

Deployment in the Real World

Session Number

Richard A. Schummer

President

White Light Computing, Inc.

42759 Flis Dr.

Sterling Heights, MI 48314

Voice: 586.254.2530

Fax: 586.254.2539

E-mails: raschummer@whitelightcomputing.com

rick@rickschummer.com

Web sites: www.whitelightcomputing.com

www.rickschummer.com

Overview

Do you regularly lose sleep the night before the big release? Have you struggled with the new InstallShield Express – Visual FoxPro Limited Edition deployment package? Have you wrestled with the older VFP Setup Wizard which left you begging for a DOS batch file and the XCOPY command?

This session will discuss many deployment issues including preparing the customer, preparing the development staff, items to consider when preparing a release, check lists, mechanisms to

deploy your custom applications, building the setup files, some tips with the VFP deployment tools, a quick demonstration of one or two other commercial tools for deploying your applications, and things to consider once a release is shipped and successfully deployed.

Session Attendees will learn how to...

1. Plan for deployments from the very start and create a successful strategy for deploying applications.
2. Determine when an application is ready to ship.
3. Develop a set of checklists to make sure deployments go smooth.
4. Develop different deployments for different applications (desktop vs. web) and different environments (file server vs. client-server).
5. Determine what tools to ship with your applications for support.
6. Deploy applications via CD-ROM, email, PcAnywhere, Web site download, web distribution, and Terminal Services or Citrix.
7. Deciding which deployment tool is right for you (InstallShield Express, Wise Installation System, PC-Install, ActiveDelivery, InstaFox, others).
8. Work with the development staff and customers during deployment.
9. Work through post-implementation issues including customer follow-up, deployment post-mortems, bug tracking, developer reviews, and deployment celebration events.

Skill Level/Prerequisite

Introductory to Experienced (there will be something for all). The only prerequisite is the desire to successfully deploy applications.

Introduction

Deployment is the end result of a completed development cycle (requirements, design, develop, and test). The product that is deployed can be a component of a large enterprise wide application, a quick and dirty developer tool or a layer of an n-tier application. It can be a database conversion, a small enhancement or bug fix to an existing application or it can be an all-new application. In reality, it can be anything one person or a team of more than one developer assembles for a customer. It may take 30 minutes based on fixing a bug, or may take a year or more for new system development. It could even be one phase of a many phase implementation that is scheduled over a period of time.

Deployment is not something that should first be considered after the last of the code is developed and tested. It is a process that needs to be mapped out early in the development life cycle. There are a number of issues that should be addressed to eliminate the number of surprises that affect a successful deployment of an application.

So what happens when the customer has just signed the contract in blood? This could be the contract of a lifetime, the project you always wanted to work on. This might be the most complicated piece of software craftsmanship you have been challenged to accomplish. It seems like an eternity until it will be implemented, after all the entire development cycle is 6 months long! So should you be worried about deployment today? Will you even be responsible for deployment, or can you delegate this part of the project to the customer's MIS department? The

time to start thinking about application deployment is in the beginning. This white paper is going to cover the different steps, options, gotchas, and issues faced when deploying the application you (and possibly your team) have worked so hard to get ready for your customer.

The material covered in this white paper is based on the many application releases I have made in the last 17 years. This experience ranges from applications on “big iron” to single machine applications with one form, a report, and a set of labels. I have applications running in hundreds of sites throughout North America and South America that I developed for General Motors, its subsidiaries, and dozens of smaller customers. Today the company I work for distributes applications on a slightly smaller scale than GM, yet over all this time the issues remain the same, only the technology to implement the solutions seems to change. Even with this experience under my belt, I am sure there are issues out there that I have not experienced. I’m hoping for plenty of feedback from the attendees so we can benefit from each other’s successes and tragedies.

Getting Ready

There is much preparation for deploying a release no matter what the size of the application. This section will cover some of the details that should be considered throughout the development lifecycle (contract signing to the day before deployment).

Preparing Technically

Having a list of what items need to be addressed before a release is readied is critical to the success of the implementation. This list includes all the preparation outside of compiling the code and building the install process. This list is not meant to be inclusive for all releases. What items are needed will depend on the items that are shipping.

Images

There are several images that can make an application look that much more polished. I typically leave these for last (or near the end) since most of the effort of development should be directed toward solving the business problem. The most obvious images are the application icon (remember the 16x16 & 32x32 [16 and 256 color] sources), splash screen, about window, and wizard images.

There are many sources from which you can purchase images. I usually utilize Icons (.ico) and JPEGs (.jpg). I like the JPEGs over bitmaps for two reasons. The first is the size of the images, JPEGs are compressed and bitmaps are huge. The other reason is that the JPEGs are web ready so the images are reusable for Web sites or a Web interface to the application data. The key to purchasing images is that you have the license or right to distribute them. There are plenty of image editors out there if you want to create your own. I occasionally still use IMGEDIT.EXE that came with VFP 3.0 to create icons. Recently I came across a free tool called Icon Studio (available from www.html-helper.com).

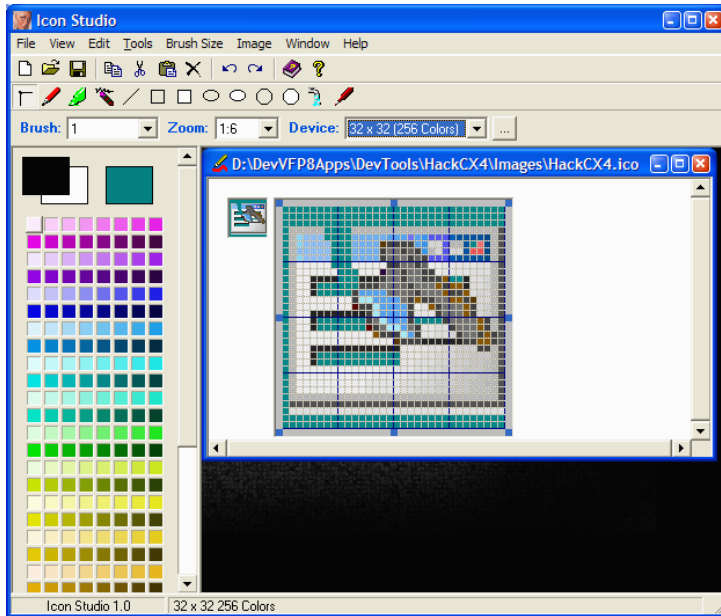


Figure 1. Icon Studio is an excellent tool to create icons.

The reason I like Icon Studio better than ImgEdit is that it supports 32x32 at 256 colors, which is the quality of icons used in Windows XP. The Microsoft Image Editor (comes with several Microsoft packages including Visual Studio) works satisfactorily for JPEGs and GIFs, but I rarely create these images and mostly use it to convert images from one format to another.

EXE Version Details

The release of Visual FoxPro 5.0 introduced internal EXE version information. The version information includes application version number, and text for comments, the company name, a file description, legal copyrights and trademarks, a product name, and language id. This information is entered through the EXE Version dialog (**Figure 2**) or through the new Project Object Version properties. The minimum properties to include in the executable are the version number, company, copyright, and product name.

This information can be accessed via the `AGETFILEVERSION()` function in Visual FoxPro. If you are using VFP 5.0 or earlier you will need to use the `GetFileVersion()` method included in `FOXTOOLS.FLL`.



Figure 2. The EXE Version Dialog

Data Conversion

Update releases are common. Changes to the applications often require the database to change. I have successfully used the Update() method in the Stonefield Database Toolkit (SDT) to handle these structure changes. There simply is no other third-party tool as important to a Visual FoxPro developer as SDT, and this is the biggest reason to buy it. There are still cases when the new structures need to be populated or data converted from different fields or from different tables. These conversion routines need to be developed and tested.

Help

The Help file seems to be the part of development that 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 or HLP style). A bug fix release may only need an update to the documentation, while new development may require all new help to be written. Occasionally clients avoid it too since they are so close to the application. I know many custom applications never see Help since they are developed for a small user base that is already familiar with the business at hand.

Training Materials

Training materials may be required for broad released applications or vertical market applications. Many customers we have worked with in the past write and develop the training materials for their applications. We like this concept since it gives them ownership in the process. It also saves them plenty of cash and allows the development teams to concentrate on what they do best.

Tech Support

Not every company has a staff dedicated to Technical Support, but all applications are supported in one shape or form either by the person who wrote the application, by other developers or by the support staff. The easiest thing to keep track of is a list of Frequently Asked Questions (FAQs). This allows the developer(s) or technical support staff to quickly learn about the most common issues that are handled from the customer. This list of questions will be developed during beta testing and added to after the application is in production. This list can be included in the user documentation, in the README.TXT file, or on the support web site. Larger companies like Microsoft have created a KnowledgeBase of white papers that are based on issues related to the different applications that they wrote. Who is to say that even the one-person shop cannot leverage these concepts?

Duplication

Once the software is ready to ship you have to employ a mechanism to get it to the sites for deployment. Will you have the users download the setup routine from the company Web site? Will you burn a set of CDs and mail them through the postal service or through one of the many overnight carriers?

The introduction and vast acceptance of the Internet has eliminated some of the duplication for software deployment. Skip this detail if you are going to distribute via a web site.

Are the CD burners ready to roll? They are so cheap these days that there is very little excuse not to have one. They save time if you are doing a mass installation. I really dislike doing floppy installs since it takes so much more time. There are service providers that will duplicate CDs if you do not have access to a CD burner or have a mass distribution (anything over 100 CDs may be worth the cost). There are several CD label makers that will add that last minute polish to the distribution. Whether you do floppy or CD installations, make sure you have enough media on hand to cut the installations.

Users

Our experience has proven that the clients/customers we develop the application for are typically not the ones who are going to be using the product. It is important to keep information flowing to the actual user base to keep them updated of the upcoming release.

Communication allows them to get training scheduled, have the equipment installed, cut the check to pay for the package, and schedule the parade in your honor for making life easy in the business world. Seriously, there can be a lot of work preparing the marketing literature, changing office procedures, and updating web sites. Make sure you are communicating with the user community, either directly or through your customer contacts. Make sure to track email, phone calls, and coordinate possible visit(s) to their site(s) if needed.

Hardware

How many times have you shown up with the CD (or diskettes, tapes, zip disks) to find out that the special label printer they need for the application was never ordered? Have you shown up with a professional looking CD just to find out no machine in the office has a CD player because

the boss does not want them used to play music in the office? Many custom apps need new hardware. Whether it is the latest in Pentium technology or the simple fact of dumping the dot matrix printers for a 32 page per minute laser printer, many applications have special hardware needs. Verification that needed hardware is delivered ahead of the application can save some embarrassment in the delivery of your new solution.

Determining Ship Readiness

As much as I would love to start the session by discussing shipping the product, we need to step back in time to determine when shipping is determined. What exactly determines when a product is ready? Over the years I have tried to tell my customers that shipping happens when they say the product is ready. If I followed through with this philosophy, each release would happen when I demonstrated the first version of the prototype. A good tip to point out here is to intentionally compile in a GPF/fatal error of some sort to show during the demo. I do this just so they have a chance to see the prototype is not quite production ready and it is a nice way to show off the error handling features.

It is also important to note exactly what a shipping product can be defined as so we all understand what goes out the door when we “ship”. A shipping product is the end result of a completed development cycle (requirements, design, develop, test, production).

In reality there are many things that help us focus when determining when a product is ready for primetime. The focus should be to predetermine what the customer requirements are and what the release criteria is. The negotiations to determine the release criteria should happen very early in the development cycle.

Delivery Dates

Most software development projects are driven by delivery dates. How many serious Y2K projects had delivery requirements of December 31, 2002? There are real business rules that drive delivery dates. I have never had a customer say, “I have no date that I need it by so there is no problem getting this system done five years from now”. Usually it is something like “I need this as soon as possible (ASAP)”, or “I needed this last week”. I always remind the customers that the only dates that are acceptable are ones that are in the future and ones that validate in their applications. For instance, “ASAP” cannot be entered into a date field. I also note that initial dates are target dates and never chiseled in marble. Target dates are a way to focus the development team to complete a project. Target dates are also used to manage the scope of a product delivery.

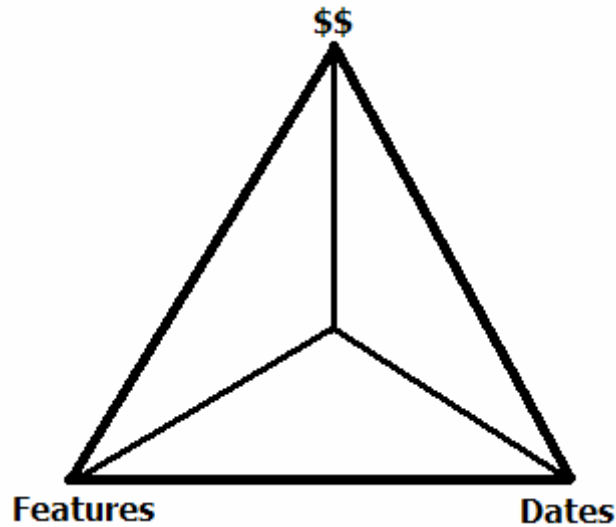


Figure 3. *The development triangle*

Often the customers want to dictate all the factors used to determine when a release can be made. One approach uses three factors: features, time (also translated to money), and delivery dates. Software developers can let the customer select and determine two of the factors first, then calculate or determine the third factor. For instance, if the customer determines that they need 50 features delivered in the next two weeks, the developer has the right to charge a lot of money because they will have to pay a number of developers to work on the project. If the customer has a restricted budget and needs it in the next month, the developer can restrict the number of features that will be deployed.

