Planning Corporate Deployment With Side-by-Side

Why use side-by-side

The primary benefit of deploying applications using side-by-side technology is application stability. Once the application works, it will keep working regardless of what other applications are installed or uninstalled on the machine. This is a direct result of application isolation. Each application has its own set of COM components and other DLLs and does not depend on installation or un-installation of other applications.

From the above we get lower packaging costs, lower testing costs and lower maintenance costs.

On top of that you can finally have conflicting versions of the same application on the same machine at the same time. These applications may not be able to run at the same time, but they would be able to run alternately. This depends on whether there are unresolvable resource conflicts. For example older versions of Microsoft MDAC use the same, named, memory mapped file for different purposes. They can not run at the same time, because they would corrupt each other's runtime data. Current versions of MDAC do not have that problem.

What about .Net

There are three aspects of side-by-side in the .Net context:

.Net (CLR managed) application loading .Net modules

Microsoft .Net environment provides its own side-by-side mechanism. It is similar to win32 side-by-side and the two were designed to coexist and cooperate. Manifest Maker does not have any facilities that influence .Net loading of .Net modules.

.Net (CLR managed) application loading Win32 (COM or other DLL) modules

To load a Win32 DLL or to access a COM resource (for example COM class) CLR must get out of its managed environment and make a call to Win32 API functions. This call is subject to standard Win32 processing rules including side-by-side. To make a .Net (CLR managed) application use COM resources just make a normal Win32 manifest for the executable as you would for a non-CLR application.

Win32 (non-managed) application accessing .Net (CLR managed) class through COM

CLR implements a seamless interface allowing Win32 (non CLR managed) applications access .Net (CLR) classes as if they were COM objects. Normally to use this interface you need to register the .Net (CLR) assembly using REGASM.EXE. There is a special provision in Win32 side-by-side that allows using .Net classes without that registration. Manifest Maker can create Win32 manifests for CLR classes as of version 2.4. Read more on using managed classes...

Please refer to Windows Server and IIS topics for discussion of web application isolation.

What are my chances of success

Every case needs to be evaluated individually. So far side-by-side has been successfully deployed in networks of tens of thousands of Windows desktops and thousands of servers. There is fine print, there are things that cannot be done and there are things that do not work. The good new is that we know what works and what does not work and we want to help you.

The bad news - things that do not work

Below is a list of major applications or scenarios that do not work. This discussion assumes that all code in question cannot be modified. Normally code purchased from outside vendors cannot be modified. Frequently it is impractical to modify software built in-house as well. If you are developing new code and are interested to make it smart about side-by-side, you can make it work with side-by-side regardless. You can simply actively manage your activation context and load all you want.

Out of process COM servers.

Out of process COM servers are separate programs that provide COM services to COM clients. They are started by COM either as services or as regular processes. There is no facility in side by side to handle any aspect of out of process COM servers.

Microsoft Office

We have not seen anybody succeed to side-by-side Microsoft Office. Any version of it. We recommend standard, out of the box installation of Office.

The silver lining: You can make manifests for Office programs (Word, Excel, Power Point, etc.) to make them use components packaged using side-by-side. This works well, although may require certain skill to set up.

Internet Explorer

It is not possible to side-by-side Internet Explorer. It is tied to the system in so many ways that we recommend against even trying it.

The silver lining: You can make a manifest for Internet Explorer (iexplore.exe) to make it use components packaged using side-by-side. This applies to components installed on the workstation. There is no facility to deal with ActiveX components dynamically downloaded from the Internet. Microsoft provides a COM component to manage the program Activation Context and it is possible to do that from inside a web page retrieved by IE. But, the page needs to be specifically developed to use this control and manage side-by-side.

Another problem with IE is that it has a peculiar logic for loading and instantiating components and it is possible to create web pages that will never use the component from your manifest. Microsoft is aware of this and they stated that changing this behavior is impossible without a major rewrite of IE.

Other applications that use unpredictable COM components

In order to isolate an application, you need to make an assembly with all components that application needs. Other applications that behave somewhat like IE may circumvent, intentionally or not, your attempts to isolate them.

The good news - things that *do* work

Almost everything else works. This is very good news. Practically all in-house applications work. Third-party components work. Third party applications work. Major middleware works side by side. All of this without any changes to the code - just add manifests.

Visual Basic 6 Programs

You need to build manifests for them. Note that due to the VB6 development environment behaviour you will not be able to develop using side-by-side. The resulting executable programs work as expected. Check Microsoft policy on VB6 support and look for required updates.

C/C++ Programs

You need to build manifests for them. Because normally C/C++ development environments create executable modules for debugging you can even isolate your application during development and debugging.

VB Scripts

You need a manifest for cscript.exe or wscript.exe. The recommended practice is to make a temporary copy of cscript.exe (or wscript.exe, as needed), place the cscript.exe.manifest file in the same directory then execute the program from the temporary location.

Vendor Applications

You need to build manifests for them.

Shrink-wrapped component libraries

All current versions and most old versions work.

Major middleware

You need to build assemblies.

Web Applications on Windows Server 2003/IIS6, 2008/IIS7,

See web application isolation topics for more details. To see a configuration example check Examples\IIS6Sample folder.

Private or shared

Once you decide to use side-by-side there is a new question: how do we package components? Options are:

- Private assemblies and merge modules.
- Shared assemblies.

The answer is very much individual, depending on your infrastructure, packaging methodology and IT philosophy. Generally private assemblies and merge modules are more local to the application but consume larger amount of disk space and create more demand for memory at runtime. Shared assemblies are more complicated to use but save disk space, lower run-time memory requirements and allow a more structured and controlled deployment and management.

We would like to help you

We provide consulting services and will be happy to help you make the decisions. We can provide a general information session to explain the technology, requirements and pitfalls. We can join you in planning and managing the implementation of side-by-side deployment.

Contact us:

e-mail
mailto:sxs@mazecomputer.com

snail-mail:
Maze Computer Communications, Inc.
3 Willow Court
Whitehouse Station, NJ 08889
USA