Chapter 3
Packaging the Installation

Creating a professional installation package can be as important as creating the defect free software product it installs. It is the first impression you create with end users as they install your product on their computers for the first time. Creating a professional installation package takes preparation and planning.

Solid planning done from the beginning of the project leads to a smooth deployment of your solutions. Constant communications with your customer throughout the development and testing phases of the software engineering cycle increases your customer's outlook, builds trust that your product is going to meet or exceed their needs, and increases the anticipation for the delivery of the product. As the development winds down and the customer prepares to perform acceptance testing, developers need to focus on what it takes to package up the installation setup.

There are many questions that need to be answered, like when is it ready to ship, what needs to be shipped to the clients, what features and schemes for deployment need to be packaged, what utilities need to be sent to the client to make support easier in the future, and how will the installation package be shipped to the customers. This chapter attempts to address each of these issues.

When is it ready?

There are three perspectives on when the product is complete and ready for deployment: the engineering viewpoint, the quality assurance check, and the customer determination that it meets the documented requirements.

The engineering viewpoint is comprised of the project leader (the customer's representative to the development staff), the project's developers, and other developers in the company. In a smaller shop, one person may be playing all the roles; in larger shops or corporate environments you may have someone different in each role. It should be obvious the product is not ready until all the features have proven to meet the customer's stated requirements. This is unit tested by the original developer, and system tested by the other developers and the project leader (if available). This proves the interface supports the business rules, which are implemented in data, and they all work together to manage the information correctly. Further, this proves all company guidelines and standards have been implemented in the code developed to support the solution that is ready to ship. These guidelines include proper coding standards, proper implementation of frameworks, proper use of third-party tools and controls, and proper implementation of industry standards if appropriate. The way to enforce these standards and guidelines is to either have a team walkthrough ("Defending Your Life," as Whil Hentzen calls it in Hentzenwerke Publishing's *The Software Developer's Guide, Third Edition*), or a simple desk check by one other developer. The goal of these review sessions is to make sure the code matches the requirements, is readable, and is understandable. A one-person shop has to do it all, but even one person shops typically have

contacts they can call on for review of code if needed. If not, take some time to print code off and review it when you are away from the computer.

The quality assurance (QA) perspective is almost self-explanatory. A quality assurance team strictly enforces the correct implementation of the customer's requirements. We know it is a real issue that many software shops really don't have a staff dedicated to quality assurance. We are not talking about hundreds of people dedicated to the testing of all developed products. This could be one person who knows how to be the "unknowledgeable user" and the "knowledgeable user" at the same time. This role can be filled by another developer(s), and is best filled by people who were not involved from a coding aspect. A key to success with a QA department/person is to have them involved from the start of the project. By having them review the functional specification and prototype, they can start building the test cases for the test plan.

The customer's acceptance testing is critical, otherwise you may not get paid and it will be time to concentrate on another hobby that can be turned into a paying job. On the other hand, do not rely on the customers to find your bugs. It is important to note that customers are usually not trained in finding bugs in software. The best you can hope for is they are business experts and find all the process bugs and miscalculations. They will likely point out every flaw in the interface (labels misspelled and unaligned by a pixel on a form). We always suggest you watch them struggle to add a new thing-a-ma-jig into your latest software creation. See how they use it—or more importantly, how they fail to use it. They may not even point out missing features for months. A thing like end of the month processing does not get tested until the end of the first production month even when you step them through the process during "testing." Little used features typically get their attention months down the road. The key to a successful user acceptance test is to give them a test plan when delivering a test version. Make sure every requirement is somehow tested by the test plan.

Testing

All software developers understand the importance of testing. The fact is we all learn the importance of testing and usually the hard way. The hard way is usually moving an application into production just after implementing a last minute feature and shortcutting the testing. This section is not going to step you through a complete system test, rather it points out a few things you can do after generating the golden executable so you can save some embarrassing last minute problems when you deploy it to your customers.

Testing is not complete unless you make sure all reports are printed (to various printers if possible), previewed, and output to various file formats (ASCII, Excel, HTML, etc.). Fire each and every menu option. Make sure the reindexing works (and the Stonefield Database Toolkit (SDT) metadata was validated if used), and the backup and restore processes function without incident. Two other issues we find developers frequently miss during testing is to verify the system conforms to the Windows' Color Scheme and the application conforms to the customer's minimum screen resolutions.

If delivery includes an EXE, always test the EXE standalone. There is nothing worse than having a major feature fail in front of the customer because one of the developers forgot to remove a `SUSPEND` from the form you want to show off. "Error 1001 - Feature not Available" errors are unacceptable because it shows the developer never tested the new functionality standalone. We always use the built-in EXE version numbers to differentiate between builds

that go to the customer. This way we know when the customer asks why a feature is not working in the version they are using (1.0.235), the answer is because it was added in the next version and they need an upgrade (to 1.1.59).

Is there a difference testing when shipping different types of applications? Is there a difference between shipping a bug fix, a complete new system, or upgrade when it comes to testing? What is the difference between delivering standalone components and a standalone app? Nothing, other than the amount of testing physically required. The same intensity used to system test a new app should be used to verify the latest bug found by the customer was squashed. The key is to test everything affected by the development completed. Maybe the engineering testing calls for you to desk check a bug fix and walkthrough a major change, but all of it gets tested by the QA group and the customer before it is declared ready for implementation.

Getting ready for delivery

Before you can package up the Installation, you need to make sure you have all the components ready to ship.

Executable components

The executable components need to be built. There are many options available that need careful consideration. Naturally, a decision about deploying an APP, EXE, or DLL is made long before you are preparing to build the executable for an immediate deployment. This decision should be made during the design phase of the application development cycle.

One thing to consider is what you want to do for your final build for the release. We strongly advise selecting the Recompile All Files and Display Errors options found on the Build Options dialog (see **Figure 1**). The reason for this is plain and simple. It is possible compiler-flagged errors will not show up in the current build even though they exist in the source code. How? If you edit the code and do a build, the build displays the errors if you select it to do so. If you do not edit the code the errors exist in and do a rebuild, the errors do not display (even if you select the Display Errors option) because the code was not changed and Visual FoxPro only rebuilds code that needs to be recompiled. Because the code was compiled and not edited, the last recompile time is after the last edit time and Visual FoxPro skips it during a build.

This is why it is important to check these two options when building an executable you intend to install for the customer to test or use in production.

The Project Information dialog (see **Figure 2**) is equally important to review before making a final build for the customer. The Debug Info option determines whether or not the source code is included in your compiled application. The source code is nice to have when errors are recorded because Visual FoxPro needs the source code to display the source line with the error and track the line number where the error occurs. However, it also makes it easier for someone to use a product like ReFox to decompile the source code. The Encrypted option does not stop products like ReFox from decompiling your applications, so we do not bother using this option (see Chapter 10, “Security” for more details). Using this option can reduce the ability to compress the executables, which leads to longer file transfers and downloads on the Internet.

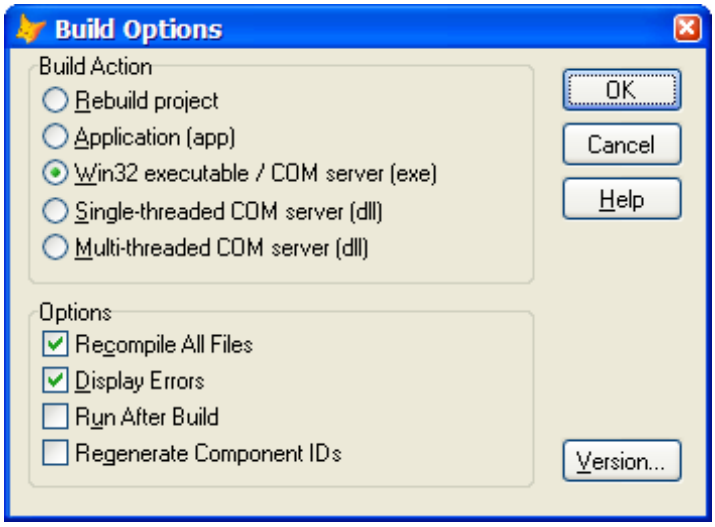

Figure 1. *The Build Options Dialog is critical to determine the contents of the executable you are going to deploy.*

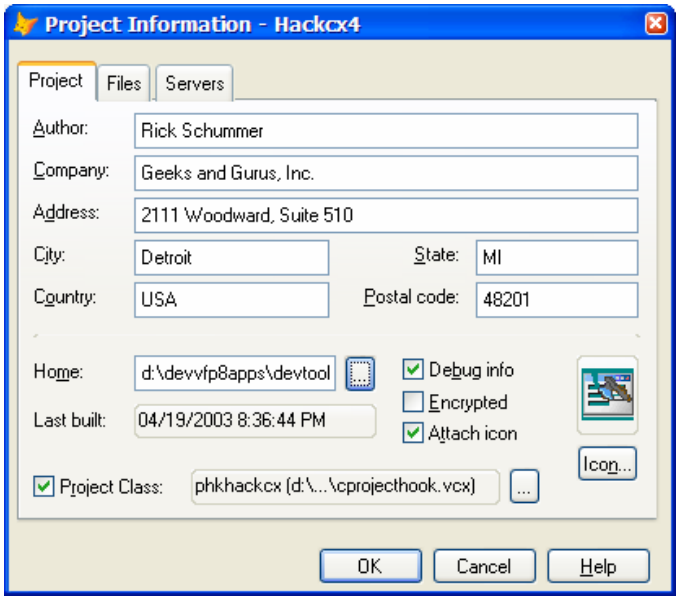

Figure 2. *Selecting the Debug Info bloats your executables by including the source code for the application into the EXE file.*

The Icon command button allows you to select a custom icon (**Figure 3**) to embed into the executables (not APP files). This icon should be an ICO file with 16x16, 32x32, and 32x32 with 256K color (if deployed on Windows XP) pixel icons. This icon is the icon that shows up

in Windows Explorer. Visual FoxPro 8 was the first version to start supporting the 32x32 with 256K color icons.