Requirements

I won't bore you with all the nit and grit details of what follows an initial customer meeting, but it boils down to two different scenarios in my experience. The first is what I call the "Documented Approach"; the other is the "Pressure Cooker Approach". I'm sure there are others that you have experienced, but when boiled down, they can fit one of these classifications or a mix of the two. The "Documented Approach" is where you write some sort of functional specification (also known as a requirement document), and/or include a prototype as the requirements are collected. The developer(s) get a pretty good idea of the scope of the project or projects. Estimates are created based on the requirements (usually overly optimistic) and a ballpark cost (hourly or fixed price) is completed.

The second approach is the "Pressure Cooker Approach". This approach is also known as "Requirements-to-Shipping Approach". This is where the customer needs to have something yesterday, or more likely a month ago. There are many causes to this scenario. The customer might have some brand new competitive pressure. They may see a brand new opportunity with a legitimate small window of opportunity. They may have just found a Y2K issue with their current software (yes, it happens). In my experience it is none of the above, but more likely some pointy hair boss who delayed the project because he has budget concerns that reflect on his quarterly bonus, then tells the software folks it is time to get cranking at the last minute. Whatever the reason, you are on the hot seat to get something in production. It is time to change into the superhero outfit of your choice and deliver the software in world record time. This

scenario is played over and over and over in software shops around the world. There are never any estimates more than a month, because this is when the application is due. Typically you get the first version of the app running, moved to production, and then allowed to step back and take the “Documented Approach” for the rest of the system. At least I would like to believe that I have a chance at the “Documented Approach” for the second phase. Sometimes the projects are in the “Pressure Cooker” for all phases.

I cannot stress the importance of written specifications for a project’s success. Contrary to popular belief, specifications on the back of a napkin are not reliably transferred into software. Please use the napkin notes jotted down at lunch or dinner when you write the real specifications. The specifications have to be written down in some form and the form needs to be consistent across all the company’s projects. This allows developers to transfer from one project to another and eases new developers into the company as the company grows. Without some specification there is no way to prove that the final product does what it was set out to do. It also helps developers taking over an existing project to know how the product is designed to work. I find very few customers actually read the functional specifications word for word, but it gives me a place to refer them as they are testing the application so they have my perspective of their business.

You as a developer are responsible to ascertain which requirements are real requirements and what is the nice-to-have. One observation I have made recently is as time moves forward more and more projects are subject to the “Pressure Cooker Approach”. It may have to do with the fact that we live on Internet time. It may be that people have poor planning capabilities. It may be that we develop better each time and the customers have higher expectations for our services. Whatever the reason, my experience proves time and time again that the “Pressure Cooker Approach” fails more than it succeeds. One fact of life is that designing applications consumes time, but it is time well spent. Many customers don’t want to make this investment, a mistake I don’t want to them to make.

A wise developer some where once said, “I can walk on water and deliver applications on time that meet all the requirements in the specification, as long as they are both frozen”.

Feature Set(s)

Once the requirements are collected they must be reviewed and organized into feature sets. This is a group of requirements that need to be implemented together. The feature set is usually the most critical path issue involved with shipping a product. The customer needs to clearly state what is needed in this version of the application. These functions are usually business driven. They usually revolve around some pain the users are feeling. This pain usually leads to statements like, “I need to track these thing-a-ma-gings and get a grip on costs and I have to have this implemented by the end of the month”, so make sure that they understand that delivery dates are in reality a feature.

This is where the negotiation starts between the developer(s) and the customer. Which functions can be delivered by what dates? Must all the requirements be delivered at the same time? Can certain features be delivered first? Can others wait until version 1.1, 1.2, 2.0, 7.0 (always skip versions so you can send the junior developers looking for them later)? Each feature set included in the version that ships next should be determined as early in the development cycle as possible.

There will always be scope creep in development so leave room to negotiate extensions to time frames later if needed. I have found success in these situations when I have the customer decide what can be taken out of the release to make room in the project schedule for the new hot features they want. If they have to have it all (and when is this not the case), then the ship date has to slip to a new ship date. If the date cannot slip, then hopefully you have a bonus provision pre-negotiated to cover the extra time you (and possibly your team) are going to be putting in after normal business hours.

Prototypes

Developing a prototype of the application during development allows the customer to see features they desire translated into a working application. At the same time I have written developer tools that reverse-engineer the prototype to develop the functional specification. These tools scan the forms, reports, menus, class definitions, and quickly format information that is dumped into a word processor. This text is later formatted by macros to make the information readable for the customers and the development team. Add in a business rule here and there and presto, writing a functional specification is not as painful as it was in the past. This also saves my team time and allows them to concentrate on the “fun” part of development.

When is it Ready?

There are three perspectives of when the product is complete and ready for deployment: the engineering viewpoint, the quality assurance check, and the customer determination that it meets 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. It should be obvious that 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. This proves that the interface supports the business rules, which are implemented in data and that they all work together to manage the information correctly. Further, all company guidelines and standards have been implemented in the code that was 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 (or as Whil Hentzen calls them in the *1999 Developer’s Guide*, “Defending Your Life”), or a simple desk check by one other developer. The goal of these review sessions is to make sure the code is matching 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 perspective is almost self-explanatory. A quality assurance team strictly enforces that the customer’s requirements are implemented correctly. I think the real issue is how many software shops really have a staff dedicated to quality assurance. I am not talking about 100 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 likely will not be 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 the customers are usually not trained in finding bugs in software. The best you can hope for is that they are business experts and will 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). I always suggest you watch them struggle to add a new thing-a-ma-thing 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.

Test, Test, Test

I know I am preaching to the choir here, but testing is so important. A Test Plan is always desirable and rarely implemented. This section can only begin to scratch the testing surface since entire books are dedicated to the art of testing. Here are a couple of rules I make my development staff check and do before a new release:

Forms

There are some straight-forward steps to test that a form is working correctly. Not all of these items will apply to all forms. These steps are not taking into account testing the business logic in the specification; these are items to test that are generic in nature. Making sure that the list is covered for each form in the application has saved us from looking unprofessional.

Verify that the form's tab order is correct. This test is one that needs to be keyboard centric. When we find tab order problems during testing it is usually a developer that is a wizard with a mouse. During this testing you might want to make sure objects that should not be in the tab order are removed by setting the *TabStop* property.

Invoke the add/edit mode and modify data in data bound objects. When adding records make sure the default values are properly set. Developers have different ways to change the mode of a form from non-edited to edit. One approach is to require the users to press the "edit" button that changes the objects from read only mode to be editable. The other approach is to auto-sense that the data has been changed. This testing also verifies that the enabling and disabling of various toolbar buttons (whether on the form or in a real toolbar) are handled correctly. Always check that the referential integrity rules are also properly enforced.

Save/Cancel all changes to all data bound fields. Making sure this works sounds obvious, but there can be changes to the underlying views that can break the saving of data from a form. We

have run into this more than a few dozen times when another developer accidentally changed the *SendUpdates* property on a view to *.F.* and forms stopped working correctly.

Invoke the delete capability and verify that the appropriate messaging occurs asking whether the user wants to continue with this destructive operation. If there are associated referential integrity rules in effect, make sure these are enforced.

Use different logins to validate security for the form if applicable. Security can be implemented at various levels. Is the menu option for the form enabled or even displayed for the various security levels? Some security implementations will display the form, but the data is read only, and others will disable only certain objects or toggle visibility. Hopefully the form security implementation is well documented so it can be well tested. This testing is very important since users can get agitated when sensitive information is seen by the wrong people.

Verify all incremental searches work correctly. Incremental searches are very common and often depend on the appropriate indexes being created for the table or cursor. Just like the save/cancel testing, it is something that can be broken at the table level or metadata level.

Push all the command buttons on the form. Command buttons are used to invoke some functionality and this functionality needs to be tested. Make sure the functionality performs to specifications. If using toolbars that interact with the form, make sure to test all the toolbar buttons as well.

Invoke sort and filtering functionality. Many commercial and custom frameworks use metadata to drive this functionality. Make sure that the form refreshes lists and grids that reflect the new sort order or filter condition. If the natural language of the filter condition is displayed make sure it is correct. Sort orders are often indicated visually with an indicator or colorization of a header.

Invoke all child forms to make sure they are included in the application. Child forms are forms that have data related to the parent or calling form, but are accessed from the calling form, not from a menu or switchboard interface. Forms are often called through indirection and can be left out of executables. Invoking all the child forms will alert developers to include them in the application project.

Invoke all delayed instantiation objects. A technique to improve performance of the form is to delay the instantiation of objects on pages of a pageframe that are not often looked at by the user. The first time the page is activated it instantiates the interface object for that page. If there is a problem with this process it will not be obvious unless that page is accessed during testing.

Right-click on everything, try all shortcuts on the form. Right-clicking can integrate context sensitive help, initiate a process, or provide a shortcut menu with additional functionality. Making sure all objects on the form that are suppose to have this functionality is natural, but one should also make sure objects that should not have these features is also important.

Verify all list objects have proper data and sorting. Testing includes validating the proper data in the list, that it is sorted in the proper order, and that columns are displayed with the correct widths so all the information can be read. This is another item that can be broken by changes to the underlying data and metadata. If the list is dependent on a view or an index order, the list can get populated incorrectly if someone makes a change to the view, index or table.

Press the F1 key to make sure it brings up the context help. This is not just for the form, but tab to each object and press F1 if field level help has been implemented. If the form has What's This

help make sure that the appropriate text is displayed. This is a great way to also test out the content of the help and to make sure the section is written. Often help is developed by someone other than the developer and this disconnect can lead to implementation of help not working properly.

Verify all abbreviations used on the form meet industry, customer, and development standards. Inconsistent abbreviations are confusing and often lead to technical support calls. Verify tool tips are appropriate and used only when needed. Implementing a tool tip on every object might not be necessary. Validate that all picture based command buttons and checkboxes have tool tips.

Verify that there are no hotkey conflicts between objects on the form, and between objects on the form and the system menu (or form menu if you are testing a top-level form). This is actually a very common occurrence in our experience. It is also something that is often overlooked by those developers that are mouse wizards.

Verify all formatting is correct. This can be testing the object's *Format* and *InputMask* properties, as well as the general layout and format of the form. Make sure that the fonts used on the form are consistent with the other forms throughout the application. Align all the objects on the form so it looks professional. The alignment tools included in the Form and Class Designer in Visual FoxPro are excellent. There is no reason to have objects that are misaligned. Check all label captions and tool tips for correct spelling. If your users are like our users, they spend more time picking on misaligned fields and spelling mistakes on the form than a buggy validation process. Save yourself some aggravation and catch these before the users do.

Validate that the form graphics are correct and being displayed. Icon and image files that are not included in the project must be distributed separately and need to be verified that this actually happens.

Verify all Stonefield Database Toolkit (SDT) metadata is set up clean and validated for cursors. This is not something everyone is going to have, but it is a popular tool. All the commercial frameworks are integrating with SDT. Several of them use the metadata for indexing, captions, tool tips, filtering, and sorting.

Reports and Labels

There are some straight-forward steps to verify a report is working correctly. Not all of these items will apply to all reports. These steps are not taking into account testing the business logic in the specification; these are items to test that are generic in nature.

Make sure the correct data is printed on the report. This might sound like a ridiculous concept, but we have run across numerous test cycles with the customer where the data was not correct. One of the common problems is that the developer started with a Quick Report and it included a view in the report dataenvironment. Initial unit testing shows the report with the correct order and fundamental information. If a different view is used for the report, or more commonly, different SQL-Select code was used to prepare a cursor for the report, the view in the dataenvironment will still be used for the report. The view in this case needs to be removed from the report dataenvironment.

Test to make sure the data is correctly sorted and the report grouping is correctly bundling data together. One immediately obvious problem is that group headers and footers are printing for

each record processed. That is a good indication that the report cursor is not sorted correctly or that the groupings are not correctly ordered in the report.

Validate report grouping functionality works as specified. Test summary fields are calculated correctly and that the calculations are reset to the appropriate grouping. One common defect we have fixed over the years is a summation that is created for group footer and then copied down to the summary band, but still resets to zero each time a group break occurs. Adding up the numbers by hand to validate summations and counting the number of records for count calculations is important. Also, run the report with SUMMARY clause for reports that have this capability exposed in the report to validate that it works. Validate that the title and summary pages are printed and have separate pages if specified.

Test all Print When conditions function properly. Our experience has shown that developers write expressions for report Print When condition the same as menu SkipFor clauses. These are opposite logic and need to be verified accordingly.

Line objects that connect correctly in the Report Designer do not always connect correctly in the preview mode or when printed depending on the printer driver and video drivers. Most of the time they are fine, but our experience has not always found success in this regard. Make sure to test this on a number of printers, especially the ones that the customer uses.

