.TakeError(CMAG_GetLastError(),'Register user message')
Else
    bResult = True
End
Dispose(szUserMessageId)
Return bResult

```

You'll notice that this method is very similar to the Init method. It accepts a string parameter which is a registry style GUID used to register the message with Windows. Because RegisterWindowMessage accepts a CString as its string parameter, a CString reference variable declared and memory is allocated with New. The GUID string is assigned to the CString and then passed to the Windows RegisterWindowMessage function. The returned message identifier is stored in the UserMessage property.

In the same way that the window procedure checks the value of uMsg against the LinkMessage property it also tests to see if uMsg equals the value stored in the UserMessage property. When they are equal and the property isn't zero the wParam and lParam parameters are forwarded to the OnUserMessage method. OnUserMessage POSTs the event to an ACCEPT loop where it can be processed by your embedded code.

```

InterprocessComs.OnUserMessage Procedure(UNSIGNED nUserMsg, Long lParam)!, Virtual
Code
Post(nUserMsg, , Self.nThreadNumber)
Return

```

The nThreadNumber property is initialized to zero in the constructor, and posting to a thread number zero sends the event to the current Target. To post the message to another thread you'll need to call the ThreadNumber method prior to receiving the message. There are two overloaded ThreadNumber methods, one to set the property and another to get the property value. You should derive the OnUsageMessage virtual method in your programs if you want to access the value of lParam.

The base class implementation posts the user message value to the window's ACCEPT loop. You can then test for this using EVENT() as you normally would. When using Clarion 6 and you're posting an event to another thread and you want to send a 32 bit value, you need to use NOTIFY/NOTIFICATION. I'll present an example of how to do this shortly.

To send the user message to your partner process, call the SendUserMessage method with your user message value and the optional Long integer passed in lParam.

```

InterprocessComs.SendUserMessage Procedure(UNSIGNED nUserMsg, Long lParam=0)
Code
If Self.hWndPartner And Self.UserMessage
    CMAG_PostMessage(Self.hWndPartner, Self.UserMessage, nUserMsg, lParam)
End

```

Return

Before you call `SendMessage` you must register the user message with `RegisterUserMessage`. If no message was previously registered, nothing will be sent.

Testing

It's time to look at a very simple hand coded example program that brings together all of the concepts discussed to this point. The example is presented as a hand coded project and `.clw` file because it allows you to see all of the source code at once without needing to dig through embed points in an application.

The source for this program is in `ipcA.clw` and `ipcA.prj` in the downloadable zip at the end of this article. You will also need the two class files `ipcACL.inc` and `ipcACL.clw`.

Program

Include('ipcACL.inc'),Once

Map

End

