sign up log in tour help stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ×

Generate manifest files for registration-free COM

I have some applications (some native, some .NET) which use manifest files so that they can be deployed in complete isolation, without requiring any global COM registration. For example, the dependency on the dbgrid32.ocx com server is declared as follows in the myapp.exe.manifest file which sits in the same folder as myapp.exe:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
  <assemblyIdentity type="win32" name="myapp.exe" version="1.2.3.4" />
  <dependency>
    <dependentAssembly>
      <assemblyIdentity type="win32" name="dbgrid32.ocx" version="5.1.81.4" />
    </dependentAssembly>
  </dependency>
</assembly>
```

The dbgrid32.ocx is deployed to the same folder, along with it's own dbgrid32.ocx.manifest file:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
  <assemblyIdentity type="win32" name="dbgrid32.ocx" version="5.1.81.4" />
  <file name="dbgrid32.ocx">
     <typelib
        tlbid="{00028C01-0000-0000-0000-000000000046}"
        version="1.0"
        helpdir=""/>
```

```
    <comClass progid="MSDBGrid.DBGrid"
       clsid="{00028C00-0000-0000-0000-000000000046}"
       description="DBGrid  Control" />
  </file>
</assembly>
```

This all works fine but maintaining these manifest files manually is a bit of a pain. Is there a way to generate these files automatically? Ideally I would just like to declare the application's dependency on a list of COM servers (both native and .NET) and then let the rest be generated automatically. Is it possible?

com manifest dll regfreecom type-library

edited Mar 20 '13 at 10:39
GSerg
39.7k 5 57 103

asked Jan 21 '09 at 16:07
Wim Coenen
45.9k 4 103 183

+1 also: Retagged regfreecom as that tag is more common for registry free COM – MarkJ Mar 17 '09 at 20:12

4 Answers

It looks like the perfect solution does not yet exist. To summarize some research:

Make My Manifest (link)

This tool scans a VB6 project to look for COM dependencies, but it also supports manual declaration of late-bound COM dependencies (i.e. those used via CreateObject).

Interestingly enough, this tool puts all information about the dependencies inside the application manifest. The application exe and its dependencies are described as a single assembly consisting of multiple files. I hadn't realized before that this was possible.

Looks like a very good tool but as of version 0.6.6 it has the following limitations:

- only for VB6 applications, starts from VB6 project file. Shame, because a lot of what it

does really has nothing to do with VB6.

- wizard style application, not suitable to integrate in a build process. This is not a huge problem if your dependencies don't change a lot.
- freeware without source, risky to rely on it because it could become abandonware at any moment.

I did not test whether it supports .NET com libraries.

regsvr42 (codeproject link)

This command line tool generates manifest files for native COM libraries. It invokes DllRegisterServer and then spies on the self-registration as it adds information into the registry. It can also generate a client manifest for applications.

This utility does not support .NET COM libraries, since these don't expose a DllRegisterServer routine.

The utility is written in C++. The source code is available.

mt.exe

Part of the windows SDK (can be downloaded from MSDN), which you already have if you have visual studio installed. It is documented here. You can generate manifest files for native COM libraries with it like this:

```
mt.exe -tlb:mycomlib.ocx -dll:mycomlib.ocx -out:mycomlib.ocx.manifest
```

You can generate manifest files for .NET COM libraries with it like this:

```
mt.exe -managedassemblyname:netlib.dll -nodependency -out:netlib.dll.manifest
```

However, there are some problems with this tool:

- The first snippet will not generate progid attributes, breaking clients which use CreateObject with progids.
- The second snippet will generate `<runtime>` and `<mvid>` elements which need to be stripped out before the manifests actually work.

- Generation of client manifests for applications is not supported.

Maybe future SDK releases will improve this tool, I tested the one in the Windows SDK 6.0a (vista).

edited Dec 7 '09 at 22:04

answered Jan 25 '09 at 0:50

Wim Coenen
45.9k 4 103 183

I think you missed one option: mazecomputer.com but I know nothing about it the web site doesn't describe. – Bob May 17 '09 at 2:59

MMM will also redirect non-COM (standard) DLLs. I'm not sure the other tools do this. – Bob May 17 '09 at 3:04

Just a note for the nervous: the source for MMM has been released. On the downside, this appears to be because the author has decided to stop working on it. Still a positive sign. – Gavin Jul 25 '11 at 4:10

The site for MMM is nolonger up but the place where the source code was put is still available for v0.9 and v0.12. – Scott Chamberlain Mar 17 at 20:06

With the MSBuild task GenerateApplicationManifest I generated a manifest at the command line identical to the manifest Visual Studio generates. I suspect Visual Studio uses the GenerateApplicationManifest during the build. Below is my build script which can be run from the command line using msbuild "msbuild build.xml"

Thanks to Dave Templin and his post that pointed me the the GenerateApplicationManifest task, and MSDN's further documentation of the task.