If you are testing a label, test it on the actual label it was designed to print on. There are a number of predefined labels in the Visual FoxPro templates that do not exactly match the specification of the Avery labels that they were designed to print.

Verify reports to different printers if possible. Definitely test the reports on printers of the targeted customer. This might not be a simple task for a vertical market application. One thing we have found that saves us tech support calls is to test on one laser printer and one ink-jet color printer. These two types of printers have different unprintable areas around the margins. It also verifies that we do not have the problem of the hard coded printer information being stored in the report metadata file (FRX). See section “How to avoid hard coded printer problems”, page 523 of Hentzenwerke Publishing’s *1001 Things You Wanted to Know About Visual FoxPro* for more details on how to deal with this problem.

Make sure graphics are printed both in color and on a black and white laser printer. This will verify that the company logos print correctly and images in the application render clearly on both types of printers. You may want a different logo for color implementations like forms and reports that go to color, and a different image for reports that are output to black and white lasers.

Spell check the report before distributing it with the application. Unfortunately this is a manual process since Visual FoxPro does not have a spell checking capability (completely removed with Visual FoxPro 7). You could write a tool that checks label objects in the report metadata via automation with Microsoft Word or one of the third-party spell checking tools.

Validate that the report has standard items that you require on the report like date and time report was run, page numbers, and natural language filtering criteria. The filter criteria are especially important when the user can select different criteria when selecting the data for the report. It is just as handy when the customers call to report a defect that certain data is not showing on the report and you see right away that they filtered it out.

Testing is not complete unless you preview the reports to screen, and output reports to various file formats (ASCII, Excel, HTML, PDF, Word, etc.). This will depend on how you expose exporting capability for the reporting mechanism. Make sure each export type is tested and that it is opened up in the native editor or viewer. Also make sure when previewing a report that the second and last page is viewed. It might be obvious why to view the last page since you will want to make sure the entire report is executed and that the final summary information is calculated correctly. Why would the second page be important? It will make sure that the reports run with the `NOWAIT` clause do not have code to close and delete the report cursor.

Verify all formatting is correct. This can be testing the report expression *Format* property, as well as the general layout and format of the report. Make sure that the fonts used on the report are consistent with the other reports throughout the application. We standardized on numeric fields typically aligned on right, and alphanumeric data on left. The key is to align all the objects on the report so it looks professional. The alignment tools included in the Visual FoxPro Report Designer are excellent. There is no reason to have objects that are misaligned. Check all label captions for correct spelling.

Test the report with the executable. Many developers ship report metadata files (FRX) separate from the executable so reports can be changed without recompiling the EXE and redistributing it. If this is the case, the EXE needs to be able to find the report on the path, or the path to the reports needs to be included in the `REPORT FORM` code. If the report is included in the EXE, then running the EXE will verify it was included in the project.

Visual FoxPro developers who have used the Report Designer at some point have run up against the "Variable <variable> not found" error message while testing their latest application executable. This message is aggravating since it is displayed without telling you where the expression is flawed in the report. This expression is sometimes difficult to find since there could be dozens of fields on the report. To compound this problem, the expression failure could be in the calculation of fields, calculation of report variables, or Print When conditions. Fortunately there is a technique that speeds up the tracking of these painful defects. The key to a quick resolution is to suspend the program code after the final report cursors are prepared. If this is not practical, prepare the data manually. Once the data is prepared, modify the report and preview it. The error will be displayed. After you close the report preview mode the Report Designer will display the expression field that the error is occurring. At this point you can make the correction, save the report and try again.

General

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 SDT metadata was validated), and the backup and restore processes function without incident. Two other issues that I find developers frequently miss during testing is to check that the system conforms to the Windows' Color Scheme and that the application conforms to the customer's minimum screen resolutions.

Always have a separate area on the server/PC for testing the application. This allows you to test the implementation of the system and develop the process to accomplish the implementation. We have a directory for the latest development, a directory for testing, and one more with the latest version of the production system. This way my development staff can separately unit test, while

the QA staff can test a release candidate to be sent to the customer once we have completed our in-house testing.

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 that 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 I know when the customer asks why a feature is not working in version they are using (1.0.235) that the answer is because it was added in the next version and that they need an upgrade (to 1.1.175).

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 that the latest bug found by the customer was squashed. The key is to test everything that is 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.

Signoff

Sign-off should be obtained from the developer(s), the quality assurance team, and the customers. The sign-off is a commitment from the various teams that they tested the application that it conforms to the stated requirements and objectives.

Preparation before Creating the Deployment Package

There are a number of issues that need to be resolved before actually creating the SETUP.EXE.

Where should I install my application ActiveX controls?

ActiveX controls can cause developer problems when it comes to versioning issues. We can all relate to the technical support call from the customer that follows the pattern described in one brief discussion:

“I have a problem with this other application ever since I installed your application. It is causing an error on some TreeView control. I called their technical support and they said that they only support version 6 of this control and your application installed version 7. What are you going to do to fix this problem?”

We dislike taking this kind of support call, and we know that they are inevitable if you are using common ActiveX controls either provided with Visual FoxPro or ones that you purchase from a third-party provider.

To reduce the number of support calls and to follow the Windows logo standard we have adopted a standard. We have two folders to load our application ActiveX controls and components. These folders are based in the folder that the operating system understands as “Common Files”. This can differ on each user’s machine based on preferences and native

language. Typically it is found in the Program Files folder. We create a shared folder for our company in the Common Files folder.

If the components are specific to an application we install them in a folder under our company shared folder, under the application name, in a Component folder. Here is an example:

```
C:\Program Files\Common Files\GeeksAndGurusShared\OurCustomApp\Components\
```

If the controls are commonly shared across a suite of apps we developed for the customer we will install them into a directory patterned after this directory structure:

```
C:\Program Files\Common Files\GeeksAndGurusShared\Components\
```

The current install tools provide you a reference to the Common Files directory, which simplifies the installation. It keeps the System32 cleaner and hopefully there will be fewer support calls about any versioning issues. The Visual FoxPro 6 Setup Wizard forces the System32 directory route, so you will need to use a custom program if you want to use the old wizard and the new standard folders. Either way, the registry handles where to find them so that is a non-issue.

Another thing to consider when building the installation process is to see if you can mark the file to only be installed if it is a newer version. Many, if not all of the latest install building tools provide this feature. This can help with two issues. The first is that it can save a potential reboot of the user's machine since some ActiveX controls require the computer to be restarted after it is installed. The second advantage is that the installation will run faster.

Where do the Visual FoxPro runtimes have to be installed?

Visual FoxPro developers have been trained that the runtimes have to be in the Window's System directory. This is where the Visual FoxPro 6 Setup Wizard installs them. The truth is, they have to be available on the Windows Path, can be in the same folder as the executable, or can be installed anywhere and specified using the -D parameter to the executable.

The runtimes can be installed with the EXE on network or on client workstation. The consideration of loading the runtimes on the workstation is significant. Visual FoxPro can definitely access the workstation hard drive much faster than pulling the runtimes from the Local Area Network (LAN) file server or over a Wide Area Network (WAN). We always recommend that the runtimes be installed on the workstation for performance reasons. The issue needs to be addressed anytime a new version of the runtimes is released (via a service pack from Microsoft). If you upgrade the development environment, you will need to upgrade the production environment. This means that the runtimes have to be reloaded on each workstation. This can be quite a chore for a company's support staff.

There is no reason to "install" the runtimes via InstallShield Express, the Setup Wizard, or another installation package. There are no Registry entries to update during the installation. The runtime files can be copied from one machine to another or from a CD or Zip disk to the hard drive.

The drawback of the -D parameter is it requires a shortcut to the executable. If the user sets up a shortcut and forgets the parameter you could see different results. The big advantage of using the -D parameter is that it allows for multiple runtimes modules to be on the same workstation. This is important to know if you have releases of different apps on different versions of Visual FoxPro (service pack deployment issue).

How do I know which runtime files are being used after the install?

Since we can install multiple versions of the Visual FoxPro runtime files in different directories, we might want to know within an application, which set of runtime files are being used. The directory of the runtimes is retrieved using the `sys(2004)` command.

How can I distribute new versions of the runtime files?

Periodically Microsoft will update Visual FoxPro with a version release or a service pack. Applications that are updated and released after a Visual FoxPro version upgrade will require all the runtime files to be shipped with your application. The service packs can include updated runtime files that need to be release 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 I 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 you installed Visual FoxPro.

- 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 FOXHHELPS7.DLL.
- C:\Program Files\Common Files\System\Ole DB\ contains the VFPOLEDB.DLL
- 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.

You can review the list each time a fix is delivered by Microsoft. Now that we know which files can change, the question begs, how can you redistribute the runtime updates to the client sites? There are a couple of options.

The obvious way is to rebuild the distribution files via your installation tool of choice. If you are using InstallShield Express – Visual FoxPro Limited Edition, the new runtime files will be 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 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 only need to be registered using REGSVR32.EXE if your application uses Active Documents. Taking this approach might be the easiest way if you are onsite at a clients and just need to move a couple of runtime files to a couple of workstations.

The method of getting the runtime files to the client workstations will depend on many factors. You will need to evaluate the problem and determine the best mechanism for the situation. You have many alternatives. In the past many developers thought that they needed to build a complete install package each time new runtimes needed to be loaded.

What executable format can I release my application?

Visual FoxPro has four different executable file formats that can be released, EXE, APP, DLL, and FXP. Each of the formats can be used in the various types of implementations (desktop, client/server, n-tier, COM, and web).

APP files need either the Visual FoxPro development environment or can be called from another runtime Visual FoxPro EXE. If your clients have the development environment loaded on a computer you can run the APP by passing the APP file name as a parameter to the Visual FoxPro 7 executable. Here is a sample program call:

```
VFP7.EXE MyCustom.app
```

Visual FoxPro APP files are also used to implement Active Documents. This requires the main Visual FoxPro runtime file be loaded on the client workstation and registered. This is the only time that the Visual FoxPro runtimes need to be registered. The VFP7R.DLL is self-registered with REGSRV32.EXE. Active Documents can be executed via the VFP7RUN.EXE as well as the VFP7R.DLL runtime files. One advantage of shipping an APP file is that you can compile a component and just deliver the component. This is a common approach for a suite of applications. You deliver one main executable and a set of different APP modules that provide the various features. If only one of the features is changed you send the one APP instead of the entire executable.

The Window executable (EXE) is the most common Visual FoxPro executable implementation. It is an APP file with a Windows boot segment added to the APP file. This file can be executed from a shortcut, from Explorer, and even from the Windows' DOS command prompt. It requires all the runtime files (VFP7R.DLL, and VFP7R<LANGUAGE>.DLL which is the corresponding language resource file) to be available. Visual FoxPro EXEs can be called from other Visual FoxPro executables (using `DO <EXE>` and `RUN`). Objects in the EXE can also be instantiated by Visual FoxPro and non-Visual FoxPro programs (via the `SET CLASSLIB TO <class> IN <EXE>` and `CREATEOBJECT()`). Visual FoxPro EXEs can also be executed within the development environment in the same manner as the APP files. If there are classes compiled in the EXE marked OLEPublic, then other Visual FoxPro and other COM clients can instantiate the Visual FoxPro classes and manipulate the class properties and execute methods. The advantage of shipping an EXE is that you can literally ship one large file to your client installations. There is

no need to track a bunch of source files to send. The disadvantage is that it takes longer to ship the entire EXE over the web or longer for it to be downloaded by the customer.

The Visual FoxPro DLL is an in-process COM object. The decision you will need to make for implementation is whether you will be using the standard single-threaded runtime or the multi-threaded runtimes. The single-threaded runtimes simply blocks more than one object from executing code in the DLL. It queues up the requests and processes them in sequence. When the first object completes the property assignment or method execution, it processes the request from the second process. If the object method takes a half a second and 1000 objects simultaneously make a request, then it will take 500 seconds to process all the requests. The multi-threaded runtimes (VFP7RT.DLL) will not block the other processes from running. It will time slice the requests. The multi-threaded runtimes are also lighter and have no capability to display a user interface. Therefore many capabilities to output messages and data to the screen have been eliminated and the DLL is smaller in size. The multithreaded runtime library will also take advantage of multiple processors in the computer.

The compiled Visual FoxPro programs files (FXP) can also be released. These programs need to be run in the Visual FoxPro development environment, called from another executable (EXE or APP), or can be run directly from the VFP7RUN.EXE. The FXP can call all other source code objects like forms, reports, visual classes, etc. The advantage of shipping individual compiled programs is that you can implement a component or feature quickly, without the need to kick all the users out of the application. The disadvantage is that you need to distribute many files instead of one EXE or a few components. You will also be sending source code when delivering forms called from FXPs.

