Issue Date: FoxTalk November 2000

Deployment: Installation Preparation

Richard A. Schummer
www.rickschummer.com

Late last year, we introduced a new series in *FoxTalk* that addresses various deployment issues. This month we return to the series, as Richard Schummer discusses the preparation that's needed to build the installation process.

The last of the features have been tested in the customer's super killer application. You and the rest of the developers have developed every last requested feature that can be squeezed into the application with the deadline approaching. All of the walkthroughs went off without a hitch. More importantly, the last of the Coke has been consumed, the fridge is empty, and the Thai food take-out place closed an hour ago, so the application must be ready to ship. You sit back in your chair and take a deep breath because the hard work is complete and the system is ready to be delivered. You have to put together an installation process to get the system loaded for the hundreds of users at the customer's site. What else have you forgotten? Does panic set in at this point, or are you a veteran of this process? Do you rush to the 24-hour convenience store to pick up more Coke and a case of microwave pizzas, or do you open the "Deployment Folder" and print out the company checklist of items to prepare for the creation of the installation process?

Whether releasing applications is brand-new to you or old hat, you know this can be a nerve-racking experience. I remember the first time one of my applications was going to be distributed to 300 sites across the country. I don't think I slept much the night before because I was worrying about all of the details. Will the batch file successfully copy all of the files? What happens if one of the DOS Copy commands bombs early in the process? What if the diskette duplicator messed up only 10 percent of the install diskettes? What if the mailing department or the mailman rubs a magnet on the package? Today it's slightly different because of the diverse technology and tools, but you still might lay awake wondering about all of the details. Who's going to burn the CD-ROMs? Will they do it right? Will the copies be solid? What files need to be deployed? How many copies are created? Where are the installations mailed? Do you need people onsite? Which installation utility do you use?

Hopefully most of these questions have been answered long before you reach this stage. The planning for this phase of deployment should begin right at the beginning, as you're collecting the requirements from the customer.

One last test

I've been harping quite a bit during this series that testing is important. At this stage, the application has been thoroughly tested by you and the testing partners. The one last test that I'm referring to is to run the application and most of its components one last time. During this test, I never add or alter data in any form. I never run any data destructive batch processes. This test just verifies that the executables that are cut for the installation disks aren't corrupt in any way. A quick test to verify that each form starts and each report runs can save you the phone call from the customer saying that the application doesn't function. First impressions are important in this business.

Distribution directory

The first step in preparing your installations is creating the distribution directory or directories. Installation tools, including the Setup Wizard that ships with Visual FoxPro, need a directory structure that's identical to the directory structure you want installed on the customer's site. Setup tools will replicate the directory structure that you create at this point. It's good practice to create a clean directory tree for the installation. A common mistake I've seen developers make is to use their source code or developer directory as the distribution directory. When this approach is taken, the final installation files will include every file in the directory—this means that the source will be shipped.

What files should be copied to this directory tree for distribution? This question is difficult to answer because it depends on your customer's needs, the type of application that's installed, the architecture of the system, the network scheme used on the customer's site, whether ActiveX files are used (including or not including data), whether source code is included, and whether Help files are being sent.

At this point, you need to determine which files are required at the customer's site. Items to consider are the executable, the application metadata (DBC, DCT, DCX, and Stonefield files), application configuration files (like INI or DBF files that hold startup information), language conversion or generic message tables, sound files, ReadMe text, external reports, and Help files.

Don't forget the "excluded" files in the project manager if the executable needs them at runtime. This is a frequent mistake because it's easy to forget. Excluded files aren't compiled into the EXE, APP, or DLL. One example of this is the technique of excluding report files so the customer can modify them. If you don't send them, the users won't be able to modify them—or run them—because they won't even have them! Another example is the graphic files used in the application. Many developers exclude these files because they're large, and excluding them reduces the size of the executable and the memory requirements needed to run the application. Yet you'll still need to include them when assembling the contents of your distribution directory.

The following directories are a typical structure and list of files that I ship with an application.