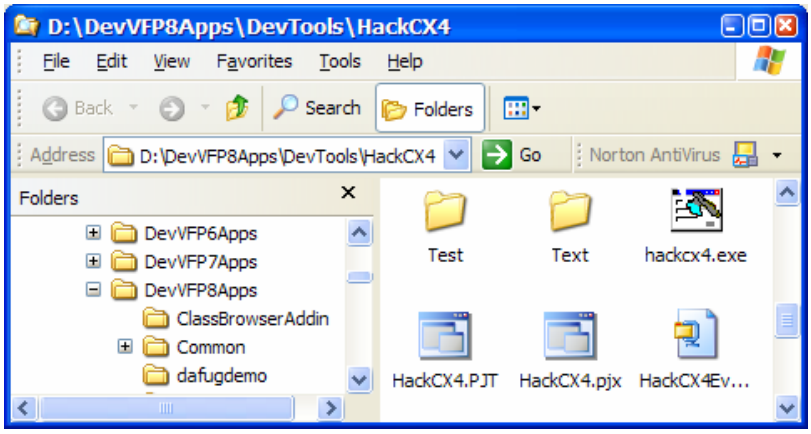

Figure 3. *Selecting the icon on the Project Information page determines the icon that displays in Windows Explorer.*

Help

Implementing a Help file seems to be the part of development the majority of developers avoid. Help can come in the form of user documentation (book style from a word processor) and/or online (Windows based HTML). A bug fix release may only need an update to the documentation, while new development may require all new help to be written. We find our clients avoid the cost of creating and implementing Help because they are seriously involved in the development of the application. We also know many custom applications never see Help because they are developed for a small user base already familiar with the business at hand.

If Help is implemented you need to make sure you include the necessary Help subsystem files so your Visual FoxPro applications can interact with the Help files. See the sections on "Visual FoxPro runtimes and dependencies" and "Help and documentation file" for details.

Data conversion

Update releases are common with custom software, in fact we always encourage our customers to add new functionality and make changes to existing functionality in their application because it means more business for us. Changes to the applications often require the database to change. New applications might be replacing a legacy application and we need to convert the data from the old data stores to the new database. If there is no legacy system to convert data you may still need to initialize a new database for the first use.

Updating existing VFP data is simple if you own a copy of Stonefield Database Toolkit (SDT). We have successfully used the SDT Update method for years to handle the structure changes. There simply is no single third-party tool as important to a Visual FoxPro developer as SDT if you use VFP data, and this is the biggest reason to buy it. However, there are still cases when the new structures need to be populated or data converted from different fields or from different tables.

Updating a backend SQL database is not as easy because there is no SDT for SQL Server, Oracle, MySQL, etc. You need a mechanism to run update scripts in the database. If you have a sophisticated customer with an on-site database administrator you might be able to just send a script file for the administrator to run. You might have access to the server and be able to run the script yourself. You might have to create a program that executes the necessary scripts programmatically. The scripts need to be run in a specific order and it might be a situation where you have to run some of the scripts, convert some data, and then run some other scripts.

The conversion of data from the legacy system can present numerous challenges. Data clean up and scrubbing, proprietary formats, lack of referential integrity, values outside of valid ranges or values that break new field level rules, and missing or corrupt fields are just some of the problems you face. Creating a test set that covers each and every case or situation can be difficult.

A new database can be implemented in many ways. Visual FoxPro data (with empty/initialized tables) can be copied to a customer's folder. You can also use the utility called GenDBC that ships with Visual FoxPro to generate a program with code necessary to create the database container, the empty tables, indexes, views, relations, and connections. SQL backend databases backups can be restored or scripts can be run to generate the necessary database objects.

Regardless of the data conversion needs, these conversion routines need to be developed and tested. They need to be tested often because data changes can happen at any time during the development phase. We cannot stress enough the need to test the conversions with real data. Work with your clients to get a recent copy of their production data to use in this test. Make sure to perform one final test of the conversion before delivering the installation.

Visual FoxPro runtimes and dependencies

Applications updated and released after a Visual FoxPro version upgrade require all the runtime files ship with your application. The service packs can include updated runtime files that need to be released with the updates to your custom applications.

It is very important to keep the runtimes in sync with the development version of Visual FoxPro. One example of a problem could be delivering an application using edit boxes with the original Visual FoxPro 7 runtimes. Microsoft released a hot fix for the runtimes soon after the release of Visual FoxPro 7. If you do not update the runtimes at customer sites, they will not have scrollbars in the edit boxes on their forms. This was corrected in the hot fix (and included in Service Pack 1).

So what files do you need to distribute? There are a few directories to check for new runtime files. Your directories could be different depending on the operating system and the directory structure where you installed Visual FoxPro. For Visual FoxPro 7 you have the following files:

- C:\Program Files\Common Files\Microsoft Shared\VFP\ contains the VFP7R.DLL, VFP7T.DLL, VFP7RCHS.DLL, VFP7RCHT.DLL, VFP7RCSY.DLL, VFP7RDEU.DLL, VFP7RENU.DLL, VFP7RESN.DLL, VFP7RFRA.DLL, VFP7RKOR.DLL, VFP7RRUS.DLL, VFP7RUN.EXE, FOXHHELP7.EXE, and FOXHHELPPS7.DLL.
- C:\Program Files\Common Files\System\Ole DB\ contains the VFPOLEDB.DLL (VFP 8 overrides the same named file as the one installed with VFP 7)

- C:\Program Files\Common Files\Microsoft Shared\Merge Modules\ contains the merge modules used by InstallShield Express and other install tools that leverage the Windows Installer technology. These files include VFP7RCHS.MSM, VFP7RCHT.MSM, VFP7RCSY.MSM, VFP7RDEU.MSM, VFP7RESN.MSM, VFP7RFRA.MSM, VFP7RKOR.MSM, VFP7RRUS.MSM, VFP7RUNTIME.MSM, VFPACTIVEDOC.MSM, VFPHTMLHELP.MSM, VFPODBC.MSM, and VFPOLEDB.MSM. The merge modules are not directly distributed to the customers, but are used by install tools like InstallShield Express and Wise for Windows Installer.

For Visual FoxPro 8 you have the following files:

- C:\Program Files\Common Files\Microsoft Shared\VFP\ contains the VFP8R.DLL, VFP8T.DLL, VFP8RCHS.DLL, VFP8RCHT.DLL, VFP8RCSY.DLL, VFP8RDEU.DLL, VFP8RENU.DLL, VFP8RESN.DLL, VFP8RFRA.DLL, VFP8RKOR.DLL, VFP8RRUS.DLL, FOXHHELP8.EXE, FOXHHELPPS8.DLL, AND GDIPLUS.DLL.
- C:\Program Files\Common Files\System\Ole DB\ contains the VFPOLEDB.DLL (VFP 8 overrides the same named file as the one installed with VFP 7)
- C:\Program Files\Common Files\Merge Modules\ contains the merge modules used by InstallShield Express and other install tools that leverage the Windows Installer technology. These files include VFP8RCHS.MSM, VFP8RCHT.MSM, VFP8RCSY.MSM, VFP8RDEU.MSM, VFP8RESN.MSM, VFP8RFRA.MSM, VFP8RKOR.MSM, VFP8RRUS.MSM, VFP8RUNTIME.MSM, VFP8HTMLHELP.MSM, VFPOLEDB.MSM, and VFP_GDIPLUS.MSM. The merge modules are not directly distributed to the customers.

You absolutely need to release at least two runtime files along with your executable (EXE, DLL). The primary runtime DLLs are the VFP8R.DLL (or VFP7R.DLL, VFP6R.DLL) and one of the VFP8RXXX.DLL (VFP7RXXX.DLL, VFP6RXXX.DLL) language resource file, where the xxx is three letters that specify the selected language. If you are releasing a component that needs the multi-threaded runtimes then you need the VFP8T.DLL (VFP7T.DLL, VFP6T.DLL) and the language resource file. The rest of the runtime files are optional.

You might have noticed the VFP8RUN.EXE and VFPACTIVEDOC.msm are missing from the list. The ActiveDoc class still exists in VFP for backward compatibility, but Microsoft removed the functionality in Visual FoxPro 8 because of security reasons.