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 is likely to occur. The following are concepts for the tools that we have developed 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 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 is 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 do not 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 do not 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 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. I hear of this all the time. I'm just not the kind of developer that 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 does not handle that you will 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 will 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 are not using SDT and/or want a mechanism to generate the database and all 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, which is 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 will 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 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 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:

```
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 is a good idea to run this process for all the tables in the application periodically. One red flag that indicates that 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.

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 since I 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 **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 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 VFP development environment, but it could be a tool that you will find indispensable for your customers when you are onsite or doing some remote support via pcAnywhere. FoxBox is a Command Window (and much more) without having a full copy of VFP loaded on the customer site. It does require the same runtimes that are installed for your application.

NOTE: This tool can be downloaded from www.KirtlandSys.com or www.RickSchummer.com

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

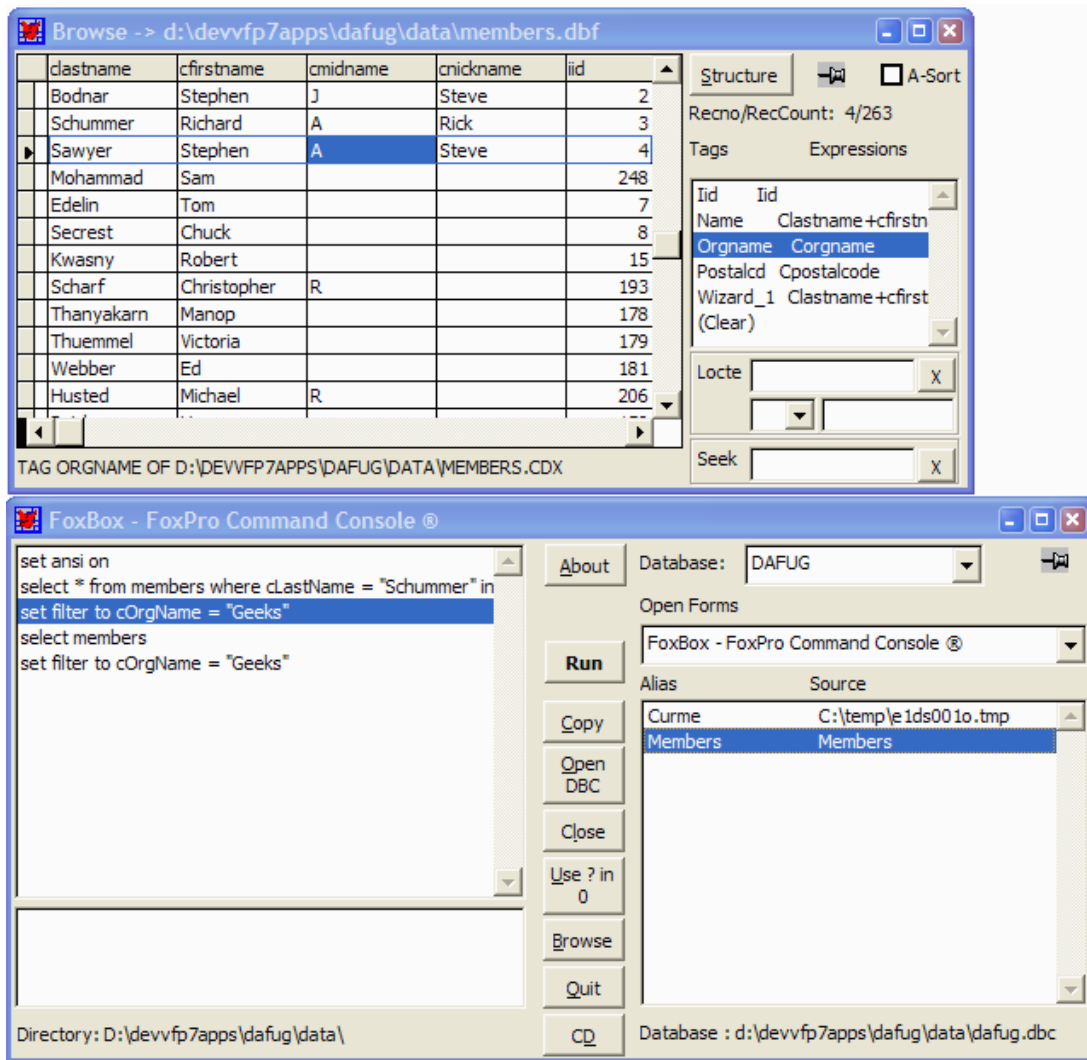


Figure 4. FoxBox is indispensable when debugging support issues on a customer's computer that does not have Visual FoxPro installed.

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.

Installation schemes available

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 have 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 have separated the upcoming discussion based on the different options that we can include in an installation package.

File Server Install

This is the main application installation and is included in nearly every installation package that is 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 will typically be 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 will have access to the application provided that they have network access and rights to these files. Using this scheme also requires that all the workstations have the Workstation Install (discussed next) loaded so they have the Visual FoxPro 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 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 that have 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 have broken the Workstation Install into 5 installs, all included in separate directories on one CD-ROM. You may find there to be more or less than this depending on your needs.

The first install is the Visual FoxPro runtimes. We find that 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 `VFP7.EXE` (and consequently, your custom executable), but it is faster since you are not pulling 4MB down the pipe every time you start the app. This install will load all the Visual FoxPro runtimes including `VFP7R.DLL`, and `VFP7RXXX.DLL` (where the `xxx` is the language of the Visual FoxPro version you have, i.e., `enu` for English).

The second install is the Multi-threaded Runtimes. This loads the new `VFP7T.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.

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 will not need this installation.

The fourth install is for the 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.

The fifth install is the 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 will need the ActiveX controls loaded on the computer that the installation is being built on. It is important to note that 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.

All of these installs are copied to one CD-ROM in separate directories. You need to do this because the install tool can name the setup (SETUP.EXE, SETUP.INF, SETUP.INI, SETUP.LST, SETUP.STF, SETUP.TDF) and corresponding cab (SETUP1.CAB, SETUP2.CAB) files identical for each install. You may decide to customize this CD as well 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 CD (or copies of it) can be passed around from computer to computer. We also recommend the CD is dated and note the Visual FoxPro version and Visual FoxPro service pack that the runtimes apply. It sure can be embarrassing to have that new Session class given to us 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 though may complicate this seemingly easy setup. What files need to be sent? Is this Visual FoxPro local data or are we 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?

The initial installation will require that the all 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 since 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. There are however many differences that your application may encounter. You will likely be installing the Multi-threaded runtimes for scalability, COM components or an EXE, HTML files, and be making Web Server settings (via executable or another manual settings) like scripting files, user security, and the mapping of drives to the data.

How do we package the install?

Now that we have developed schemes for the installation we need to determine how we will package it. I am not referring to the box that the CD is delivered in. The marketing experts best handle this. I am suggesting that you need to think out what installs discussed in the previous section 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 may 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 may only need some new HTML files so you can skip the

need to update COM objects. Single computer customers may 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 perfect software since it is likely the first impression that 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.

Mechanisms for Deployment

Preparation

First rule is to keep the customer informed of the progress so they are not surprised to see a new application in a basket on the doorstep. Communication is absolutely important during a successful delivery and implementation. At this point in the development cycle you should have a good working relationship with the client. I always send my clients an email and make a phone call to tell them I sent them an email. The email details the functionality of the release. This will vary depending on the type of release I am making. A new release of a new application will take on a different content since all the features are new. Enhancement or bug fix releases will have specific details on each fix or new feature implemented. One of the best examples of this is the information found in the update notes that Stonefield releases with SDT. Doug Hennig does a great job outlining each change made to the product. I find this to be invaluable as a developer and decided a long time ago that the same care needs to be made with the products I deliver. I keep track of this detail by adding a README.TXT file to my project. Each time I add a new feature or fix an older one I make a note. These notes are separated by version number and are dated. I copy the notes from this file and paste it right into the email to the customer. Here is an example of the ReadMe file from my HackCX developer tool, packaged as an email to the customer base:

Bob,

Good News! We have completed the updates to the G2 HackCX developer tool and want to let you know that we are ahead of schedule for the release. Here are the latest features and fixes that you can expect to test after we make the delivery next Wednesday:

Changes completed for v4.0.72 (17-Oct-2002):

- * Bug Fix: Not selecting a file during start up in standalone mode left VFP hanging
- * Bug Fix: Added ON SHUTDOWN code when in standalone so no "Cannot Quit FoxPro" message
- * Now allow User specific settings that override HackForm.ini settings (or add if not set up)
- * Changed the INI display form to show Custom User config and original INI contents

I will be following up with a specific test plan and let you know when the setup is officially ready to begin testing.

Thanks,
Rick Schummer
Geeks and Gurus, Inc.

If necessary I will also send follow-up emails and make a follow-up phone calls. Each customer requires a different level of customer care at this point. I have worked with CEOs that have let me into their office after hours to deliver a new update without anyone being in the building. I have also delivered with someone watching my every move as if I was an enemy spy, asking

pointed questions about each keystroke I have typed. Regardless of the level of information you provide, be prepared to provide more.

Methods

There are a number of mechanisms to deliver the Setup executable and supporting files. Understand the infrastructure so you are not surprised that the application cannot be loaded because they do not have a machine with 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

I know that diskettes seem like yesterday's technology, but I work with a notebook that does not have a CD burner built in. I have used my diskette drive to copy files to a customer in a pinch while I was at their offices. This is not common, but it is still a mechanism for installation. The VFP setup wizard will build the needed files sizes to be copied to the diskette media. It is uncommon today, but there are still companies that are using computers that do not have CDs in them and the only way in physically is a diskette drive.

There are two big problems with diskette delivery. The first is packaging. CDs are cheaper to send via snail mail based on the fact that you send fewer items. Ever had to ship six diskettes to 100 customers? The cost can be expensive with postage, duplication and assembly. The second is ease of use and image. Obviously diskettes are considered older and CDs are considered closer to current technology. I 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, I really hate the idea of installing a package than comes on several diskettes. It simply takes more time because the CD drives are faster and will work with less operator intervention.

One other problem with diskettes is the reliability issue. This technology has been around for decades, but it never seems to fail Murphy's Law 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. I know many developers that have been lucky over the years, and I know others that have had nothing but trouble. Your mileage might vary.

CD-ROMs

CD-ROMs are probably the second most popular delivery mechanism next to downloading from the Internet. The CD format provides a minimum 650MG of disk space for files and is supported by the VFP Setup Wizard by selecting the Netsetup or Websetup options in Step 3 (Create Disk Image Directory). Copy the single directory to the CD. If the Netsetup option (uncompressed) creates a directory larger than 650MG, try the Websetup option for compression. CD-ROM burners are cheap these days and the software works reliably. I 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 that there are fewer disks, which leads to cheaper packaging. The cost savings will be 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 my customer's office.

I 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. This way support staff (either onsite MIS, technical support personnel, or a representative from Geeks and Gurus) can go from PC to PC and install the runtimes. 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 or getting a label creation software package. I cannot tell you how big a difference the reception is when I 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. I simply note the application name, version number, release date, and include my company logo as well as the customer's company logo.

The disadvantage to the CD-ROM delivery mechanism is that it has to be either hand delivered or via the snail mail system. Depending on geographic distance and whether you ship priority overnight or through standard mail channels will determine how fast it will get in your customer's hands. 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 CDs because they do not want employees to play music while they are working (yes, this is not just a Dilbert story).

Email

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

Sending an executable or an update to the databases works great over email. The clients detach the files I send, unzip them to the appropriate test folder, run the application and verify the results of the test plan (do I 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 utilize.

One of the problems I have run into with customers using email is that they do not know how to detach the attachments even if I send explicit instructions on how to accomplish this task. This is common with computer phobic people. A common resolution to this is to call the customer on the phone and step them through the instructions. Another issue I have run into over the years is that some email systems in the corporate world limit the size of attachments on outgoing emails. This may set up a roadblock for corporate developers shipping releases to their customers. The same limitation has been found with certain ISPs.

Symantec's PcAnywhere

If you have only one tool that 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 as well to allow you to deliver files needed in the implementation and subsequent support of your custom application as well. This tool has been a must have for our company to the point that I will buy it for a customer as part of the deal for the custom application. It saves on onsite travel and provides immediate support to their problems.

The file transfer functionality of PcAnywhere 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 VFP executables that have a small set of bug fixes or enhancements. I have seen executables around 8MGs ship in a minute or less over a slow dial-up connection and it works fabulous over high-speed Internet connections. It is almost as if you were running the computer directly.

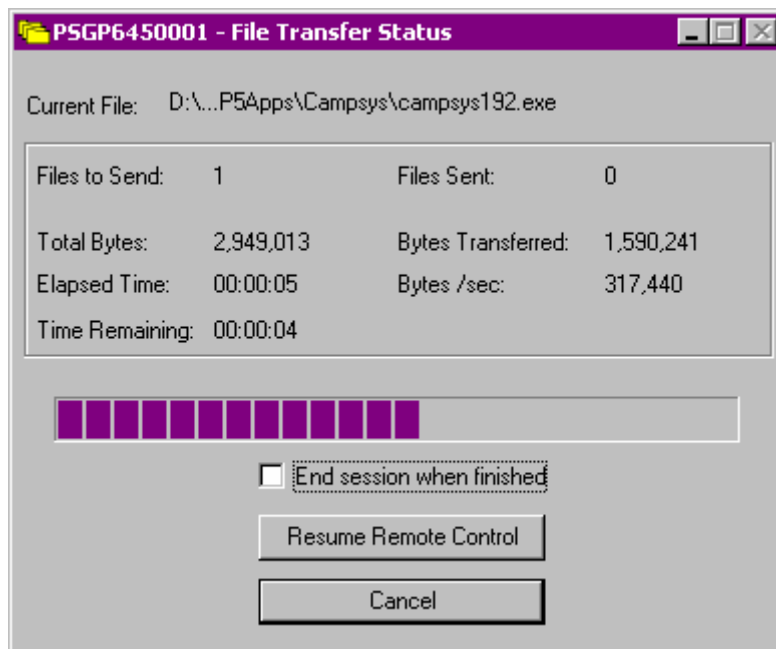


Figure 5. File transfer feature of pcANYWHERE in action.

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) the customers tend to get cranky. We have customers that connect to the web with the @Home cable modem service. There seems to be a built in firewall and will not allow a PcAnywhere connection via TCP/IP. Those computers that are 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. I have found this to be a real pain, but working with them and explaining the reasons for the package can usually find some compromise.

The biggest issue with PcAnywhere is that it is single-threaded. This means that you can only support or transfer files to one computer at a time. This is not an issue for a small customer, but will not work as well in an environment where there are numerous computers to install the application.

Citrix and Terminal Server

Citrix and Terminal Server is a product that allows 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 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 previously. The big difference is that the Citrix 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 like us 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. Since the work is done on the server there is no network traffic generated other than the screen updates.

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 very common yet.

Web Site Download

The Internet is obviously a terrific mechanism in our industry. Personally, I rarely buy software in a brick and mortar store these days. I typically download it from a Web site. If we can do this, why can't our clients? While I can say that 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 that they decide when they want to take the hit on their bandwidth. If we send it in email, they will take the download hit just after we send it. This can be a problem if it takes 30 minutes when they are waiting for an email order from one of their customers. This is an ideal transfer mechanism for vertical market apps as well since 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.