Root directory
The root directory contains the main application executable files as well as any free tables or INI files needed during the application startup. I also include a number of developer tools used in the maintenance of the application:

- Application executable (exe)
- Application COM objects (exe and/or dll)
- Application API files (like the FoxTools.fll)
- MsgSvc.dbf, fpt, cdx (use with Steve Black's MessageBox)
- Strings.dbf (use with Steve Black's INTL & Visual MaxFrame)
- CheckPK.exe (see the discussion of tools in the "Handy utilities..." section later in this article)
- ReGenCDX.exe (see the discussion of tools)
- AppInfoUpdater.exe (see the discussion of tools)
- Config.fpw (if used and not compiled in executable)
- ReadMe.txt (if used)

I usually define the next set of directories as sub-directories of the root directory.

Data directory
The data directory contains the database (dbc, dct, dcx), the application tables (dbf, fpt, cdx) contained in the database, free tables holding application information, and the Stonefield metadata tables (CoreMeta, SDTMeta, SDTUser, DBCXReg). This directory is optional because not all application installs need data, especially since updates or service packs and some applications don't have their own native (Visual FoxPro) data set at all. Alternatively, you might have more than one data directory, because your application might have, for example, a production data set and a tutorial data set.

System directory
This directory contains a number of system tables used for information like users, security rights, user preferences, states and ZIP codes, a report catalog, framework referential integrity files, spell-checking word tables, and FoxUser (dbf, fpt). The key differentiating factor between this and other directories is that there's only one of these for the application—regardless of how many data directories there might be. This directory is optional, as not all application installs need this information, especially updates or service packs.

Sounds and images directories
These directories contain any of the files needed by the application to produce a sound, or large image files displayed on the forms or in reports. One file I prefer to exclude from the application is the company logo. Corporations are known to have marketing departments that shift the company image by changing logos. This way, I can dump in a new image file with very little work. These directories are optional for service packs as well.

Other directories
Your directory needs might vary depending on the framework that you've developed or purchased. You might want to include a separate directory for system metadata, links to a SQL database, Microsoft Office automation documents for Word or Excel, or HTML Help files.

Installation schemes
After the directories are created and the correct files are placed in them, you'll need to determine what installation scheme is needed for the release. There are numerous schemes to create an installation process. I've found the following ideas successful for Visual FoxPro applications. You might combine several installs into one package, or you might ship separate installs based on the customer's environment or needs. I've separated the upcoming discussion based on the different options that you can include in an installation package.

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

This install will typically be used in a network environment. The installation will load the core application files on the file server in a specified directory. Once the files are loaded, the users will have access to the application provided that they have network access to these files. Using this scheme also requires that all of the workstations have the Workstation Install (discussed in the next section) loaded so they'll have the VFP runtimes.

This installation will also work on a client PC for a single-user application provided that the files from the Workstation Install and the Data Install are also installed. Many developers who have "mom and pop business" clients use this technique all the time.

Workstation Install

When the application is loaded on the file server, is it ready to be executed by the connected workstations that have security access? Not necessarily. Each workstation needs additional files loaded. These include the VFP 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. I've broken the Workstation Install into five installs, all included in separate directories on one CD-ROM. You might find there to be more or less than this depending on your needs.

The first install is the VFP runtimes. I find that application startup performance is best when the VFP runtimes are loaded on each computer. Do the runtimes really have to be on the client workstation? No. You can use the –R directive with the VFP.exe (and, consequently, your custom executable), but loading the runtimes on the user's computer is faster because you aren't pulling 4M down the pipe every time you start the app. This install will load all of the VFP runtimes, including VFP6r.dll and VFP6rxxx.dll (where xxx is the language of the VFP version you have; that is, enu for English).

The second install is the Multi-threaded Runtimes. This loads the new VFP6t.dll file for multi-threaded COM objects. Not all of the applications require the multi-threaded functionality, so this install for our customers is rarely needed.

The third install is for the 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 won't need this installation.

The fourth install is for the VFP ODBC driver (and others if needed). This loads the VFPODBC.dll and VFPODBC.txt files. This gives your users a way to analyze their data via tools like a spreadsheet, do mail merges from a word processor, or build their own queries via an end-user database or tool.

The fifth install is for 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'll need the ActiveX controls loaded on the computer that the installation is being built on. It's important to note that the ActiveX controls loaded during an install can be the ones included with VFP and Visual Studio as well as any third-party controls purchased.

All of these installs are copied to one CD-ROM but in separate directories. You need to do this because the install tool name files are identical for each install. You might decide to customize this CD as well for a specific customer. It might 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 CD (or copies of it) can be passed around from computer to computer. I also recommend that the CD be dated, and note that the VFP version and VFP service pack runtimes apply. It sure can be embarrassing to have that new VFP 6 Service Pack 3 Session class that's required by the new executable ready but unavailable because the runtimes on the machine are for a prior version.