Visual FoxPro 7 introduced a new dependency for runtimes, the Microsoft Visual C++ 7.0 Runtime DLL (MSVCR70.DLL). This file is found in a merge module called MSVCR70.MSM, located in the C:\Program Files\Common Files\Microsoft Shared\Merge Modules\ folder. The same dependency is necessary for Visual FoxPro 8.

Visual FoxPro 8 introduced a new dependency for the runtimes, the Microsoft Graphics Device Interface Plus (GDI+) runtime DLL. This file is found in the VFP_GDIPLUS.MSM merge module, located in the C:\Program Files\Common Files\Merge Modules\ folder. The GDIPLUS.DLL file must be registered before the VFP8R.DLL if you are manually installing them or using an installer package that is not using Windows Installer Merge Modules.

If you are using the **`CursorToXML()`** or **`XMLToCursor()`** function you need to include the Microsoft XML Parser (MSXML3.DLL, MSXML3R.DLL, MSXML3A.DLL) to the Windows

System folder. These files are found in the MSXML3.MSM merge module, located in the C:\Program Files\Common Files\Merge Modules\ folder. We strongly recommend using the merge module because it works around the issues of installing a set of files that are likely to be in use. Installing the new files requires a reboot of the computer.

If you are using the new XMLAdapter class introduced in Visual FoxPro 8 you need to include the Microsoft XML Parser 4 DLL files. These files are found in the MSXML4.MSM merge module, located in the C:\Program Files\Common Files\Merge Modules\ folder. We strongly recommend using the merge module because it works around the issues of installing a set of files that are likely to be in use. Installing the new files requires a reboot of the computer.

If you are creating an application that accesses a Web service you need to include the SOAP SDK Files (SOAP_CORE.MSM) found in the C:\Program Files\Common Files\Merge Modules\ folder, the Visual Basic Virtual Machine (MSVBVM60.MSM), Microsoft Component Category Manager Library (COMCAT.MSM), and the Microsoft OLE 2.40 (OLEAUT32.MSM). The last three merge modules can be downloaded from the InstallShield Express Merge Module download page. You go to the InstallShield Express support page and select the Merge Module link.

If you are creating an application for release on a Middle Eastern operating system, be sure to add VBAME.DLL file to your installation package and have it installed in the Windows System folder.

Microsoft released a new version of the Visual FoxPro OLE DB driver in August 2003 and again with Visual FoxPro 8 Service Pack 1. This driver supercedes the one released with original release of Visual FoxPro 8 and Visual FoxPro 7. It is Microsoft's recommendation to use this new driver over any of the drivers previously released no matter what version of Visual FoxPro you developed your solution in. If you need to have both loaded on the same machine, you need to rename the older DLL because the file names are identical. Here are the instructions provided by Microsoft:

1. Shut down any applications that might be using the provider, such as IIS, or any desktop applications that use the provider.
2. Open the ...\Program Files\Common Files\System\Ole DB folder
3. Right-click vfpoledb.dll, select Rename, and then change the name to vfpoledb_save.dll.
4. To restore the saved version, delete or rename this release version and rename vfpoledb_save.dll to vfpoledb.dll. It should not be necessary to re-register the provider, provided you did not uninstall the previous version.

The latest version of the Visual FoxPro OLE DB driver can be downloaded from the Microsoft download Web site: ***http://microsoft.com/downloads/details.aspx?FamilyId=0F43EB58-7A94-4AE1-A59E-965869CB3BC9&displaylang=en*** *and is included with Visual FoxPro 8 Service Pack 1.*

The Visual FoxPro 6 Setup Wizard placed the runtimes in the Windows System or Windows System32 folder. This was common practice and is an old standard. Newer operating systems attempt to clean up the situation referred to by the development community and recognized by Microsoft as "DLL Hell." The new location for the runtimes as they are installed by the merge modules is the C:\Program Files\Common Files\Microsoft Shared\VFP\ folder. You can still place the Visual FoxPro runtimes in the Windows System folder and your custom applications will find the runtimes just fine, but your application will not be following the new standards. We recommend following the new standards to avoid problems.

One of the problems with loading the runtimes to a single folder can occur when you have multiple applications developed with the same major version of Visual FoxPro. There are five service packs (SP) deployed for Visual Studio 6, four of which included new runtimes for VFP. If you developed and system tested one of your applications with VFP 6 SP3 and another with VFP 6 SP5 and installed the runtimes to the same folder, you would have only one set of runtimes (the ones from SP5). If Microsoft changed the behavior of a feature between the two service packs and you did not have time to test the application developed under SP3, your clients could find a problem with your application. The recommendation in this case (if you cannot perform the testing of the older app) is to include the runtimes in the same folder as the executable or in a special folder and pass the –D parameter to the executable to specify which set of runtimes are used. If you use this mechanism/setup, you will not be able to use the merge modules to install the runtimes. You need to deliver the runtime files individually and directly specify the folder to which they are to be installed.

The obvious way to distribute runtimes is to rebuild the distribution files via your installation tool of choice. If you are using InstallShield Express – Visual FoxPro Limited Edition or another Windows Installer based tool, the new runtime files are available in the merge modules. Include the correct merge modules, rebuild the setup, test, and distribute. This is the safest and possibly the most polished way to redistribute the runtimes.

There is nothing limiting you from directly copying the updated runtime files to the workstation. You can copy the changed runtime files from a network server to each workstation via something as simple as a DOS batch file, create a self-extracting zip file to be run on each workstation, post them on a web page with instructions to download them, burn them on a CD with an auto play that copies them, or have a process check for new updates each time the application is started to see if an update is available. The runtime files (VFP8R.dll/VFP7R.dll/ VFP6R.dll and the language resource file) only need to be registered using REGSVR32.EXE if your application uses Active Documents or are not loaded in the Windows System folder. Taking this approach might be the easiest way if you are on-site at a client's location, in a corporate environment, or just need to move a couple of runtime files to a couple of workstations.

In the past many developers thought they needed to build a complete install package each time they needed to load new runtimes, but in fact you have many alternatives. The method of getting the runtime files to the client workstations depends on many factors. You need to evaluate the problem and determine the best mechanism for the situation.

Third party controls and libraries

The hardest part about dealing with ActiveX controls is finding Visual FoxPro examples and needing to translate Visual Basic code to Visual FoxPro code. Over the years we became

good at doing this and found deploying the controls is now a close second as far as problems are concerned.

The first problem is determining what files need to be deployed. Is the file an OCX? Do associated licensing files also need to be deployed (or more importantly not deployed because they are for the development environment only)? What about COM registration and dependencies? All of these questions might make it sound difficult, but the reality of the matter is there is plenty of documentation and techniques to help developers find all the needed distribution files.

One technique we use to find the name of the ActiveX OCX file is using the Object Browser (see **Figure 4**) included in Visual FoxPro 7 and 8. The Object Browser allows a developer to browse the list of classes, constants, enumerators, events, interfaces, methods, and properties. For deployment purposes it shows us the file name and the folder where it is located.

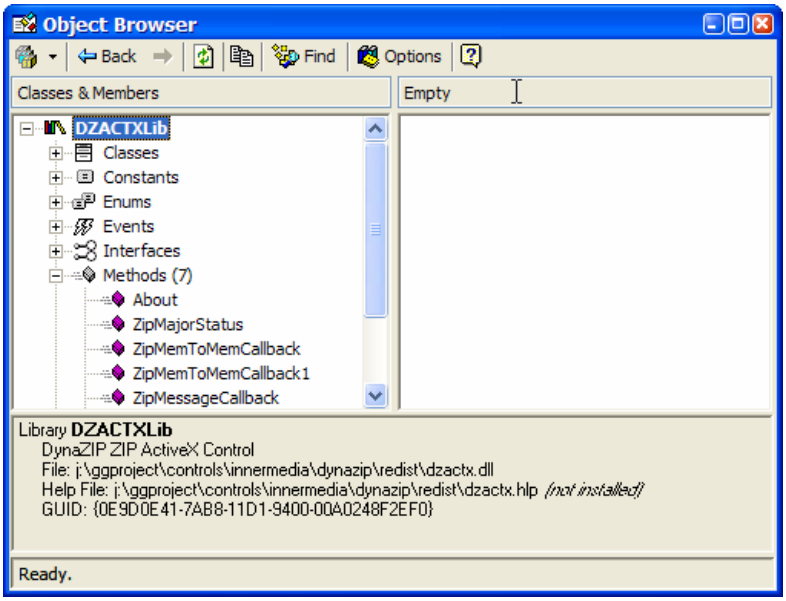

Figure 4. *Opening an ActiveX control in the Visual FoxPro Object Browser and selecting the top node displays the file name of the ActiveX control (in this case you can see the DynaZip ZIP ActiveX Control) in the lower pane.*

Organizing the setup

Before you can build the installation setup you need to know where the various files are going to be installed in the customer's environment. First you need to determine the files as we previously discussed, now you need to determine the directories where the files will reside and on which machine those directories will be placed. After you resolve the directory issues, you need to determine the installation schemes and how to deploy the feature sets so the users can implement the system efficiently.