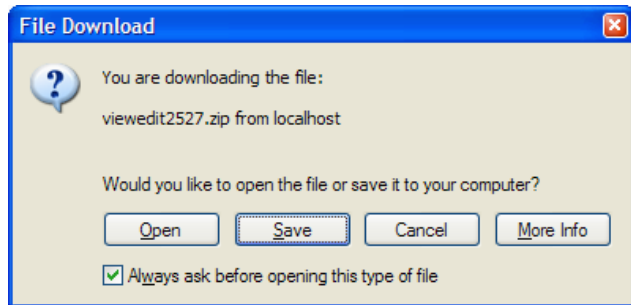


Figure 6. Downloading files from the Internet via the Internet browser has become commonplace.

There are drawbacks to this mechanism as well. If you do not have a logging feature on the Web site, you will not know if the customer came to get the installation. If several clients are running the software 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 ISP does not give you this capability, users or any other surfer could potentially get the installation files.

Deployment Tool Selection Process

Geeks and Gurus experienced the need for a deployment tool soon after we formed our company in April 2001. Our first large scale deployed application (read: more than one computer) was ready for delivery in September. In May we began the search for the deployment tool (soon after landing the project that needed the large scale deployment).

NOTE: The evaluation process was traveled in mid-2001. Several of the installation tools have improved since then, the results we concluded could be different if the evaluation was done today, it might also be the same. One thing is for certain, the tools are better, some tools are named different, and some tools are no longer available.

NOTE: The following tool selection process is the experience we went through. It does in no way provide an endorsement of the tools selected, or reflect negatively on the tools we did not select. We encourage all developers to look at all tools available to make an informed decision.

We started out with the InstallShield Express – VFP Limited Edition (beta). The tool initially appeared to provide everything we needed. The criteria that we looked for was as follows:

- Professional look and feel (compatible with today's Windows standard deployment)
- Easy to setup
- Registry entries
- ODBC datasources
- Multiple directories
- Create shortcuts on desktop/start menu
- Licensing and ReadMe text
- Ability to create CD-ROM and Web download installations
- Must work with Visual FoxPro as well as other tools
- Custom Actions

Nice to have features included:

- Merge Modules
- Features functionality

- Configurable dialogs

Steve Sawyer (my partner at Geeks and Gurus) previously looked into professional installation packages and was quite happy with PC-Install because he successfully used it for custom software he developed years prior. He did an independent review of several installation packages:

PC Install

Inexpensive, fairly robust and flexible installation tool, with a simple scripting language, including a variety of user input dialogs. Able to branch installation execution based on system information or user-supplied information or response to prompts. Can write to registry and INI files, and register DLL's and ActiveX controls. Can run pre-setup and post-setup executables, as well as executables in the middle of the installation process.

Limitations:

- 1) Unable to read registry entries
- 2) Able to assign files to installation groups, but group selection is mutually exclusive. The group selection dialog can allow selection of multiple groups, but it is not possible (as in InstallShield InstallBuilder) to programmatically build (based on user input) a selection criteria that will install multiple groups. Thus it's necessary to create a file installation group for each installation pattern, necessitating maintenance of multiple file entries.
- 3) Self-registering DLL's and ActiveX controls must be explicitly registered
- 4) No ability to create custom dialogs, alerts or end-user instructions. Only "canned" dialogs are available.

Was available from 20 20 Software (www.twenty.com), we are unsure if it is still available since their website does not respond, and Hallogram says it is not available.

ActiveDelivery

Overall, a good and simple installation utility. Suitable primarily for duplicating a particular directory structure on a target machine. No scripting capability, thus there is no ability to implement different installation scenarios based on platform, user input, or compile-time arguments. Might be suitable for doing simple updates of executables and metadata, and for running a post-install executable to do necessary housekeeping to implement the new metadata. Can run pre-setup and post-setup executables.

Limitations:

- 1) Lack of a scripting language
- 2) Requirement to modify the destination path of each file individually if the original path stored in the ZIP file needs to be modified
- 3) No provisions for specifying more than one extraction directory at installation – all files are extracted relative to the specified directory, or to directories hard-coded into the installation package
- 4) No provision for reading registry entries, although the product can make registry entries and register .DLL's and ActiveX controls
- 5) Limited "rules" for overwriting existing files

Available from InnerMedia (www.innermedia.com)

InstallShield Express

Simple, point-and-click interface. No scripting capabilities. Can make registry entries and register DLL's and ActiveX controls, as well as make modifications to INI files. Automatically reads version information from the executable to be installed, and automatically displays this in the setup screens. Can group installation files into components that can then be installed according to one of three pre-set setup types, custom, typical and compact. Allows incorporation of custom graphics and billboards into the installation.

Limitations:

- 1) Unable to prompt the user for more destination or setup information other than the installation directory and components defined under one of the three pre-defined setup types.
- 2) Unable to create user-defined setup types beyond custom, typical and compact
- 3) Unable to read registry entries
- 4) No capability to set rules for overwriting existing files
- 5) No scripting language

Available from InstallShield (www.installshield.com)

Wise InstallBuilder 8.1

Powerful, flexible tool. Full scripting language. ability to register DLL's, ActiveX controls. Can read and write registry entries and INI files. Can call DLL functions. Scripts can be run with compile-time variables that are passed as arguments from the command line, or from user prompts generated when creating an installation to allow various types of setups from a single script. Scripting language makes use of user-defined variables and user-designed dialogs as well as standard pre-built dialogs and system variables. Can search for existing components and branch conditionally based on what components are installed. Installation files can be grouped into "components" which can be installed conditionally at run-time based on either user input, compiler variables set at compile time, or on searches for existing components. Components are not mutually exclusive, allowing highly granular component definition, and installation of multiple components based on user input or compiler variables. Provides full control over overwriting of existing files – "Always", "Never", "If newer", "If version is higher" etc. as well as the ability to prompt the user for permission when overwriting files. Custom billboards and dialog graphics. Includes ability to install VFP runtimes automatically. Extremely thorough documentation. Has a command to automatically register all self-registering DLL's and ActiveX controls. Self-registering controls are often recognized as such, but those that are not recognized as self-registering can be so flagged manually.

Limitations are:

- 1) The "Installation Expert" interface, which is the easiest interface for initially selecting files for the installation, is incapable of recognizing NetWare volumes, forcing the user to select installation files one-by-one through the script editor interface. Wise tech support states "some of the times we do see this with Novell servers", but are raising the issue with their developers. On the upside, once the files are entered into the script, it's possible to select a large group of files, and set common properties at one time, such as source path, overwrite rules and destination directory. As a work-around, setup files can be drag-and-dropped into the file selection list from Windows Explorer. Any subsequent editing of the file list will probably be handled through the script editor, which doesn't have a problem with NetWare volumes.
- 2) Unable to automatically detect and set the executable version number to be displayed during setup (as can InstallShield Express), although the compiler can be set to prompt the user with the correct version/build number for each installation

Available from Wise Solutions (www.wise.com)

Making the choice

The beta of InstallShield Express – VFP Limited Edition looked very promising initially and eliminated some of the limitations of the evaluation of the older version of InstallShield Express, primarily the addition of the DATABASEDIR which allowed a second directory prompt so users can direct data to another directory (not just the application directory like the VFP Setup Wizard). Then during testing we discovered that the Limited Edition was indeed very limiting since it disabled the “Upgrade Path” feature. Essentially this means that the user needs to uninstall the previous version before installing the upgrade. This was unacceptable (see section “What are the disadvantages of using InstallShield Express vs. Setup Wizard?” later in this paper for more details.

Wise InstallBuilder first, Wise Installation System next

Based on Steve’s evaluations we jumped into Wise InstallBuilder 8.1. It recognizes the VFP runtimes, handled all the requirements, and all the optional requirements except for the merge modules. We worked with the scripting capability and the dialog editor (to allow for the customer required data directory selection) to create the setup for our customer. It was easy to learn and was far more powerful than the InstallShield Express Limited Edition package. We literally created the setup the same day the package was delivered. Wise upgraded InstallBuilder 8.1 to Wise Installation System 9.0 almost immediately after our purchase, so we got a free upgrade. We updated our project without any problems. Our satisfaction level was extremely high and we knew we made a good choice. The technical support was outstanding and another advantage we learned is that Wise Solutions is located in the suburbs of Detroit, right in our own backyard.

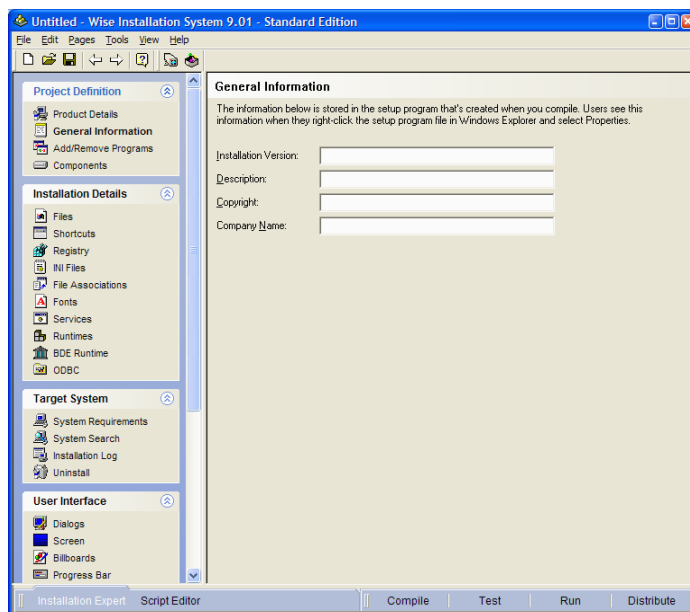


Figure 7. Wise Installation System 9 looks every similar to InstallShield Express – VFP Limited Edition, but is far more powerful.

Ch-Ch-Ch-Changes

Our software package went through a couple of updates, and the Setup process was a no-brainer each time. Then our customer (another developer shop) called one day with the question that at first seemed harmless, but turned out to be a nightmare. They had an installation that had multiple users that shared the same machine which had Windows 2000 installed. They found out that they had to install the application for each user. This was required because the installation created Registry entries for the current user.

I remember the quote I said to our customer: “no problem” (which my partners are kind enough not to remind me all the time). I called Wise tech support and they noted that the Installation System did not support the feature, but that the “weaker” Wise for Windows Installer could. No problem. I gave them my credit card number and downloaded the package. The feature was available, but we had a heck of a time getting the DATABASEDIR functionality. Wise for Windows Installer did have basic scripting and dialog creation, but I could not get it to work, despite the tech support staff at Wise working a week to help me out. There was a quirk in the scripting between the editor and compiling the setup. I put in over 40 hours trying to get this to work, over 40 non-billable hours.

My last call to Wise was to see if I could get the Installation System to do the All Users installation. The sales person that I got this time around noted that there was an add-on that we could purchase. Unfortunately it came with a per-seat license scheme. How much? The sales guy noted it was only \$4 a seat. Our customer did not have many seats (under 100) so I figured I would cut our losses and buck up the \$400. Unfortunately the sales guy informed me that I needed to buy the minimum 1000 seats, \$4000. No bleeping way (pardon my French)! It was time to find another solution.

InstallShield Express (full version)

I called InstallShield to verify the feature was available in the full version (it is in the Limited Edition, see section “How do I have the install files registered for all users of the computer?” later in this paper). No problem. I created the same installation as I did in Wise Install System the same day it arrived, in less than 3 hours, including testing.

The problem with InstallShield Express was the fact that there was no upgrade path from the Limited Edition, despite the wording in the product to the contrary. When I called the sales people I found out that I needed to pay full price. I had no problem doing this if it worked, but it was the principle of the marketing that made me hesitate. I purchased it anyway because it was cheaper than wasting more time with Wise quirks.

Several of us in the Fox Community voiced our displeasure to Ken Levy who was kind enough to pass along our concerns to the InstallShield Express product team. The results of this discussion were the short-lived upgrade discount of \$100 offered to VFP developers back in March 2002. Unfortunately that offer no longer exists for developers purchasing VFP 7 now.

Lesson Learned (again!)

If we need the “All User” feature we go with InstallShield Express, if we don’t need this feature and need more control over the installation we use Wise Installation System. They are both fine products and serve us well. If cost is a concern and you figure to need the “All User” feature, and

you like the way InstallShield Express - VFP Limited Edition works, you will really like the full version. We have found it more than satisfactory and it is worth paying for the full edition. We got over the “upgrade” and have always felt that professional tools that save us time and money are well worth the investment.

Other Choices available since our evaluation

InstaFox from Fowler Software Design LLC (www.fowersoftware.com), installation tool completely written in Visual FoxPro. Full Source code provided.

- FoxPro Advisor review by Art Berquist. “Create Professional Installation Routines” in March 2002.

I have not used this product, but wanted to provide you with a link so you can do further research to see if this tool meets your deployment requirements.

What is in store with VFP 8?

Microsoft is shipping an upgraded version of InstallShield Express – VFP Limited Edition with Visual FoxPro 8. Here is the list of new features documented in the VFP 8 Beta ReadMe HTML. I have listed these in the order of importance for developers needing to make a decision of upgrading.

Upgrade View

Creates new product versions that automatically uninstall the previous version (must be MSI-based) and install the new one. This eliminates the limitation posed by the missing “Upgrade Path” in InstallShield VFP Limited Edition v3.03 (shipped with VFP 7).

Windows XP support

Conditionally installs the project or individual files based on the presence of the Windows XP operating system. The previous version did not have this option, thus if you selected that it required a specific OS, and tried to load it to Win XP your customers would not be able to load it.

Post-setup EXE

Specifies that a dialog box containing a check box for launching an EXE file is included at the end of an installation. This is a feature that we had in the old VFP 6 Setup Wizard that was lost when Microsoft went to InstallShield Express Limited Edition v3.03.

Merge Modules path modification

This functionality provides the ability to add additional directories to the search path used to populate the Merge Module view. This is handy when you have created specific directories for downloaded Merge Modules, or want to build installations with VFP 7 executables and runtimes with this updated version of InstallShield. Many third-party tools and ActiveX controls that you