Data Install

Obviously, this installation section is for the application data. There are additional questions that might complicate this seemingly easy setup. What files need to be sent? Is this VFP local data, or are you using a SQL back-end 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?

The initial installation will require that all of the application data be installed, and this data can be found in the installation data directory. I like to keep this installation separate, especially for a new service pack release. Vertical market applications will like this scheme as well, as 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 might mirror a File Server Install scheme in many ways. There are, however, many differences that your application might encounter. You'll likely be installing the Multi-threaded runtimes for scalability, COM components or an EXE, HTML files, and making Web Server settings (via executable or another manual setting) like scripting files, user security, and the mapping of drives to the data.

Packaging the install

Now that you've developed schemes for the installation, you need to determine how you'll package it. I'm not referring to the box in which the CD is delivered. The marketing experts handle this. I'm suggesting that you need to think out what previously discussed installs need to be packaged up and sent to the customer.

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

The packaging of the install is as important as developing the perfect software because it's likely to be the first impression that most users have of the application in production. I'm not talking about the people you developed the software specs and performed acceptance testing with; I'm talking about the possible hundreds of end users who actually get the package loaded on their computers.

Handy utilities to ship with the installation

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's likely to occur. The following are concepts for the tools that have been developed for the types of applications delivered to your 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 release is made and the latest alterations to the database tables, indexes, views, and relations need to be implemented? You could write a custom program each and every time that makes 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 that 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. I don't want this to become a commercial for the Stonefield Database Toolkit (SDT), but I find so much value in this product that I 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 what the structures are 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 that new columns can be added or removed, column name changes are applied, and code pages can be changed. The same is true for indexes. It will also create new tables if they don't exist. This is all handled based on using the DBCX extensions to the database via the Stonefield Database Toolkit or your own developed tool (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 my very first release with this product <g>). SDT can be used in initial installations to completely generate all of the tables as well.

There's another option to solve the database changes, and that's 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. I hear of this being done all the time. I'm just not one who trusts myself to remember to make the changes in the same fashion as I did in development. I'm 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 that SDT doesn't handle that you'll 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'll 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 need a separate program that cleans up the data before implementing a new field or row-level rule for a table.

GenDBC/GenDBCX

If you aren't using SDT and/or want a mechanism to generate the database and all of the table structures, views, indexes, and relations, take a look at GenDBC (included with each release of VFP) or GenDBCX. GenDBCX is a third-party tool written by Steve Arnott, and it's available for free. It can be downloaded from www.sfinsider.com. Both of these tools generate VFP program code that will build the database from scratch. Just like the SDT Update process, neither of these tools populates the tables, so you'll need a mechanism to accomplish this task.

Checking next id table

Developers who use surrogate keys (meaningless integers or characters that uniquely identify a record in a table) will have a table that contains the next key for tables. Periodically these tables will get misaligned with the real data in the tables. This can happen because the developer writes bugs in the application, tables get zapped moving from development to production without updating the next surrogate key table, incorrect referential integrity rules are introduced, or the planets are out of alignment.

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

```
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. It's a good idea to periodically run this process for all of the tables in the application. One red flag that indicates it might be time to give this process a run is constant calls from a customer saying that they can't add any records into any form because they're getting a message indicating that keys contain duplicate values.

Configuration/control table updater

Many applications have an INI file or a configuration table. When new options are added, a mechanism to get these options into an existing application needs to be considered. Personally, I prefer to work with tables because I work with these all day and Visual FoxPro provides plenty of commands to manipulate data. Adding new options to a configuration table is as simple as an APPEND FROM or running code to do INSERT INTO. Updates are as simple as a LOCATE or SEEK and using REPLACEs. I can also write code that does a SQL Update. The INI text files are also easy to work with, since VFP 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 that you develop some mechanism to update this information so the customer application doesn't malfunction when new options or features are added.

The next step

Next time I'll build the installation package. Tackling the Visual FoxPro Setup wizard isn't a daunting task, but I'll discuss some of the advantages, pitfalls, and limitations. After this, I'll discuss the actual deployment process at the customer's site. Feel free to send me your ideas about deployment and I'll pass them along as the series continues.