Directories

The Setup Wizard that shipped with Visual FoxPro 3/5/6 required you define the directory structure and include the files in the directories for the Wizard to include in the Acme Setup exactly as you wanted them on your customer's machine. This forced developers to plan the installation directories before they created the installation routine, but required a second copy of all the files, and as new ones were generated, developers had to remember to copy the new files to this Setup Wizard structure. The new installer tools do not require the separate directory structure to indicate which files are distributed, but you do need to understand where the files reside in the production environment.

Each developer is likely to have their own ideas on how their customer's environment should look. There are corporate standards, developer recommendations, and industry standards to consider too. We offer some items and ideas for consideration in this section.

We dedicated a complete chapter (Chapter 5) to the Windows Installer technology as well as four separate appendices on how to work with and use some of the installer tools. Each of these tools has techniques to create directories and copy the files to these directories. You will see in the chapters a number of operating system and installer-recognized folders exist to assist you in picking standard directories and allowing the flexibility to create your own.

Executables

Executables are installed on servers (multi-user), on workstations (single-user and multi-user), on local area networks, wide area networks, in distributed mode, and on Web servers. The current standard is to default the install of the executables in a folder under the Program Files folder. The ultimate control should be placed in the user's capable hands. We find it quite irritating when a package installs itself where it deems fit, not where we want it. In fact, we consider deleting and not using a package not friendly in this regard. We hold ourselves to this standard as well. We also understand some corporate environments do not want the users to control things of this nature and we respect those deployments by removing this flexibility.

Some packages install the application in a folder directly below the Program Files folder and some companies create a company folder below the Program Files folder, and then install all the applications from their company in a folder under the company folder. This is the approach we like, but it is not required. In fact, there is nothing saying you cannot install your application directly off the root folder. We just feel it gives the application a more professional look when it follows industry standards.

Help and documentation files

The Help files are typically installed in the same folder as the executable to simplify locating the Help when the F1 key is pressed or the Help menu option is selected. Documentation can come in many forms such as word processing document files, Adobe Acrobat files (PDF), or additional help files. We typically install the documentation files in a folder called DOCUMENTATION under the directory where the executables are loaded. There is no industry standard to reference for these files.

In the case of HTML Help, you need to be aware there is a runtime Help subsystem that needs to be installed to access Help from a Visual FoxPro based application. The subsystem is easily installed if you use the VFP8HTMLHELP.MSM merge module. If you do not install the Help runtimes via the merge module you need to install the FOXHHELP8.EXE and

FOXHHELPPS8.DLL files (for Visual FoxPro 8) or FOXHHELP7.EXE and FOXHHELPPS7.DLL files (for Visual FoxPro 7).

FLLs

The only FoxPro Linked Library (FLL) files we are currently distributing with our applications are FoxTools (that comes with VFP) and, when necessary, the Amyuni PDF-generating FLL. Our experience has lead us to install the FLL files in the same directory as the runtime files. The only reason for this is to be able to find the file easily with the **HOME()** function. When we want to set up the library we have code similar to this:

```
IF "FOXTOOLS" $ SET("LIBRARY")
   * Nothing to do
ELSE
   SET LIBRARY TO HOME()+"FoxTools.fll" ADDITIVE

   THIS.lAddedFoxTools = .T.
ENDIF
```

When we want to release the library we have code that looks like this:

```
RELEASE LIBRARY HOME()+"FoxTools.fll"
```

This does require two copies of each FLL to reside on the development machine because sometime we run the application inside Visual FoxPro (which looks in the VFP folder for the FLLs), and other times we test the EXE standalone (which looks in the folder with the runtimes). We find this extra copy on the development machine to be a non-issue because it simplifies testing and implementation in the customer environment.

Shortcuts

Shortcut files are a nice way to give your clients access to the applications you develop. A shortcut file is nothing more than a system level file that opens the file it is a shortcut for. The most common place shortcuts are created is on the Start Menu, but they can be created on the Windows desktop, on the Windows taskbar in the Quick Launch toolbar, or in the Start Up folder. You can also create subfolders under these folders and add shortcuts to these new subfolders.

What you create shortcuts to is going to depend on your client's needs. Things to consider are the application executables, the Help file, custom Adobe Acrobat files (containing documentation, special instructions, last minute announcements, or updates), a bug reporting application, and any utilities you might have for the application. Consider anything that is important for the customer to have quick access.

Runtimes

The runtime location was discussed earlier in the runtime section under "Getting Ready for Delivery." We recommend you follow the standards deployed by using the merge modules, but as stated in this section, different circumstances will dictate the course of action you need to take for your own installations.

Data

Selecting the data directory folder should be similar to selecting the application folder, and under the control of the user/customer. At least this is the case for Visual FoxPro data. The location might also be dictated by the architecture of the application.

SQL backends have processes of creating and restoring data that dictate and/or allow data folders to be selected. Again, corporate standards or a database administrator might dictate where the data goes, which is all the more reason to make this part of the installation as flexible as possible.

Third party ActiveX and COM

We recommend checking the documentation provided by the vendors of all third party ActiveX controls and COM objects. The documentation should have a deployment section that tells you if there are any deployment requirements or recommendations. Also check to see of the vendor provides a Windows Installer Merge Module to include in your installation. The nice thing about merge modules is they already contain the necessary registration information for the control.

Images, icons, video, and sound

The rest of the "external" files are files integrated into the application at runtime instead of compile time. These files may be generic or custom to a specific deployment. For instance, you might include the company logo as an image on the splash screen, about window, and on reports. Each customer naturally has their own custom logo so you might include a LOGO.JPG in an image folder that each company can replace with their own custom file (or might be one you include in the setup). Icon, sound, and video files might be included or excluded depending on size, and the need for customization at a particular location.

Installation schemes/feature sets

After the directories are created and the correct files are placed in them, you need to determine what installation scheme you need for the release. We are sure there are numerous schemes to create an installation process. The following ideas are ones we found successful for our Visual FoxPro application deployments. We may combine several installs into one package or we might ship separate installs based on the customer's environment or needs. We separated the upcoming discussion based on the different options you can include in an installation package.

File server install

This is the main application installation and is included in nearly every installation package delivered to the customers. It includes all the core application executable files needed to run the application. This is the "base" installation and includes all the files found in the installation root, system, sound, images, and any other directories needed by the application.

This install is typically used in a network environment. The installation loads the core application files on the file server in a specified directory. Once the files are loaded, the users have access to the application provided they have network access and rights to these files. Using this scheme also requires all the workstations have the Workstation Install (discussed next) loaded so they have the Visual FoxPro runtimes.

This installation also works on a client PC for a single user application provided the files from the Workstation Install and the Data Install are also installed. Many developers who have single PC clients use this technique all the time.

Workstation install

If the application is loaded on the file server, is it ready to be executed by the connected workstations with security access? Not necessarily. Each workstation needs additional files loaded. These include the Visual FoxPro runtimes, any ActiveX controls, and the Help system engine. You could go around to each workstation and reload the File Server Install (discussed in the previous section), but this can be time consuming and unnecessary. The idea of the Workstation Install scheme is to load only the files needed. We broke the Workstation Install into five components, each can be included on one installation CD-ROM. You may find there is more or less than this depending on your needs.

- Visual FoxPro runtimes: we find application startup performance is best when the Visual FoxPro runtimes are loaded on each client workstation. Do the runtimes really have to be on the client workstation? No, see the –D directive to the VFP8.EXE (and consequently, your custom executable), but it is faster because you are not pulling 4MB down the pipe every time you start the app. This install loads all the Visual FoxPro runtimes including VFP8R.DLL and VFP8RXXX.DLL (where the xxx is the language of the Visual FoxPro version you have, e.g., ENU for English).
- Multi-threaded Runtimes: this loads the VFP8T.DLL file for multi-threaded COM objects. Not all applications require the multi-threaded functionality, so installing this for your customers may not be required.
- HTML Help Engine runtimes: these files are only needed if you include a CHM file with your application. If you decide to build WinHelp files (HLP) or a table based Help file, you do not need this installation.
- Visual FoxPro ODBC driver (VFPODBC.DLL and VFPODBC.TXT file) or OLE DB driver (VFPOLEDB.DLL) and others if needed: this gives your users a way to analyze their data via tools like a spreadsheet, perform mail merges from a word processor, or build their own queries via an end user database or tool.
- ActiveX controls: the key to this install is to make sure the ActiveX controls included in the application are installed on the workstation. These files are loaded into the appropriate directory and installed in the Windows' Registry. You need the ActiveX controls loaded on the computer where the installation is built. It is important to note the ActiveX controls loaded during an install can be the ones included with Visual FoxPro and Visual Studio as well as any third-party controls purchased.

Any of these components can be included or excluded from the install based on the requirements of the application and the deployment. You may decide to customize for a specific customer. It may be that you build one app with various ActiveX controls and another app for a different customer without ActiveX controls, or a different app with different controls. This installation can be passed around from computer to computer or run from a directory on the server (just installed on the workstation). We also recommend the CD is dated

and note the Visual FoxPro version and Visual FoxPro service pack that the runtimes apply. It can be embarrassing to have that new Session class in Visual FoxPro 6 Service Pack 3 not be available with a new executable running on prior versions of the runtimes.