would ship with your installation have precanned merge modules for you to download and include in your deployments.

Windows Installer 2.0 support

Windows Installer (WI) 2.0 is used by default, with an option to revert to WI 1.2 engines. The newer operating systems already ship with WI 2.0, and you can optionally ship the installers with your deployment package.

Detects previous installations

Installation detects existing versions of InstallShield Express and displays a warning message about overwrite during installation. The previous version of InstallShield Express (ISX) Limited Edition would automatically load over any version of ISX, including the non-limited editions. This version recognizes that you have a better version (or more current version) than the one you are trying to load. This will save some developers some pain down the road, and has nothing to do with the creation of installation packages).

Ensure a smooth deployment

We have spent years developing and deploying applications. While there is no perfect checklist or plan to successfully deploy applications, there are several items to note that definitely lead to a smoother implementation. Planning up front (even as soon as the requirements collection phase of the development life cycle) is the key.

The process of moving an application into production is the end of a long road traveled. Hours, sometimes weeks or months can go into a new set of features. Depending on the project even years of blood, sweat and tears can go into an application. Now the light at the end of the tunnel approaches. Is it the sun at the end of the tunnel or a freight train about to run you over?

Deploying a project into production can go smooth, or not, depending on the planning and attention to detail. This month, the next to last installment of this series, discusses some things to verify as your clients move the product into production and ideas on supporting a release.

Checklist/Instructions

[*InstCheckSample.doc*](#)

Having a checklist or step-by-step process of implementing a release is important. I have seen the panic on other developers' faces or have myself had that pit of the stomach "oh no" moment. This is when something goes wrong with no way back to the original state of the data, or the application, or both.

Installations software is much more sophisticated today than the DOS batch files we used in the 1980s. Still, having complete step-by-step instructions on how to install the software is essential to the success of an implementation, especially if one of your customers is the one implementing. Do not write volumes on the topic because they will never be read because they will be perceived as overwhelming. Make them concise, but informative. The instructions will include all the steps up to running Setup.exe and all the steps needed to after the setup is completed before they start using the new application or update.



Production Installation

Before Installation

- ☐ Sign-off from customers that application is ready for production
- ☐ Double check implementation date and time with clients
- ☐ Inform team of pending implementation
- ☐ Verify types of installations needed (server / workstation)
- ☐ Create test area and rehearse implementation
- ☐ Create backup of application files in save directories
- ☐ Verify or complete backup of production data
- ☐ Verify or complete workstation installs (if necessary)

Installation

- ☐ Perform Setup on Workstations (if necessary)
- ☐ Perform Setup on Server
 - Run Setup.exe for server
 - Update application Configuration Files (if necessary)
 - Update database structures (if necessary)
 - Update data via data conversion (if necessary)
 - Update/Fix primary key generation table

After Installation

- ☐ Test installation
- ☐ Notify users
- ☐ Follow up emails and phone calls

Figure 8. *This is a sample checklist for installing an application in production.*

Here is a short list of items I like to make sure I do handle to avoid the failure and possible embarrassing situation with my clients.

Test Area

Creating a test area has to be the least used and most important concepts in moving an application into production. The test area is an exact replication of the production area. All files needed to run the application need to be included in the directories. This includes executables, databases, tables, metadata, external source code files (like reports), language translation tables, parameter/setup tables or INIs, ActiveX controls, COM objects, data conversion programs, problem solving utilities, help files, shortcuts, and any other files used in production. For existing systems that are getting an update, copy the base set of files from the current production directories to provide a true experience of what can be expected when you work on the production implementation. This is the area to run a complete dress rehearsal of the implementation without directly working on the production files.

It is important to understand that this is separate area from the beta test area that you have been using all along for customer acceptance testing. Using the beta directories introduces the possibility of bad data or errant test versions of the code which might introduce invalid test results that are hard to reproduce in the development test environment. This final test area provides you and the customer one more chance to test the implementation steps necessary to insure a successful production implementation. Each test should start out with a fresh copy of the environment so you might want to make a second copy of the files before making the first test.

There are several items to make final testing at this point of the process. You might have to test various machine configurations at different customer locations. For instance, you can install on a machine that has older runtimes, one that has no runtimes, older processors vs. current ones, and various memory and video combinations. You are not looking to be surprised that the application will not work on these at this time, just confirming that the implementation goes smooth.

Backups

Another big mistake that I have made in the past is not backing up the files I am changing. This takes an extra few minutes to create a save directory, copy the executables and any other files that you will be changing. These files can later be restored if something goes wrong. This precaution is for those updates that you are 110% sure (or possibly less) that the 5 minute bug fix you just made will work flawlessly. Next thing you know the customer is calling asking why all the invoices no longer show the line items or worse, have some how destroyed data.

I always feel most comfortable when the user has just backed up the data. If they do not have a current backup I ask that they perform a backup before implementation. This is typically a scheduled event so there is plenty of time to ensure that it is not only completed, but that it is verified to be a good backup (especially if done to a tape backup). With disk prices as cheap as they are these days I really prefer backups of this nature to be made to a second drive or another computer on the network. This provides the best performance in case I make a mistake and need to get the backup restored. The customers are never more anxious than when their system is down and the phones are ringing.

Runtimes

The runtimes files are need for our FoxPro applications to run on computers that do not have a copy FoxPro installed. The biggest key with the runtime files is that they are the same version as the VFP version you developed the application with. This can be trouble if the users have several applications that were developed over time. VFP 6 for instance had new runtimes in several of the five service packs that Microsoft shipped. If you developed one application with service pack 3 and another application with service pack 5 you can have different results with the executable sharing the latest runtimes. It has been more than a couple of years since I have run into trouble with this type of problem.

I discussed the various possible locations to load the runtimes in the Deployment: Installation Preparation article published in November 2000 issue of FoxTalk. I also discussed several installation schemes including a separate Workstation Install Setup vs. the Server Install Setup. During the implementation of the production application you will need to make sure that the proper installations are run on each workstation and on the server. The workstation installs might not be necessary if no ActiveX controls were added or updated and the runtimes are consistent with a prior installation.

One issue that I did not address is the Windows 2000 logo issues which have the runtime files and other associated dynamic link libraries loaded with the application executable or in its own directory outside of the Windows System directory. If this is important to you, be sure to take the extra steps necessary to instruct the installation routine to load them outside of the Windows System directory.

Configuration Files & Registry

Many commercial frameworks have a mechanism to save user preference and application settings to a system table or to the Windows' Registry. New applications will not have to deal with updating these settings, but further updates to the application will likely require a program or a step in the installation process to add new options and default values.

If you are using the VFP Setup Wizard, updates to tables can be handled with a program that is fired from the post-setup executable. Registry entries can also be handled in a similar program as well. Most commercial installation routine tools provide a Windows' Registry key creation and value updater built into the product. This saves you the extra step of rolling your own program and is a little bit cleaner because the failure errors are handled right in the setup and do not have the second process executed.

Update Structures and Indexes

Previous articles in this series have addressed the database updates, table structure changes and changed/updated indexes. If you are using a VFP DBC you have two basic options. The more difficult way is to track all the changes you have made and write a program to make the changes with `ALTER TABLE` and `INDEX ON` commands. You also can write your own routine that will determine the differences programmatically and generate the code. The second option is to purchase a subscription to the Stonefield Database ToolKit which has the `Update()` method that handles this for you. The implementation of this is simple. All you need to do is copy the new database files (DBC, DCX, DCT) and the SDT metadata and programmatically call the `Update()` method.

SQL databases will require a different implementation because there is no equivalent SDT for SQL databases. Each database has its own method to generate script to make changes in production.

Conversion

You might be thinking that I have a screw loose because I just talked about converting the database structures and want to discuss conversion again. This conversion is a different and separate step in the process. Once the data files have the new structures in place you might optionally need to move data around to normalize or denormalize records, add default values to new columns, add records to support/lookup tables, correct or update foreign keys, or even completely populate a new database from the system that was used before the new one was implemented. This is best handled after the structures are in place.

Like the structure updates, if you are using the VFP Setup Wizard to generate the setup routine, you will need to use the post setup executable. If there are several items you need to accomplish after the application files are copied to the appropriate location you will need to write a program that will handle all the needed steps.

Test

Once the application is loaded and configured it is important for one last test. You are not looking to do a complete system test, just make sure the executable starts and some primary forms and reports execute correctly, the data converted correctly if a conversion was done, and that the version number on the About form matches the expected version implemented. One other thing I like to verify is that any ActiveX or COM components used in the application are accessed correctly. Just give it a general once over to give you confidence that the installation went as expected.

Notify the Customer

Finally the moment we all look forward to, telling the customer that the application is in production and all the new features and fixes are available for them to use. You might have an opportunity to tell them in person if you are onsite. If you are implementing after hours and they already went home for the night you might just leave a hand written note on their desk. Remote implementations present a different challenge. Typically I open an instance of Notepad and just leave a quick message that I am finished with the implementation.

A follow up email and phone call are also important telling them the status of the implementation. Hopefully you will be informing them that all went well and that the application is ready for their use. Sometimes things do not go well and you need to tell them that the application is the same one they have been using previously, or if new, that the application is not yet installed.

Provide documentation of exactly what was installed from a feature set point of view. The documentation I like to provide is a Word document detailing all the changes made, both new features and bug fixes. (see **Figure 9**.)

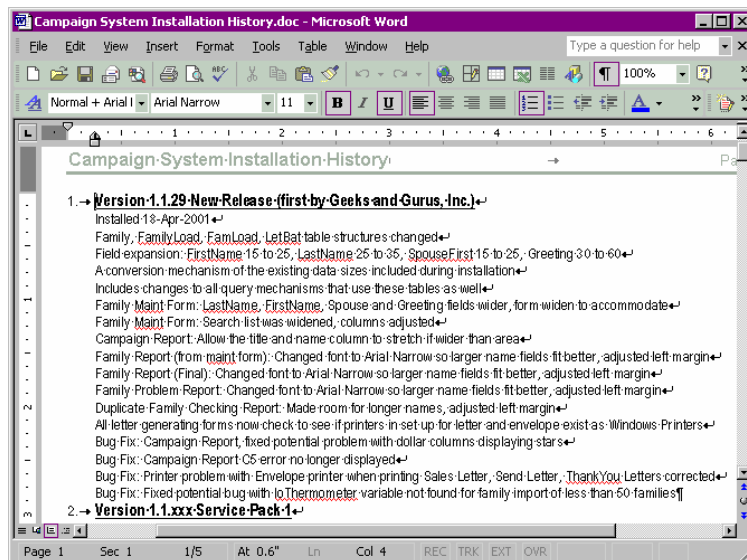


Figure 9. Example of documenting release history for an application.

Technical Support

Not every company has a staff dedicated to Technical Support, but all applications are supported in one shape or form. This is accomplished by the developer who wrote the application, or by other developers (inside or outside of your organization), super users onsite, or by the help desk support staff.

The need for support of a release will vary depending on the business environment, the type of application, the hardware configuration, contractual obligation, and your relationship with the client. For instance, a corporate developer might be developing onsite and just need to walk into the next room and walk cubical to cubical with the bug fix CD or stop in to answer a question. On the other hand, when I was a corporate developer my applications were running in 300 sites across the United States and Canada, and some in South America. The support for this

environment was quite different. Other developers might have to drive across the city and do the same or might just dial into the server and inspect the situation.

I encourage users to inform technical support of problems that they are experiencing. I cannot tell you how many times I have been onsite to review something with a client when I poke my nose into the application error log and find a slew of records. I then ask the \$64,000 question: “How long were you planning on suffering with this error?” The users usually hide these things thinking it was entirely their fault because they did their job incorrectly. While we could ride this excuse and not take blame for a bug, it is important to inform them that the software can only get fixed if we know something is wrong.

Manning Communications

The fundamental mechanism of technical support during a production deployment is communication of issues from customers. In today’s world it is easy to be connected with telephones, beepers, cell phones, email, fax machines, and Instant Messaging. Which ever communication mechanism used, make sure that you are available for the customer during implementation of the production application. After all, their business may succeed or fail with your application and this translates to the possibility of your business succeeding or failing as well.

What are reasonable hours? The answer is: it depends. I have worked for a number of companies both large and small. I have been very fortunate in my career never to be woken up in the middle of the night to handle a support call. It will depend on the location and times that your customer is using the applications. Deployments can be done any time of the day or night, but the reality of the situation is that they are done when it is most convenient to the users. Typically this means that after hours or during lunch the application is not used. Sometimes it means working a weekend. This is something you should negotiate long before the CD is inserted into the computer and run.

