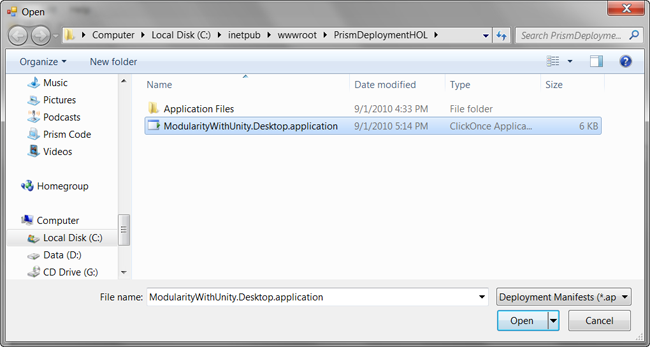
**To open the deployment manifest in the Manifest Manager Utility**

1. Open the file ManifestManagerUtility.sln, build it, and run it.

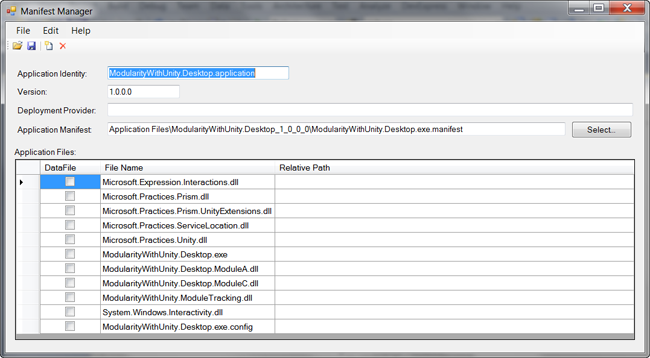
|  |
| --- |
| **Gg405497.note(en-us,PandP.40).gifImportant:** |
| You must run this utility as an administrator. If you are running this from Visual Studio, you must start Visual Studio as an administrator. |

1. On the **File** menu, click **Open**, and then navigate to the publish folder location where you published the QuickStart in the previous task. In that folder, select the deployment manifest file ModularityWithUnity.Desktop.application, and then click **Open**.



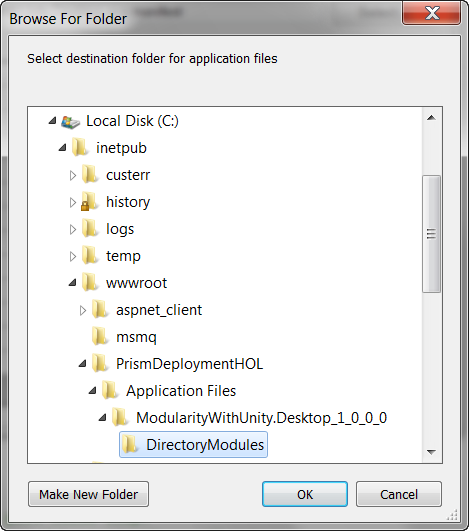
Open dialog box from Manifest Manager Utility in publish folder location

1. The deployment and linked application manifest files will be opened by the utility and will be presented in the unified view of the utility, as shown in the following illustration. You can see that the shell executable file and all referenced assemblies that are not part of the framework are automatically included. Note that Modules A and C are included because they were referenced for static loading by the QuickStart, but you will need to add the additional modules using the utility.



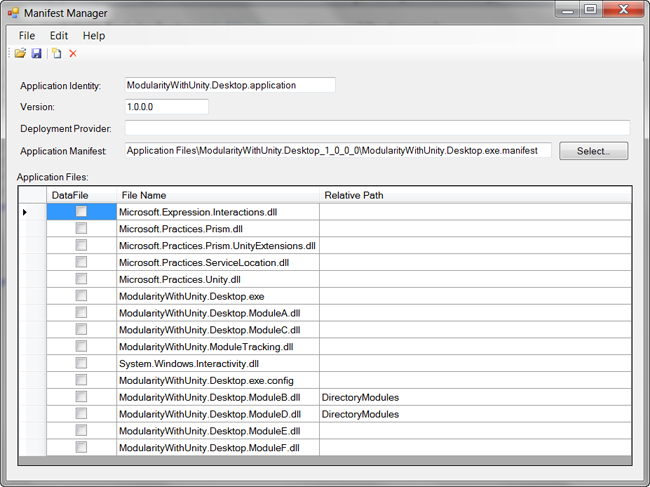
Manifest Manager utility

1. **To add the dynamically loaded modules to the manifest**
2. On the **Edit** menu, click **Add Files**. In the **Add Application Files** dialog box, navigate to the build output folder for Module B (such as C:\temp\ModularityWithUnity\ModuleB\bin\Debug\) and select the module DLL (such as ModularityWithUnity.Desktop.ModuleB.dll). In the **Add Application Files** dialog box, click **Open** to add the module DLL to the manifest.
3. When you click **Open**, a **Browse For Folder** dialog box appears. In this dialog box, you can specify the destination folder to copy the module file to the publish folder. Modules B and D are loaded in the QuickStart through directory scan, and the bootstrapper sets the folder it scans to a relative path of .\DirectoryModules from the executable file. This means the files need to be in that same relative path in the published application.
4. Select the version-specific Application Files folder, and then click the **Make New Folder** button at the bottom of the dialog box.
5. Name the new folder DirectoryModules.
6. Make sure the new folder is selected, and then click **OK**. This copies the Module B DLL into the DirectoryModules subfolder of the application files, as shown in the following illustration.



Browse For Folder dialog box with DirectoryModules subfolder selected

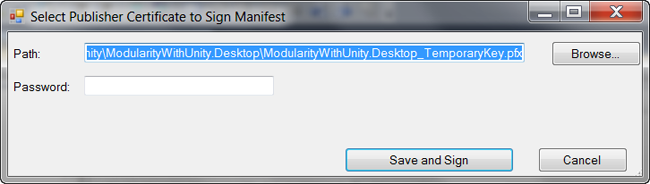
1. Repeat the preceding steps to add Module D to the manifest and place it in the DirectoryModules subfolder.
2. Repeat the preceding steps to add Modules E and F to the manifest, but those both go in the root Application Files folder (ModularityWithUnity.Desktop\_1\_0\_0\_0).
3. At this point, the additional modules should be listed in Manifest Manager Utility with the relative path shown for Modules B and D, as shown in the following illustration (order does not matter).



Manifest Manager utility with Modules B, D, E, and F added

**To sign and save the manifests**

1. Click the **Save** button on the toolbar of the utility. This opens the **Select Publisher Certificate to Sign Manifest** dialog box.



Select Publisher Certificate to Sign Manifest dialog box

1. Click the **Browse** button, and then locate and select the ModularityWithUnity.Desktop\_TemporaryKey.pfx file that was generated when you added the test certificate to the project in the first task of this lab.
2. Click the **Save and Sign** button, leaving the password blank again.

At this point, you have successfully published the application with modified manifest files and it is ready to install.