Data install

Obviously this installation section is for the application data. The questions that need to be asked may complicate this seemingly easy setup, though. What files need to be sent? Is this Visual FoxPro local data or are you using a SQL backend database? What tables need to be pre-populated? What files can be generated at the customer site? What about installations that already have previous installations with data loaded?

Initial installations require all the application data be installed and found in the installation data directory. We like to keep this installation separate, especially for a new service pack release. Vertical market applications like this scheme as well because it allows a development shop to build a single installation package for new customers and existing customers getting a new version.

Web Server Install

A Web Server Install may mirror a File Server Install scheme in many ways. However, you may encounter many differences with your application. Most likely you will install the Multi-threaded runtimes for scalability, COM components or an EXE, HTML files, and will make Web Server settings (via executable or manual settings) like scripting files, user security, and the mapping of drives to the data.

How do you package the install?

Now that you have developed schemes for the installation, you need to determine how to package it. We are not referring to the box the CD is delivered in. The marketing expert can best handle this. We are suggesting you need to think about what install schemes discussed in the earlier section need to be packaged and sent to the customer.

Typical brand-new installs for a LAN based application require the File Server, Workstation, and Data Install schemes. Updates may only require the File Server Install. On the other hand, if the executable is built with a new version of Visual FoxPro you may need to ship at least part of the Workstation Install to include new runtimes. A Web Server may only need some new HTML files so you can skip the need to update COM objects. Single computer customers may require the File Server Install and the Workstation Install be combined into one installation process.

The packaging of the install is as important as developing perfect software because it is generally the first impression most users have of the application in production. We are not talking about the people you developed the software specs and performed acceptance testing with; we are talking about the possible hundreds of end users that actually get the package loaded on their computers.

OS and third party components

Today's software can easily be based on components. While we like to think Visual FoxPro is a comprehensive development tool, it is not uncommon to find our applications interacting with a specific operating system and third party components. The component environment is

powerful and can provide features that would take us a long time to develop. This saves developers' time and clients' money. The flip side of the equation is you need to ensure the components are installed. This means an extra burden to find out the licensing issues for deploying the component, how it can be done, and what mechanisms are available that can be leveraged.

While we hope our customers have the latest and greatest operating system, and they have all the components available, this is typically not the case. You definitely cannot rely on the environment being established exactly as our applications need it. It is not unusual to interact with the Windows Script Host (WSH), Microsoft Data Access Components (MDAC), and the Microsoft SQL Server Desktop Engine (MSDE). All of these components can ship with our installation setups, or be downloaded from the Microsoft Web site by our customers. How they get there depends on how you want your application perceived, how sophisticated your customers are (they may have an internal computer department that will update the machines), and how well you can integrate the component in your setup.

The same concepts need consideration when dealing with ActiveX controls and third-party COM objects. You may find the vendor already has a package you can use for your deployment, or you might have to consider creating the necessary registration process. Most of the current installer packages read the ActiveX control or COM object and make the necessary registrations for you during the installation. If you are not using an installer to do this or are doing things manually, you need to figure out what has to be done so the application and the operating system are set properly to work together.

You will find the optional operating system components from Microsoft, ActiveX controls included with Visual FoxPro and Visual Studio, as well as many third-party products have merge modules to use with installers that leverage the Windows Installer. More information on merge modules and the Windows Installer is found in Chapter 5. The install package vendors usually have many merge modules for components available for downloading on their Web site.

Utilities to consider shipping

While the main goal of the developer is to install the files needed for the customer's application, there are a number of developer utilities that can assist in the installation and ongoing maintenance that is likely to occur. The following are concepts for the tools we developed, downloaded from the Internet, or purchased for the types of applications we deliver to our customers.

Reindex and database updater

Initial releases usually start with an empty database or have a data converter preload the database. What happens when an upgrade is released and the latest alterations to the database tables, indexes, views, and relations need to be implemented? You could write a custom program each time to make these changes via the powerful `ALTER TABLE` and `INDEX ON` commands. You can also keep track of each change made to the data and make sure this custom program is updated every time a change is made in development. Even for single developer projects this can be a tough task, and the odds of missing one critical change are nearly even. Multi-developer projects get to be more than a challenge in this respect; they become a communication nightmare.

Keeping track of these changes can be automated. We do not want this to become a commercial for the Stonefield Database Toolkit (SDT), but we find so much value in this product we have to give it a plug. It keeps track of your changes to the data structures in the DataBase Container eXtensions (DBCX) and SDT metadata. SDT provides a *NeedUpdate* method to check for differences in the metadata and the structures in the database. If there are differences you can run the SDT *Update* method and the structural changes are applied to the database tables. This means columns are added or removed, column name changes are applied, and code pages can be changed. The same is true for indexes. It also creates new tables if they do not exist. This is all handled based on using the DBCX extensions to the database via the Stonefield Database Toolkit or a tool you developed yourself (yes, you can develop one since DBCX is a published standard). The key to this is to validate the database extensions before you ship out the metadata with the release (a lesson learned on this author's very first release with this product). SDT can be used in initial installations to completely generate all the tables as well.

There is another option to solve the database changes and that is to simply make the changes manually. If you develop on-site with the application, you can just use VFP live on the data and make the changes. We hear of this all the time. However, we are just not the kind of developers that trust ourselves to remember to make the changes in the same fashion we did in development. We are sure there are plenty of war stories to be told that would convince you not to do this. On the other hand, emergency fixes that can be done to keep a customer alive are made all the time.

One thing SDT does not handle that you need to consider for all types of releases is data conversion. Even if you have an automated way of updating database structures and the like, you need to consider a mechanism of populating new fields, converting data from old tables, and cleaning up data that might violate new field or row level rules. You might even need a separate program that cleans up the data before implementing a new field or row level rule for a table.

SQL databases require a different approach because there is no tool like Stonefield Database Toolkit for SQL backend databases. You need to generate scripts to execute on the backend database. These scripts can be generated or programmed. The scripts can then be run manually or programmatically on the backend database. We know developers that create script logs in a DBF and programmatically execute the scripts in a certain sequence to bring databases up to date for a specific version of the application. We recommend checking Chapter 10, "Application Distribution and Managing Updates" in Hentzenwerke Publishing's *Client-Server Applications with Visual FoxPro 6.0 and SQL Server 7.0* for plenty of ideas and suggestions on approaches for deployment mechanisms and tools for new and updated database schemas.

GenDBC/GenDBCX

If you are not using SDT and/or want a mechanism to generate a Visual FoxPro database and all the table structures, views, indexes, and relations, take a look at GENDBC.PRG (included with each release of VFP) or GENDBCX.PRG. GenDBCx is a third party tool written by Steve Arnott, which is available for free. You can download it from **http://www.leafe.com/dls/vfp**. Both of these tools generate Visual FoxPro program code that builds the database from scratch. Just like the SDT *Update* process, neither of these tools populates the tables so you need a mechanism to accomplish this task.

If you run the code generated by GenDBC or GenDBCx on a production directory that already has the files populated with customer entered data, the data will be lost because the programs create new structures without data.

Checking next ID table

Developers who use surrogate keys (meaningless integers or characters that uniquely identify a record in a table) often have a table that contains the next key for tables (Visual FoxPro 8 has the new AutoIncrement field type to also use for this purpose). Periodically these tables get misaligned with the real data in the tables. This can happen because the developer writes bugs in their applications; tables get zapped moving from development to production without updating the next surrogate key table, incorrect referential integrity rules, or the planets being out of alignment.

For whatever reason, the next ID table needs to be synchronized with the data in the tables. This process needs exclusive use of the database and each table. The general algorithm is to get the maximum key value from the table via code like:

```
SELECT MAX(nTablePK) ;
   FROM Customer ;
   INTO ARRAY laMaxID
```

Once you have the maximum ID for the surrogate key, the next ID table record for the table is updated with this new value if necessary. It is a good idea to run this process for all the tables in the application periodically. One red flag that indicates it might be time to give this process a run is the constant calls from a customer that they cannot add any records into any form because they are getting a message indicating duplicate keys values.

If you are using Globally Unique Identifiers (GUIDs) you do not need a tool like this. GUID keys are not incremented; they are generated on the fly by making calls to the operating system.

Configuration/Control table updater

Many applications have an INI file or a configuration table. When new options are added a mechanism to incorporate these options into an existing application needs consideration. Personally we prefer to work with tables since we work with these all day and Visual FoxPro provides plenty of commands to manipulate data. Adding new options is as simple as an **APPEND FROM** or running code to do an **INSERT INTO**. Updates are as simple as a **LOCATE** or **SEEK** and using **REPLACE**s, or writing code that does a SQL Update. The INI text files are also easy to work with because Visual FoxPro has low-level file input and output commands to manipulate text files. There are also Windows' API calls to INI files available for developers with this knowledge.

The important item to note is you develop some mechanism to update this information so the customer application does not malfunction when new options or features are added.