FAQs

The easiest thing to keep track of is a list of Frequently Asked Questions (FAQs). This allows the developer(s) or technical support staff to quickly learn about the most common issues that are handled from the customers. This list of questions can be developed as soon as we start collecting specifications and will certainly be developed during beta testing, and added to after the application is in production. This list can be included in the user documentation, help file, in the readme.txt file, or on the support web site. Larger companies like Microsoft, Symantec, Adobe, Corel and the like have created a KnowledgeBase of white papers that are based on issues related to the different applications that they support. Who is to say that even the one-person shop cannot leverage these concepts?

Help File

I can already hear the laughter from all corners of the world. Who the heck writes help files for custom software? Reality is that many custom projects do not have help files because they are written directly to the users specifications and that the clients are continuously using and testing the software and know it intimately before it is released to production. This can be perfectly

acceptable. Some customers will not pay for the development of the help system because it can cost as much as the development of the software. Some users are corporate IS departments and assume the responsibility of developing the User Manuals, help files and training materials.

However there are advantages of developing a help file. Naturally it can be useful to the people using the software each day. Would you develop applications without the VFP help file? The second advantage is that it is a place to document the specifications to the end users. It describes the business rules, the process re-engineering, and the various decisions that were made on their behalf by their peers or superiors when the software was developed. It can also be used to validate a problem that does not meet the specification.

Training

Training can be a one-on-one with the user, or it might be a formal class for all the users, or even a train the trainer session. What ever the needs are for the users, training should be handled initially long before the application is moved into production. Ongoing training is sometimes required based on user turnover, a loss of a trained super user, or new customers coming online with an application.

Web site

A number of developers have asked me my feelings on the topic of Web sites and the roll it can play for developer shops. I think it can have a significant roll in the arena of technical support and customer service as you move an application into production. It provides one stop shopping for three previous topics of communication, FAQ, and help file, as well as place to publish white papers about the products and services you have available, downloads of demo products and secure updates for existing customers.

First and foremost it gives your customers and future customers a place to find out how they can contact you. The company address, phone number, fax line, and email are all important communication methods to make available.

I have already discussed the idea of allowing customers to download updates earlier in this white paper. One of the nice advantages of allowing them to download the file is that they decide when they want to take the hit on their bandwidth. If we send it in email, they will take the download hit just after we send it. This can be a problem if it takes 30 minutes when they are waiting for an email order from one of their customers. This is an ideal transfer mechanism for vertical market apps as well since 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. This also works well for those patches made during the production and post-implementation phases of deployment.

Later in this white paper, I will discuss about this in more detail, but your Web site is also an excellent place to put in a bug reporting mechanism. These reports can easily be turned into fixes and posted as part of the technical support knowledgebase. Vertical market applications have an additional element to them because they typically have a larger and more diverse user base. Publishing reported issues and the resolution or workarounds on the Web site can help cut down the number of technical support calls taken.

Considerations for Small Shops

There are definitely some special considerations for small shops (1 to 5 developers). Our clients sometimes run our applications 24 hours a day, seven days a week. What happens when we need a vacation or attend a conference? In the case of Geeks and Gurus (a four VFP developer shop as of the writing of this white paper) we are all attending the GLGDW conference in Milwaukee (where this session was initially given). How do we support our clients when we are attending technical sessions and the various events when we are out of town? For the most part the same as we do when we are in the office with some special handling.

First and foremost we have contact available via our cell phones. Our cell phones have voice mail in case we are unable to take the call immediately. We carry our notebook computers so we can investigate the problem via source code and to dial into their systems if necessary from our hotel room. I have even supported my clients from a remote campground in the past so almost anything is possible. We also handle our email as many times a day as needed to provide top-notch service. The key as usual is fast response and appropriate action to satisfy the customers' needs. Our customers are also notified in advance when we are leaving town or are unavailable for a period of time.

Post Implementation

Finally you have moved the customer application into production and the users are tracking their mission critical information. No major issues need to be resolved and no bugs are being reported (for the moment). So one might be asking if it is time to sit back and smell the roses or is there still more work to complete for this project? While it might be nice to kick back in the easy chair and take a breather, there are a few key details to be covered before we can completely relax. We discuss some things to verify that all went well for the customer, to verify that the internal processes set up for software development at our company are optimized, and ideas on how to leverage the recent success to bring more business your way.

The Post Implementation stage of deploying applications is as important as any other. Now that the users are busy tracking mission critical data, what do you and possibly your team do to finish up the deployment? There are several key steps in wrapping up this phase of the development cycle.

Some developers believe that all the steps outlined in a development methodology must be followed to the letter. This is far from the truth and definitely does not apply when it comes to this part of the deployment phase. This section of the white paper will offer some suggestions on what you can do after the application is in production. Feel free to pick and chose what works for your project. Some of these items have no purpose when implementing a 30 minute bug fix, others are mandatory when completing a full fledged development cycle.

Customer Follow-Up

It is always a good idea to get together for a meeting to do a follow up survey as well. I like to have these meetings at neutral lunch site or occasionally at the customer's office. You can have a formal survey sheet for them to fill out at their convenience or can take it verbally while waiting for the order to be delivered. This is a perfect opportunity to get a feel for additional business or see if they have business associates that could use your expertise in solving problems. Follow up

emails to get the survey returned and occasionally keeping up on their progress with the software needs should be a natural process in your business dealings.

There are four reasons to follow up with the customer. The first is to make sure they continue to be happy with the application and that it meets or exceeds their expectations in the production environment. There is nothing more real than using the application in the real world to flush out those issues that were not covered during acceptance testing. If there is one thing that gives us more business, it is picking up applications from new clients where the last developer or shop failed to provide acceptable customer service. Do you want your current clients to look at the competition for their next project or to take up the enhancements to the project just implemented? I know, sometimes the answer is yes depending on the customer, but the majority of the time we want to retain our business.

The second is to determine how well we did in the customers' eyes. No matter how cool or killer you think the latest version of the app is; it may not meet the customers' needs. It may be something simple to fix in the interface that is hard for the user to work with daily. One of my favorite examples of this was a simple grid entry. I missed the *SelectOnEntry* property of a textbox in one column. It just so happened that the entry was normally overwritten with new numbers and the customer was constantly highlighting the contents with the mouse before typing in the new value. One 5 minute fix saved them hours a week doing data entry. This feedback will also improve your development processes at the same time as you learn common problems that your customers reveal.

The review with the clients will also help out in another important area, figuring out the separation of the bugs from the "by design" features. How many times have your customers called up and said the system was failing to perform a certain way, when in fact it is working exactly as they specified just hours earlier? It is my experience that the customers need to be trained over time on what a bug is (a true failing of a feature not meeting the requirements) and an enhancement request. These reviews with the clients over a period of time will help in establishing this difference.

The last reason for a post implementation meeting will depend on how well the implementation and review have gone to this point. There should be no surprises at this point because we have been communicating all along. If it has gone well, this is the perfect time to ask for a letter of recommendation. Having positive testimonials on the company Web site and in the company portfolio will benefit future business opportunities. Take this opportunity to request the letter of recommendation. Most customers I have worked with are more than accommodating.

Bug Tracking

A perfect world would not have applications released with bugs, but the complexity of software today sometimes makes it almost impossible to do so. Customers are shy about reporting what they think are bugs because they feel they have done something wrong to break the program. Others are more than willing to pull your chain every time they find something that goes wrong.

Tracking issues will assist you in a number of ways. First it will better the software you deliver to the customers which in turn improves customer relations. Second, it will provide you a list of things to work on for the various clients. Third it provides the developers with a training mechanism for better development and testing techniques.

Fatal error bugs, ones that are unexpected and ones that our customers find ways of producing, are fairly easy to track. Using the `object.Error()` method or a global error handler expose a chance to collect specific application state information. Typical information that I have collected in my error trapping include: the date and time the error occurs, the error number, the program or object method, the user login id, line number, the `MESSAGE()` when it crashed, code that caused the error, the results from `AERROR()`, the call stack, `LIST MEMORY`, `LIST STAT`, hardware configuration, the contents of `CONFIG.FPW`, the environment settings, information about the various data sessions, information about all the active forms, and any user comments. It is important to provide a mechanism for the users to transmit the collected problems. Options include a report that formats the information collected. Another option is to print the report to PDF format and attach the PDF file to an email and have it sent to your support email address. The most recent idea I have is to transform the error information tracked in the error table into XML and have the XML sent via email. The advantage of this method of transfer allows you to import the information directly into the support database.

So what do you do for the non-fatal errors? This is the kind of bug that the user reports that allegedly does not meet the requirements or when the application fails to operate as they expect. It is important that the user specifies a number of key elements. The elements that I like to have reported are the exact steps to reproduce the error, what was observed, what was expected, additional comments, version of the application, the date the error occurred, and who is reporting the error. Additional information that can be helpful in these cases is the machine configuration. The same mechanism to transport fatal errors can be used to transport user reported issues.

Back at the bat cave you will need to have a mechanism to track reported bugs. There are a number of ways to handle this including pencil and paper, a Word document, or more likely a software package dedicated to tracking bugs. Doing a web search revealed more than a dozen different bug tracking solutions. The BugTrackingSoftware topic on the Fox Wiki links more than a half dozen products that are available. Visual Studio 97 (which VFP 5 was included) includes a solution called Anomaly Tracking System (ATS) written in Visual FoxPro. Visual Studio 6 includes a web version called ATWeb. One nice thing about being database developers is that we can augment the ATS products or build a solution that meets our needs if a solution cannot be found.

Other options or implementations that can be used or modeled after include the new Customer Service Center from F1 Technologies (makers of Visual FoxExpress), BugCentral.com (based on VFP and WebConnect), and Steven Black's Wiki (another VFP and WebConnect implementation). Office XP automatically send statistics back to Microsoft via the Internet when ever a GPF or other error occurs. Am I the only developer in the world that gets some pleasure each time this happens and hoping that some Microsoft web server is getting bombarded with the same reports?

Once a method to track bugs is implemented you need to evaluate the reported errors and alleged bug reports rapidly. It is important to determine which are real and determine the priority assigned to the issue. This will determine the order the bugs are to be corrected. The reported issues that are determined to not be real bugs might be a change in requirements or something that needs to be addressed in training. Some reports are questions to be addressed in the Frequently Asked Questions (FAQ) list.

Post-Mortem Review

One of the keys to doing a better job the next time is learning what you did well this time, how you might have missed the goal, partially failed, or even completely failed to satisfy the customer on this project. Learning from our mistakes is something we learn very early in life.

We believe that the best way to start this is to do a self evaluation, a post mortem review. Each staff member (as little as one, as many as all) needs to participate in this evaluation no matter how much experience they may have or how much they participated in the project. The evaluation contribution will be heavily influenced by a number of factors including experience, length of service, type of work they did on the project, and how familiar they are with various processes involved with software development. There is no “one-size-fits-all” list of questions for a project post-mortem, but here is a starting point that might generate more ideas of what type of information can be gathered from this evaluation.

Please answer all questions that are appropriate, provide specific examples:

- Overall, what did we do well?
- Overall, what did we not do well?
- Were the specifications accurate?
- How much of the original specifications changed during the construction/testing phases?