build.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild
/2003">
    <Target Name="Build">
        <ItemGroup>
            <File Include='MyNativeApp.exe'/>
            <ComComponent Include='Com1.ocx;Com2.ocx'/>
        </ItemGroup>
        <GenerateApplicationManifest
            AssemblyName="MyNativeApp.exe"
            AssemblyVersion="1.0.0.0"
            IsolatedComReferences="@(ComComponent)"
            Platform="x86"
            ManifestType="Native">
            <Output
                ItemName="ApplicationManifest"
                TaskParameter="OutputManifest"/>
        </GenerateApplicationManifest>
    </Target>
</Project>
```

answered Jun 14 '12 at 15:22

mcdon
2,120 1 17 20

I think this should really be marked as the answer to this question. I am now using this to automate all of our manifest generation. Thank you @mcdon, you've saved me a lot of work. – Pete Magsig Jul 25 '13 at 1:55

I agree this is the best solution when building with Visual Studio. It is probably not rated higher only because it was posted so much later than the other answers – dschaeffer Mar 12 '14 at 21:09

Make My Manifest (MMM) is a nice tool for doing this. It's also possible to write a script to process all your DLL/OCX files using mt.exe to generate a manifest for each one and then merge them all together. MMM is usually better/easier, because it also handles lots of special/weird cases.

answered Jan 21 '09 at 16:17

jdve
128 5

2 I'm a bit nervous about this MMM thing; it's just a blog, freeware but no source code available, only a link to a "self-extracting exe" and I see comments about the utility causing XP to crash. mmm... – Wim Coenen Jan 21 '09 at 22:21

Those "crashes" were the MMM utility itself dying. This was fixed in version 0.6.5 but you'll want 0.6.6 anyway, since while still a beta it no longer expires. You can always use MT.EXE instead though as already suggested. – Bob Jan 21 '09 at 23:19

mt.exe is not generating progid's when I use it on native com servers like dbgrid32.ocx – Wim Coenen Jan 23 '09 at 1:21

Use of reg free COM with .NET authored components can apparently cause XP to crash - see this stackoverflow.com/questions/617253/… – MarkJ Mar 17 '09 at 22:23

You can use Unattended Make My Manifest spin off to generate manifests directly in automated builds. It uses a script file to add depended COM components. This is an excerpt from the sample ini with the available commands:

```
# Unattended MMM script
#
# Command names are case-insensitive. Reference of supported commands:
#
# Command: Identity
#
#   Appends assemblyIdentity and description tags.
#
#   Parameters       <exe_file> [name] [description]
#      exe_file      file name can be quoted if containing spaces. The containing folder
#                    of the executable sets base path for relative file names
#      name          (optional) assembly name. Defaults to MyAssembly
#      description   (optional) description of assembly
#
```

```
# Command: Dependency
#
#   Appends dependency tag for referencing dependent assemblies like Common Controls 6.0,
#     VC run-time or MFC
#
#   Parameters       {<lib_name>|<assembly_file>} [version] [/update]
#     lib_name       one of { comctl, vc90crt, vc90mfc }
#     assembly_file  file name of .NET DLL exporting COM classes
#     version        (optional) required assembly version. Multiple version of vc90crt can
#                    be required by a single manifest
#     /update        (optional) updates assembly_file assembly manifest. Spawns mt.exe
#
# Command: File
#
#   Appends file tag and collects information about coclasses and interfaces exposed by
#     the referenced COM component typelib.
#
#   Parameters       <file_name> [interfaces]
#     file_name      file containing typelib. Can be relative to base path
#     interfaces     (optional) pipe (|) separated interfaces with or w/o leading
#                    underscore
#
# Command: Interface
#
#   Appends comInterfaceExternalProxyStub tag for inter-thread marshaling of interfaces
#
#   Parameters       <file_name> <interfaces>
#     file_name      file containing typelib. Can be relative to base path
#     interfaces     pipe (|) separated interfaces with or w/o leading underscore
#
# Command: TrustInfo
#
#   Appends trustInfo tag for UAC user-rights elevation on Vista and above
#
#   Parameters       [level] [uiaccess]
#     level          (optional) one of { 1, 2, 3 } corresponding to { asInvoker,
#                    highestAvailable, requireAdministrator }. Default is 1
#     uiaccess       (optional) true/false or 0/1. Allows application to gain access to
#                    the protected system UI. Default is 0
#
# Command: DpiAware
#
#   Appends dpiAware tag for custom DPI aware applications
#
```

```
#   Parameters       [on_off]
#     on_off         (optional) true/false or 0/1. Default is 0
#
# Command: SupportedOS
#
#   Appends supportedOS tag
#
#   Parameters       <os_type>
#     os_type        one of { vista, win7 }. Multiple OSes can be supported by a single
#                    manifest
#
```

edited Feb 16 '10 at 17:20

answered Oct 14 '09 at 19:22

wqw

7,394 17 32

+1 Interesting, especially because the source code is available. I'm a bit confused by the name similarity, apparently "make my manifest" and "unattended make my manifest" are different tools by different authors.
– Wim Coenen Nov 20 '09 at 13:15

protected by Community ♦ May 17 '11 at 15:30

Thank you for your interest in this question. Because it has attracted low-quality answers, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?