Runtime command window

FoxBox is a utility developed by Alex Korot of West Bloomfield, Michigan. It is not something you will use everyday interactively within the Visual FoxPro development environment, but it could be a tool you find indispensable for your customers when you are on-site or doing some remote support via pcAnywhere or a Remote Desktop Connection. FoxBox, shown in **Figure 5**, is a Command Window (and much more) without having a full copy of Visual FoxPro loaded on the customer site. It does require the same runtimes installed for your application.

*FoxBox can be downloaded from **www.KirtlandSys.com** or **www.RickSchummer.com***

This tool allows you to run any command in Visual FoxPro that does not fire the "Feature not available" error. Therefore, you can open a table and browse it, you can perform SQL-Selects to inspect data, you can fire up a `REPORT FORM`, recreate an index, or you can open text files. All the things you want to perform with Visual FoxPro, but do not want to build into your executable, can be accomplished via FoxBox.

The cursor browser (via browse button on FoxBox Console) displays a smarter browser. You can sort the cursor based on available indexes and by clicking on the column headers. The Structure button displays much more than the `LIST STRUCTURE` command produces.

Delivery mechanisms

There are a number of mechanisms to deliver the setup executable and supporting files. Understand the infrastructure so you are not surprised by not being able to load the application because they do not have a machine with a CD player or Internet connectivity. Your delivery method is also going to depend on other factors. Are you distributing to an individual, pushing to a few customers, or a vertical market with dozens or maybe even hundreds or thousands of implementations?

Diskettes

We know diskettes seem like yesterday's technology, but many still work with a notebook computer that does not have a built-in CD burner. We use our diskette drive to copy files to a customer in a pinch while we are at a client's office. Granted, this is not common, but it is still a mechanism for installation. All the modern installation tools build the needed files sizes to be copied to the diskette media. It is uncommon today, but there are still companies using computers without CD drives in them and the only way to load them physically is a diskette drive or a network connection (if they are connected to a network).

There are two big problems with diskette delivery. The first is packaging. CDs are cheaper to send via snail mail because you send fewer items. Have you ever had to ship six diskettes to 100 customers? The cost can be expensive with postage, duplication, and assembly. The second problem is your company image. Most of us are hired because we are at the leading

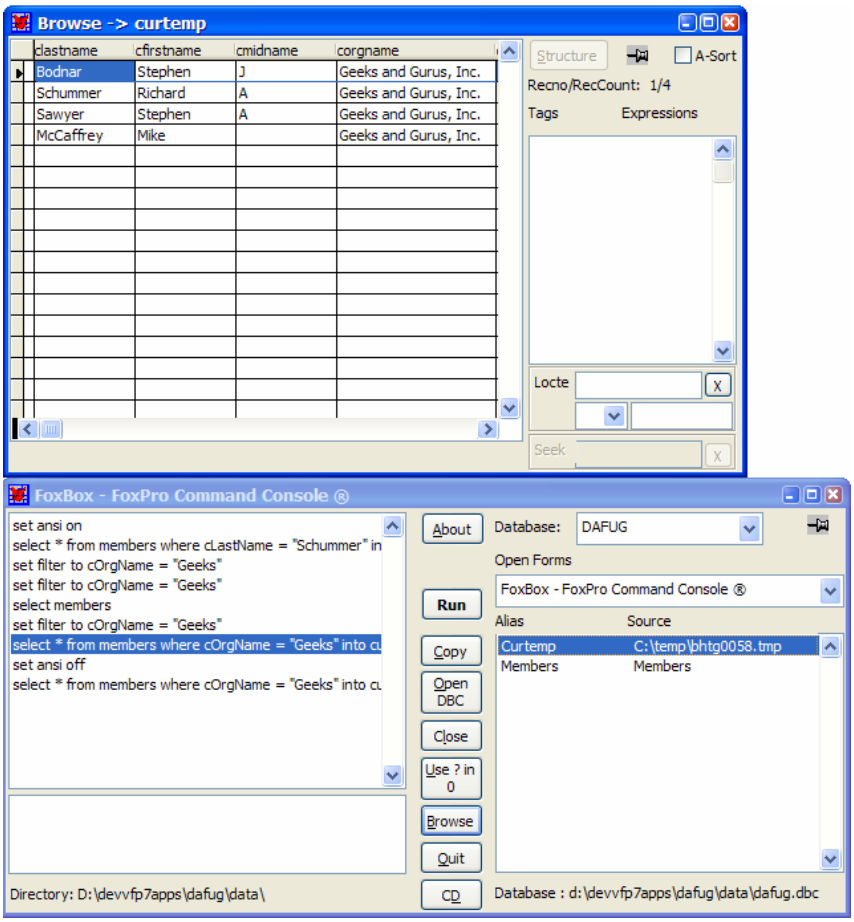

Figure 5. *FoxBox is indispensable when debugging support issues on a customer's computer that does not have Visual FoxPro installed.*

edge of technology. Obviously diskettes are considered older and CDs are considered closer to current technology. We work with a number of customers who find the latest bleeding edge technology to be a must because it helps with their image to their customer base. They expect the same from their vendors. Personally, this author really hates the idea of installing a package that comes on several diskettes. It simply takes more time because diskette drives are slower and require more operator intervention.

One other problem with diskettes is the reliability issue. This technology has been around for decades, but it never seems to fail that Murphy's Law applies when an important mission critical release is in the works. The diskette is bad, the copy process missed an essential bit, or the mail delivery process performs a magnetic re-alignment of the diskette surface. Over the years we have known many lucky developers, and others that have had nothing but trouble. Your mileage might vary.

Zip/Jaz/SuperDisk disks

The super disks are common and were a big step up from the 1.44MB floppies we were accustomed to using before CD-ROM burners became reasonably priced for developers. These drives are inexpensive, portable, and can be connected to our client's machines if they do not have one of their own.

One of the big advantages in using this type of media is we could fit an installation for a moderately sized application on one disk. It is cheaper to mail one small super disk than the comparable seventy diskettes. Today Iomega zip drives come in 100MB, 250MB, and 750MB (more space than a CD-ROM).

We used this delivery mechanism frequently for local clients before we had our CD-ROM burner. It is still a handy way to handle clients that do not have CD-ROMs (few, but they exist).

The problem you may encounter with this technology is you need to use the lowest common denominator and while you might have an Iomega Zip drive, your clients might be using a Jaz drive, have a SuperDisk, or a Syquest drive. This is why we purchased a nice case for our Zip drive and brought our own. Also be sure to check the way it is connected to the machine (parallel or USB port) and to make sure the client has the correct ports to connect the drive.

Memory sticks

Have you seen these new fangled memory sticks (also called a memory drive) that plug into a USB port? The first thing we thought of when we saw one of these is, "cool, what a nice mechanism to transfer a set of small files from one machine to another". These memory sticks started out small, but are readily available in sizes up to 256MB (at least in early 2003). We are sure they will get even more memory. A similar concept is the PCMCIA adapters or USB adapters for CompactFlash, Secure Digital, or SmartMedia cards that have even more memory.

This is not a mechanism for mass distribution of the initial installation for a large project, but it can be useful if you are on-site with a client and need to add a quick feature or fix a problem, recompile the app, and copy over the new executable for testing. These memory sticks show up as a new drive on the computer they are connected to. No need to reboot, just plug it in, wait for the operating system to recognize it, and copy the files.

CD-ROM and DVDs

CD-ROMs are probably the second most popular delivery mechanism next to downloading from the Internet. The CD format provides a minimum 650MB of disk space for files and the format is supported by all the modern installation tools and even something as dated as the VFP Setup Wizard that shipped with Visual FoxPro 3, 5, and 6. CD-ROM burners are very cheap these days and the software works reliably. We have been burning CDs for the last few years and customers find the advantages to be significant.

CDs provide for a much faster installation process because they have a faster transfer rate and there is no diskette swapping. Vertical market creators find there are fewer disks, which leads to cheaper packaging. The cost savings are more significant the more customers you have in the user base. CDs are also more durable than a diskette and take less storage space in the customer's office.

We always deliver customers a CD for initial installation, especially for runtimes, ActiveX, and other support files. One CD has a client computer installation and another has a one-time server installation. These can be on the same CD as well depending on how you architect the installation setups. This way support staff (either on-site MIS, technical support personnel, or a representative from our company) can go from PC to PC and install the runtimes on the workstations. This install can be run from the CD or run from the server.

Add polish to the package by creating a label for the CD. This can be as simple as running the Label Wizard in Microsoft Word (**Figure 6** and **7**) or getting a label creation software package at the local office supply store. We cannot tell you how big a difference the reception is when we started doing this. This is a simple process and can take less than a few minutes to do depending on the complexity of the label created. We simply note the application name, version number, release date, and include our company logo as well as the customer's company logo if it applies.

The disadvantage to the CD-ROM delivery mechanism is it has to be either hand delivered, shipped via the postal service, or possibly one of the overnight vendors. How fast it gets in your customer's hands depends on geographic distance, whether you ship priority overnight, or through standard mail channels. Although CDs on computers are very common these days, occasionally you may run into a client site where they have older machines without a CD or a policy of not buying CD readers because they do not want employees to play music while they are working (yes, this is not just a Dilbert story).