```
W WINDOW('Interprocess communication simple example'),AT(,300,200),SYSTEM,GRAY
  BUTTON('Send User message'),AT(30,33,83,14),USE(?SendMessage)
END
```

ipc InterprocessComs

aUserMessage Equate(CMAG_WM_APP + 100)

Code

Open(W)

If ipc.init(W, 'F6CF22EB-0C7F-4fb2-AC5A-3636313440F2')

ipc.RegisterUserMessage('997C8819-A006-440d-8FCC-29C298FABC72')

End

Accept

Case Field()

Of ?SendMessage

If EVENT() = EVENT:Accepted

ipc.SendMessage(aUserMessage, 0)

End

End

Case EVENT()

Of aUserMessage

Message('Received user message')

End

End

Close(W)

Return

Compile the ipcA.prj project and start two instances of this program. Click on the Send User Message button on one of the instances and you'll see the message box pop up on the other instance. You now have a basic framework to send a user defined event (message) between two process without the danger of conflicting with other applications that may accidentally process your message.

Here are the steps to add the Interprocess communication class to your own program and then use it to send a simple notification message to another process.

1. Add Include('ipcACl.inc'),Once globally
2. Instantiate the class in the data section of your top level window, use any label you wish, here it is declared as: ipc InterprocessComs
3. Declare a user defined message constant. MSDN suggests that you use constants in the WM_APP message range. aUserMessage Equate(CMAG_WM_APP + 100)
4. After the window is opened call the Init method, passing it the label of the window and your GUID: ipc.Init(W, 'F6CF22EB-0C7F-4fb2-AC5A-3636313440F2')
5. If the ipc object was successfully initialized, register the user message by passing it another GUID: ipc.RegisterUserMessage('997C8819-A006-440d-8FCC-29C298FABC72')
6. In the button's Accepted event, send the user message by calling the SendUserMessage method, passing the message constant declared in step 3 for the nUserMsg parameter and zero for the optional IParam (not used): ipc.SendUserMessage(aUserMessage, 0)
7. Catch the message using Clarion's EVENT function. The example simply displays a message box.

As you may recall from the earlier discussion, the OnUserMessage method is virtual and the base class implementation simply sends the user defined event (message) to the Clarion ACCEPT loop with the Clarion function POST. Clarion 6 introduced a new set of notification functions that allow you to send an event (message) to the active window on another thread and include an optional long parameter. Sounds familiar doesn't it?

Next time I'll discuss how you can use NOTIFY and NOTIFICATION in Clarion 6 to send the message and IParam from OnUserMessage to the Clarion ACCEPT loop.

[Download the source](#)

independent software developer who began programming with Clarion in 1987. In addition to normal database development, he specializes in connecting Clarion to external devices like SCUBA diving computers, kilns, and satellite transceivers used in medical helicopters. In other lives, he sailed Lake Superior as the owner/operator of shipwreck SCUBA diving tours and later as a Master for the Vista Fleet. When Larry is not programming you'll find him messing about in boats, or with boats.

Reader Comments

[Add a comment](#)

Clarion Magazine

The ClarionMag Blog

Get automatic notification of new items! [RSS feeds](#) are available for:

XML [All blog entries](#)

XML [All new items, including blogs](#)

Blog Categories

- o » [All Blog Entries](#)
- o » [Clarion 7 Clarion.NET](#)
- o » [Future Articles](#)
- o » [News flashes](#)
- o » [Nifty Stuff](#)

ClearType in C7 (examples)

Direct link

Posted Monday, April 23, 2007 by Dave Harms

[Lee White](#) has generously provided a couple of images to demonstrate ClearType font smoothing in the latest C7 alpha build. ClearType is a technology that improves font readability on LCD monitors.

Fonts used in the example include Tahoma, Arial, Verdana, Arial (Black and Narrow), MS Sans Serif, and, because Andrew Guidroz just had to ask for it, WingDings. But the number of font examples is pretty much irrelevant. ClearType support is there thanks to SoftVelocity adopting the Uniscribe API, so this isn't a question of tweaking individual fonts, but of using an API that supports ClearType. The use of Uniscribe should also mean that C7 apps have support for multi-byte character sets, although I don't know of anyone who has tested this yet.

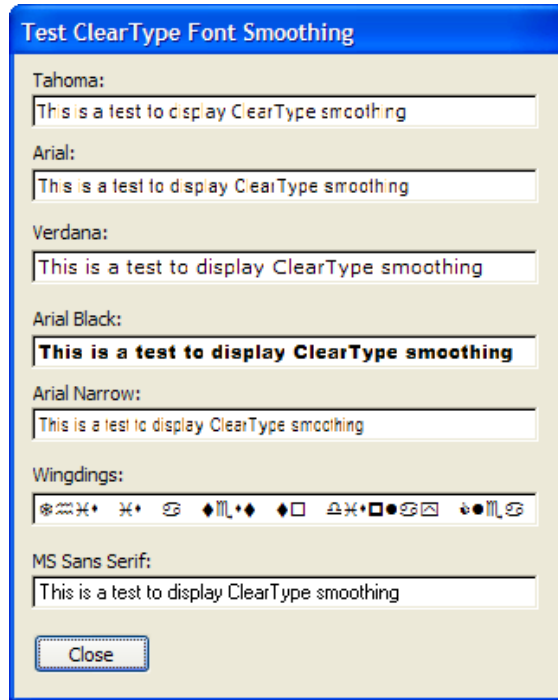


Figure 1. ClearType on C6

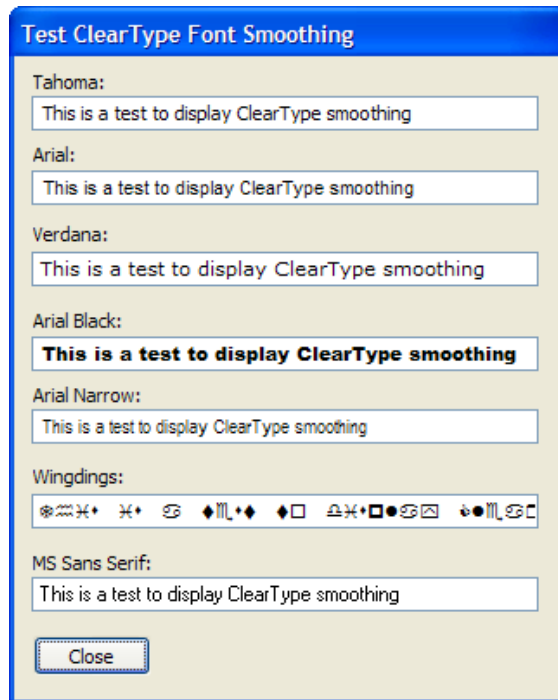


Figure 2. ClearType on C7

If your ClearType-ready app is deployed on a system that doesn't support Uniscribe, the runtime library falls back on the old method of displaying text.

ClearType, Unicode now in C7 Alpha

Direct link

Posted Thursday, April 12, 2007 by Dave Harms

The latest release of C7 Alpha 1 includes support for ClearType and Unicode. ClearType means that all fonts will correctly render on LCD monitors with ClearType turned on (without ClearType support, some letters get bunched together). Unicode support (via Uniscribe) means that Clarion apps will now have support for double-byte characters in entry fields as well as text fields.

What about older systems that don't have Uniscribe support? From the [SV blog](#):

The core Unicode support was implemented using the Uniscribe API. Although natively supported on Windows NT 5.0, the Uniscribe DLL may also be distributed for use on Windows NT 4.0, Windows 95, and Windows 98-based systems. But because this DLL may not be available on older systems we also implemented fallback code, so in the event that USP10.Dll cannot be loaded the RTL uses the GetCharacterPlacement function. According to the GetCharacterPlacement function's description it can have problems with caret placement on old Windows versions for certain languages, but it's good enough in most cases and it's available for any 32 bit Windows version. Having support for Unicode implemented with the fallback code was extra work, but it means we don't have to apply any additional requirements to which systems which can run Clarion programs.

The build with Unicode support looks like the final build for Alpha 1. We're all looking forward to Alpha 2 and the dictionary editor.