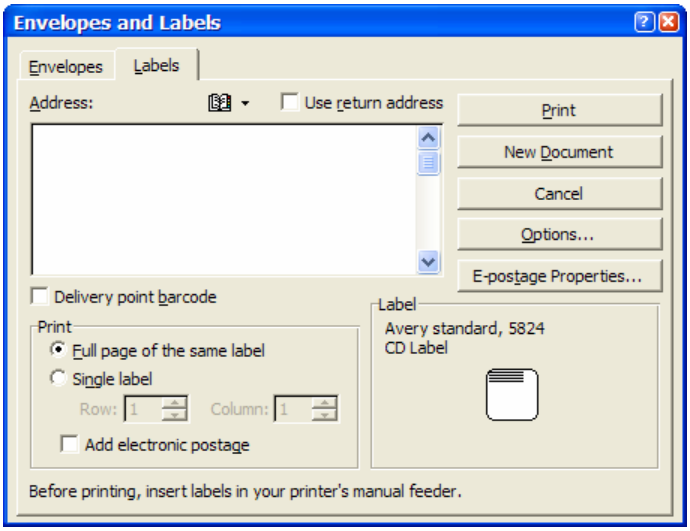

Figure 6. *The Microsoft Word Envelopes and Labels wizard is a quick and simple way to create a CD-ROM label for your install CDs. On this dialog select the Avery Standard 5824 CD Label and click the New Document button to open up a document ready for you to create your label.*

DVD burners are just starting to become available to the average developer. The initial DVDs hold 4.37GB of data, which is more than six times as much as a CD-R. Advances in the laser technology have some manufacturers looking at 27GB and others looking at more. This

technology is growing and improving at a rapid pace. Our fear is that what we write today will be obsolete before the book is even printed. If your data does not fit on a CD and your clients have DVD readers at their offices, this might be a solution to investigate.

One danger with DVDs you need to be aware of is, like so many other things in our industry, there are several standards. Unfortunately, like the battle of Betamax vs. VHS many years ago, we are not sure what standard will win out. So do you get a DVD burner that supports DVD+R/RW, DVD-R/RW, or DVD RAM? Which standard does your customer's computer support? This technology might be a bit immature to consider, but maybe not before long.

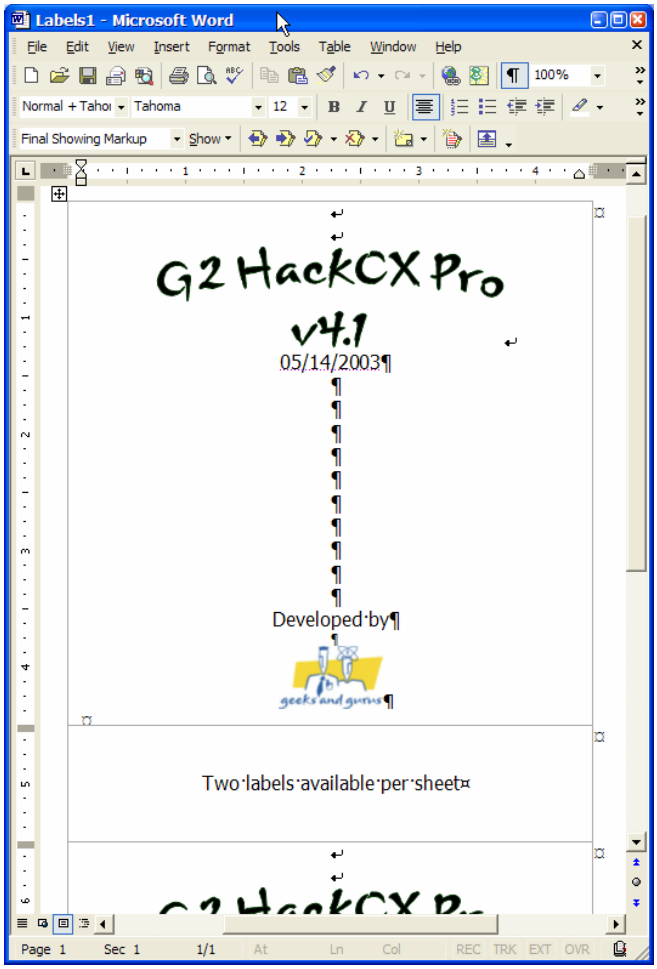

Figure 7. *The Microsoft Word Envelopes and Labels wizard creates a Word table so you can create two labels per sheet which is standard from label providers.*

E-mail

Packaging the install files and attaching them to an e-mail is a quick method to get a release to a client. Until recently, this was our most common mechanism for delivery. We are sure this is obvious to most developers, but using compression is important when utilizing this mechanism for delivery. It seems in this time of broadband access to the Internet, many people think nothing of sending attachments with megabytes of data. This is great when you are in the office or at home with a DSL or cable connection, but it is terrible when you are on vacation in Hawaii hooked up with a dial-up connection that creeps along as if you were watching a replay running in slow motion.

Sending an executable or an update to the databases works great over e-mail. The clients detach the files we send, unzip them to the appropriate test folder, run the application, and verify the results of the test plan (do we live in a perfect world or what). Once the tests are verified the same files can be detached to the production directory for all the users to use.

One of the problems we ran into with customers using e-mail is they do not know how to detach the attachments even if we send explicit instructions on how to accomplish this task. This is common with computer-phobic people. A common resolution to this is calling the customer on the phone and stepping them through the instructions. Another issue we ran into over the years is some e-mail systems in the corporate world limit the size of attachments on outgoing e-mails. This may set up a roadblock for corporate developers shipping releases to their customers. The same limitation has been seen with Internet Service Providers (ISP). Some firewalls also stop attachments with EXEs and DLLs. Some are even so intelligent they stop a ZIP file that contains an EXE or DLL.

FTP

We have worked with clients that have their own File Transfer Protocol (FTP) site. We copy the installation setup files to the server and the customer copies them down when it is convenient for them. We recommend the files are compressed to save time on the transfer. We also recommend getting a FTP client that can restart interrupted file transfers if you have unstable Internet connections and/or large files to send.

This process is faster and less expensive than burning a CD and mailing it. It does not face the same limitations as sending the files via e-mail. The files can be transferred at any time of the day so it can be done after hours when the customers are out of the office.

You do need to notify the users the files have been copied to their FTP site and are ready to be implemented. The need to communicate with the client is no different from any other mechanism. They need to be aware of the release. In this scenario they need to download the file, or depending on the configuration of the web server, copy it over the network.

Web site downloads

The Internet is obviously a terrific delivery mechanism in our industry. We rarely buy software in a brick and mortar store these days. What we are often doing is downloading it from a Web site (**Figure 8**). If we can do this, why can't our clients? While we can say we usually push the releases to our customers, we can place the update on our super secret Web site and have our customers decide when they want to pull the release. They can also do it at their convenience.

One of the nice advantages of allowing them to download the file is they decide when they want to take the hit on their bandwidth. If you send it in e-mail, they take the download hit just after you send it and they open their e-mail client. This can be a problem if it takes 30 minutes when they are waiting for an e-mail order from one of their customers. Web download is an ideal transfer mechanism for vertical market apps as well because it reduces the distribution costs of duplication and packaging. Web server security also gives you a chance to have clients log in and provide the transfers over a secured socket layer.

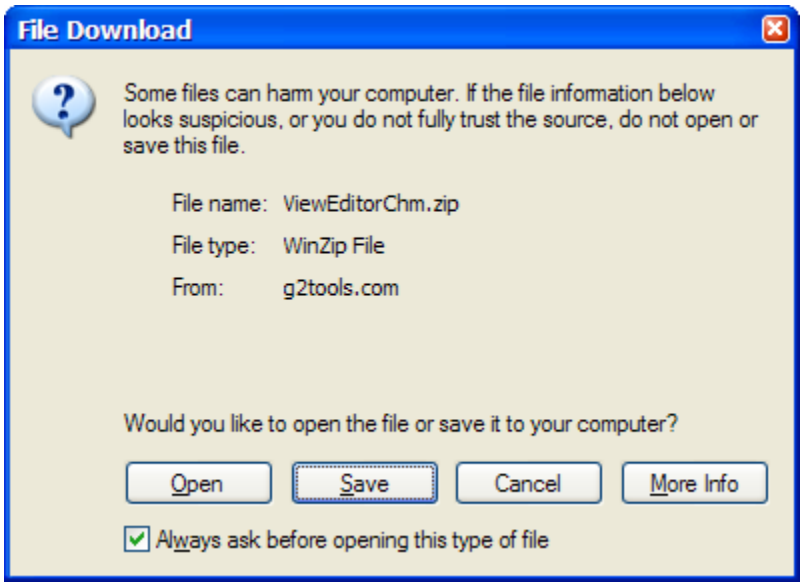

Figure 8. *Downloading files from the Internet via the Internet browser has become commonplace.*

There are drawbacks to this mechanism as well. If you do not have a logging feature on the Web site, you do not know if the customer came to get the installation. If several clients are running the software and they do not download the new version you could have them running different versions, which means you need to support the various versions available. This is not unique to this mechanism of distribution, but it is a concern. If you do not have security implemented or your Internet service provider does not give you this capability, users or any other surfers could potentially get the installation files. We recommend some sort of password access to the files. This access could be to the download directory on the web server, or could be applied to a zip file they are downloading.

Symantec's pcAnywhere

If you have only one tool you can buy to support your customers, make it pcAnywhere from Symantec. For those not familiar with pcAnywhere, it is a communications application that lets you connect to another computer and run it remotely. There is a file transfer feature to allow you to deliver files needed in the implementation and subsequent support of your custom application. This tool is a must have for our company, to the point we buy it for a customer as

part of the deal for the custom application. It saves on on-site travel and provides immediate support to their problems.

The file transfer functionality of pcAnywhere (**Figure 9**) has technology called Speed Send. This technology does a byte for byte comparison and only sends the differences in the file. This works great when sending Visual FoxPro executables that have a small set of bug fixes or enhancements. We have seen executables around 8MBs ship in a minute or less over a slow dial-up connection and it works fabulously over high-speed Internet connections. It is almost as if you were running the computer directly.

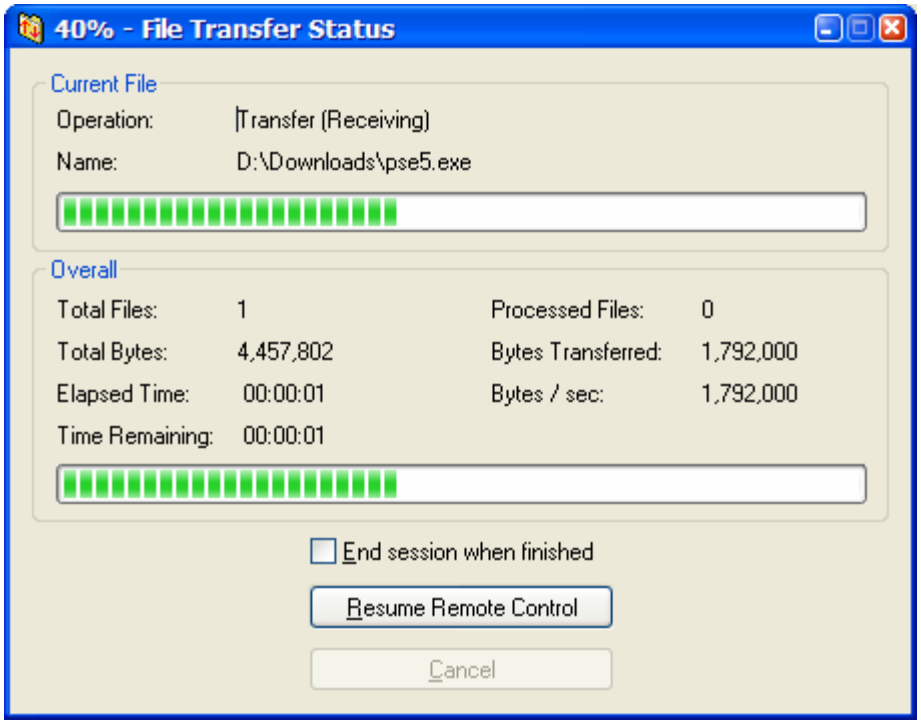

***Figure 9**. File transfer feature of pcAnywhere in action.*

There are issues with this technique of delivery and support. If you do not have a reliable connection and it keeps getting dropped (usually when less than 500 bytes of a multi-megabyte transfer remain) the customers tend to get cranky. We have customers that connect through a firewall and will not allow a pcAnywhere connection via TCP/IP. Those computers running a personal firewall like Zone Alarm or BlackIce are also a problem if they are not configured to allow access to the remote software. Sometimes we require a modem on the troubled systems. Some corporations have Security Departments that refuse to allow any sort of connection to computers at their facility. We have found this to be a problem, but in working with them and explaining the reasons for the package, we can usually find some compromise.

The biggest issue with pcAnywhere is it is single-threaded. This means you can only support or transfer files to one computer at a time. This is not an issue for a small customer,

but it does not work as well in an environment with numerous computers needing the application installed.

There are other remote application services also available. We have not used any of these, but they have become popular and are worth mentioning in case they might be of benefit to you. They are services you access via the Internet, and have fees associated with the service (usually monthly as well as annual plans). All of these have file transfer capabilities (some better than the others) in a similar fashion to pcAnywhere as well as the remote desktop capability so they can be used to deliver files and perform technical support remotely. A couple of them have the ability to manipulate the desktop from Personal Digital Assistants (PDA). They are not cheap, but they might be just what you need.

*Information and marketing details for Expertcity's GoToMyPC can be found at **www.gotomypc.com**, LapLink Everywhere's can be found at **www.laplink.com**, and I'm InTouch at **www.imintouch.net**.*

Terminal Services and Citrix

Citrix and Terminal Server (**Figure 10**) are products that allow several virtual computers to run on a Windows NT box. This allows something like a dumb terminal or a regular PC to connect and have all the applications run on the server. This connection can be established with all the conventional mechanisms like an internal network using standard network protocols, a modem, and via TCP/IP on the public Internet. This configuration allows companies to use outdated computers like an 80286 or better to run powerful applications developed for the current Windows OS platform.

The client computer has software loaded that makes the connection to the virtual computer on the server. This connection works similar to the pcAnywhere product discussed earlier. The big difference is the Citrix and Terminal Services product allows multiple connections to the "server" computer. Once connected, the session maps peripherals like printers and drives on the client computer. Having the local drive mapped as a drive on the "server" allows developers to transfer files to and from the server. Once the executable files are transferred, they can be installed. You can run the installation procedure or the customer can run it. Because the work is done on the server there is no network traffic generated other than the screen updates.

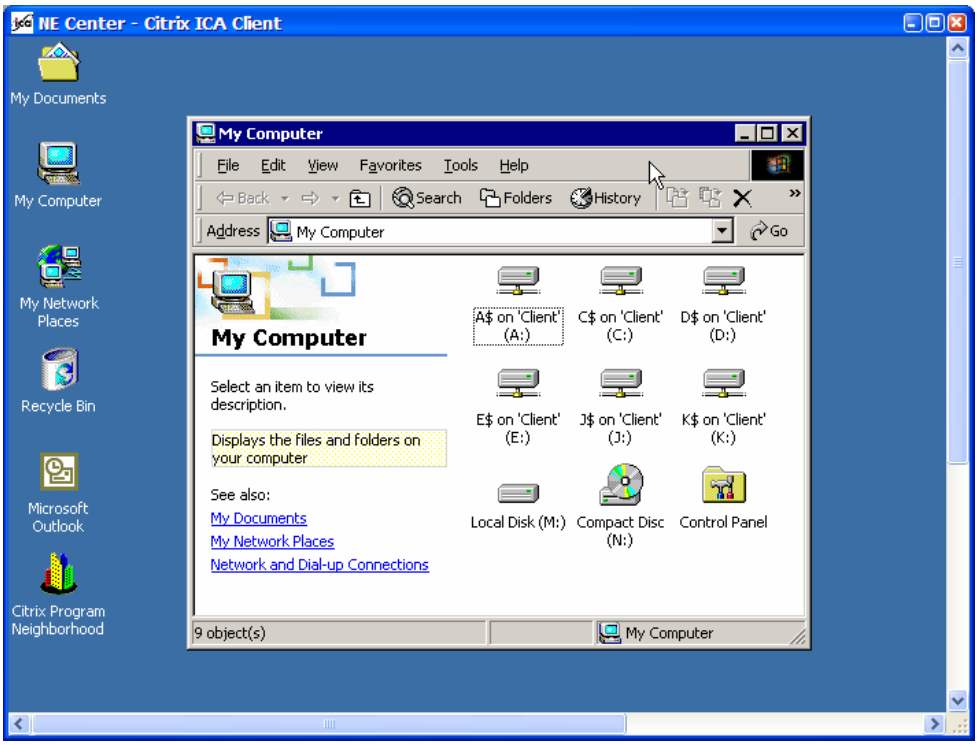

***Figure 10**. File transfer available in Terminal Services and Citrix (shown here) via the drives mapped to the local client computer in Windows Explorer.*

This situation is ideal when supporting customers that struggle with installation procedures. It also allows you to easily support them when they are not close to you geographically. This is simple to set up and works very reliably. Our client base does have customers that use Citrix, but this is not common.

Conclusion

This chapter detailed some of the considerations you should review when packaging an installation. We addressed issues like determining when it is ready to ship, what needs to be shipped to the clients, what features and schemes for deployment need to be packaged, what utilities need to be sent to the client to make support easier in the future, and how will the installation package be shipped to the customers. Now that you have prepared the packaging, you can move on to the next chapter where we discuss the tools available to create the installation setup and the features they support.

Updates and corrections to this chapter can be found on Hentzenwerke Publishing's Web site, **www.hentzenwerke.com**. Click 'Catalog' and navigate to the page for this book.