- Did the developers do unit testing effectively?
- How many bugs were reported by the quality assurance internal testing?
- How many bugs were reported by the customers during acceptance testing?
- Did we follow the implementation check list? What problems did we come across and how can we improve the check list?
- Which processes need improvement and how can they be improved?
- What was the customer relationship before the project started? How is it now that the app is in production?
- What obstacles were put in place by the customer, which were added by your team mates? What obstacles were not removed by management?
- What did we do that wasted the most time and how can we avoid this next time?
- What did you most like doing during the application creation and deployment?
- What did you really dislike?
- What was the one outstanding thing you learned during pair programming or code reviews?
- What types of metrics did we collect on the project and how do they compare to previous metrics from previous projects?
- Were there any heroics performed on this project that need to be recognized?
- If you were forced to vote one employee out of the company, who would it be? (okay, now I'm just checking to see if you are still paying attention <g>)

So developers reading this might be asking the question: now that I have all this paperwork filled out, what do I do with it? Read each and every comment made on the survey. Digest the key pieces of information then select a team of people to gather and discuss these findings. This could be one person or the entire team. Bring them into a room filled with food and open up the discussion. Throw the key points up on a slide and get the discussion rolling. Use this session to brainstorm. It is my experience that developers will talk freely when they know it is in the

interest of the company to improve (and they are pumped with free food). Keep the discussion focused on the positive and make sure all the points are noted that are key to improvement.

These points need to be written down and published to the entire staff so all can benefit from the discussion. Other items to note will be proposed changes to company development standards, development processes, administrative processes, and items in the employee handbook, customer change request forms, and the company brochures/portfolios/Web site.

There is no more demoralizing behavior in a company than spending time on something like the post-mortem, then to have nothing done with the results. I have experienced this more times than I care to count in recent years at very large companies and smaller ones. It doesn't take too many of those to get folks rolling their eyes anytime something like that is planned, and resenting the "window dressing" that is trying to make the company look like it's progressive and "on the ball". It becomes a joke and more damaging than if they do nothing.

Developer Review

It is important to meet with the development staff to make sure they are working towards the goals of the company, the goals of the customers, and their personal career goals. All too often developers wonder what the management thinks of their performance. I have always believed that a performance review should never reveal any surprises. That it was nothing more than an open discussion of the state of the projects, to establish new goals, and to find out what the next aspirations are for the developer.

Take this time to have peers recognize the good (as well as provide some constructive criticism). You might learn from the feedback something that will help you determine the share of the bonus pool.

Post Release Party

Large projects, small projects, super successful projects, and client saving projects are definitely worthy of a party when teams exceed customers' expectations. This is a good place to hand out bonuses and other reward mechanisms. You do not have to wait until the project is complete. Sometimes it is good to do a mid-stream gathering just to break up the stress that can build up during long development stretches. I find that inviting customers to a get together is a great way to build partnerships.

These parties can be held at a local eatery, as a barbeque in a backyard or local park, or right in the office. Just take the time to schedule a break to reward people for their hard work. Take a moment to verbally express the fact that you appreciate the hard work the staff (and clients) are doing to accomplish the goals established.

Bonus Pool

There are simple concepts of keeping people happy. Make sure they have adequate salary, decent benefits, current technology, good projects, a friendly atmosphere, and a fridge stocked with their favorite beverages. One way to make people walk out the door and get a job at the competition is to overlook their value to the organization. When people exceed the expectations

that you have for them, you in turn need to exceed their expectations in the complete compensation package. One way to do this is to bonus the people who made it all happen.

Plan in advance what the bonus pool can be for service over the call of duty. Do not reward people equally unless they contribute equally. Reward for their contribution. If an employee does the nine-to-five and others work a boatload of overtime, make sure the people working longer are not working inefficiently. If others are working effectively and still put in the hours to make the deadline, make sure they are appreciated. Also, make sure those with families have an understanding that you appreciate the sacrifices that the families have made as well.

You can be creative as well. Cash is not the only mechanism to reward the staff. For instance, if you have a developer who is a space geek you could send them to SpaceCamp for a weekend (hint, hint to my partners <g>). Sending the staff (and family) on a weekends away at a nice resort is a great way for people to get recharge their batteries before returning to work. One of my favorite ways to reward people is to grant them more vacation time. Top gun developers are typically willing to work late into the evenings and on weekends, why not give them back their time without any additional taxes to pay?

Single developer shops need to make sure they take care of the proprietor as well. This is so often overlooked when you have to run a business day-to-day. Make sure you take a cut of your spoils and keep the boss happy.

Do not underestimate the value of the employees. A long time ago, way back in the 1980's I worked at Burroughs, at the time the third largest computer manufacturer in the world. This was my first big paying job out of college. Jumped right from a small consulting company I worked for during my last year in college to the big time systems management group as an Associate Systems Analyst. I was still wearing the rose colored glasses when it came to viewing the corporate world.

Along came a multi-billion dollar merger. Money was flying as two 5 billion dollar companies merged into one. Our group was tasked to merge the corporate systems. A project plan was assembled in November with a completion target date of June. This was an aggressive schedule to start and we were immediately put on mandatory overtime (without overtime pay). On February 1 we were informed that we needed to have our system up and running in production on April 1st (2 months earlier than originally "planned"). This meant it needed to be in the test environment for acceptance testing March 1. Management promised that we would be rewarded for all our efforts and that they knew what it was going to take to accomplish this. We had meetings at 5:00 every afternoon to see what the developers would be doing that evening. Then the managers headed for the golf course.

I worked over 100 hours of overtime in February alone and we implemented the system on time, with very few issues. Months later I had to walk into the Directors office to find out what happened to the bonuses that were promised. He just bought a fancy new luxury car and a huge house valued at 5 times what mine was worth. It was apparent where the bonus pool was distributed. He finally gave in to our "concerns" and threw a dinner party at a local restaurant. That evening he handed out checks to the developers for \$125.00 (minus taxes, I saw less than \$100). This was less than 50 cents an hour for my effort over the length of the project. Guess where my loyalties went after this? Do not make this same mistake.

Conclusion

It might be important to note that there are always new ideas and improvements that can be made on the deployment process. If we learn from our mistakes (and those made by others) we will add steps or review our checklists to make our lives easier. We all get better and our clients benefit with smoother implementations and we will likely sleep better as developers knowing that the future deployments will go well.

There are a number of options when creating a deployment strategy. In this white paper we covered different techniques for extracting version information from the executables, different strategies for developing install packages, and covered the one install builders included with Visual FoxPro and some tips and gotchas when working with this tool.

Special Thanks

I want to thank the user groups that put up with the rehearsals to insure that this presentation was refined for primetime at the Great Lakes Great Database Workshop. The [Greater Cleveland PC User Group \(FoxPro SIG\)](#) and the [Detroit Area Fox User Group](#) members provided excellent feedback to me and I really appreciate the frank and honest evaluations that were provided.

I want to thank Patty Nowak, editor extraordinaire, who reviewed some of these notes before publication. Steve Dingle, tech editor of *MegaFox: 1002 Things You Wanted To Know About Extending Visual FoxPro* reviewed all the material in Chapter 11: Deployment, which some of this material was borrowed.

Copyright

Copyright © 2001-2002 Richard A. Schummer. All Worldwide Rights Reserved

Author Profile

Rick Schummer is a partner at Geeks and Gurus, Inc, a full service computer solutions provider in Detroit Michigan, USA, empowering organizations through technology. After hours he enjoys writing developer tools that improve his company's productivity and occasionally pens articles for his favorite Fox periodicals and user group newsletters. Rick is a co-author of MegaFox: 1002 Things You Wanted To Know About Extending Visual FoxPro and 1001 Things You Always Wanted to Know About Visual FoxPro, and a founding member and Secretary of the Detroit Area Fox User Group (DAFUG). He is a regular presenter at user groups in North America, at GLGDW 2000-2002, at Essential Fox 2002, and VFE DevCon2K2 conferences. raschummer@geeksandgurus.com, rick@rickschummer.com, <http://www.geeksandgurus.com> and <http://www.rickschummer.com>

Appendix A: InstallShield Express - Visual FoxPro Limited Edition Tips

DeploymentDemo.ism

The full name of the replacement for the Setup Wizard used in previous versions of Visual FoxPro is: Install Shield Express – Visual FoxPro Limited Edition. When we hear “limited edition” it is usually used in the context of something special that might be collectible or cherished. In the case of InstallShield Express, Limited Edition means that it has a limited feature set. It is the “lite” version of InstallShield Express that is available from InstallShield Software Corporation.

InstallShield Express is a tool that builds installation routines that run on the Windows Installer technology included with Windows. It uses a more modern interface than the old 16-bit installer that the Setup Wizard generated.

Installing InstallShield Express

InstallShield Express – Visual FoxPro Limited Edition is included on the Visual FoxPro CD-ROM. It is not automatically installed when you select all the files when installing the VFP. It is a separate installation in the same CD. You need to select the Install InstallShield Express option from the Visual FoxPro install startup screen.



Figure 10. InstallShield Express – Visual FoxPro Limited Edition installation is started from the Visual FoxPro 7 installation start up screen.

Advantages of using InstallShield Express over the Setup Wizard

InstallShield Express (ISE) is a big step forward in flexibility and is a full 32-bit application. It provides a number of features long desired by developers who have used the tried and true Setup Wizard.

The first advantage is that you no longer need to copy all of the files that are distributed in the build into a separate directory. ISE lets you specify which files are to be distributed, and where on the target system they go.

The Setup Wizard allows only for all files or no files. You did not have a choice in the matter. InstallShield Express allows the user to pick what "features" they want installed. Much like the typical options: typical, all, or custom. Picking custom will allow the user to further tailor what is installed. You can define what these options are and what files will be installed when the option is specified.

InstallShield Express provides generic references to all of the Windows folders. The Setup Wizard allowed developers to install files to the application folder and the Windows System folder (primarily for FLLs, DLLs, OCXs, and the Visual FoxPro runtimes). Now you have references like INSTALLDIR, DATABASEDIR, and ProgramFilesFolder to direct files to predefined folders based on the folder structure used on the user's machine. See the section on "How do I leverage the default Windows directories?" in this white paper for more details.

A dynamic setup mechanism allows the developers to select the screens used in the setup. This allows you to display pages for a license agreement, a ReadMe text, the entry of the user name and company, where the installation files are located, where the data is located, provide for a custom setup, and determine what is on the setup complete dialog. The Setup Wizard only allowed you to customize the name of the application and copyright information.

InstallShield will not only install selected ODBC drivers, but will install pre-built datasources (DSNs) as well. With the Setup Wizard you needed to write custom code with calls to the Windows API and have this executed as a post-install routine.

InstallShield knows where the Windows font folder is and can install fonts that you need to install with your application. It has built in capability to store things to the Windows' registry, modify INI files, and set up file extension associations. Again, with the Setup Wizard you needed to write custom code with calls to the Windows API and have this executed as a post-install routine.

What are the disadvantages of using InstallShield Express vs. Setup Wizard?

At first glance the InstallShield Express product looks incredible and a giant leap forward. There are a couple of show stoppers that makes one think it is not a complete replacement product for the old-fashioned Setup Wizard included with Visual FoxPro 3, 5, and 6.

There is a feature called Upgrade Path. This feature is only available in the full edition of InstallShield Express, not the in limited edition. The Upgrade Path feature is a way to configure how the second installation of your product is going to run. Will it replace files, update only newer files, and determine which versions it will upgrade. When you run a different build of the custom InstallShield it prompts you with a message:

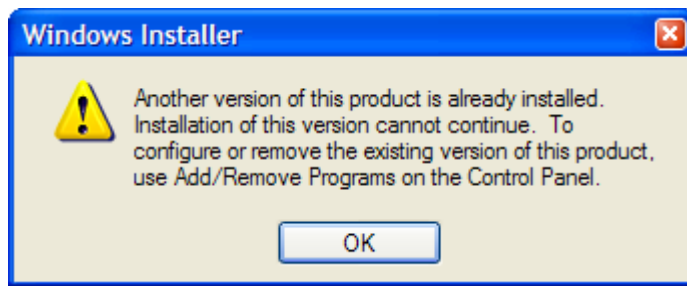


Figure 11. InstallShield Express – Visual FoxPro Limited Edition installations require you to remove the previous installation before reinstalling.

This means that the users will uninstall the executables, the shortcuts, data, remove registry entries, and any other item that was installed with the previous install. The Setup Wizard allowed for reinstallation (complete overwrite) or removal of the previous installation. This one “feature” makes the entire product absolutely useless unless you only intend on the custom product shipping once or it has no negative effect on uninstalling all the files before reinstalling the upgrade. This is different from reinstalling the current version. InstallShield does provide a mechanism to modify and repairing existing installs for the same version.

The other feature that is missing (actually it is one of the “limitations” of the Limited Edition) is that it cannot run a post installation process. While many reasons we ran post-installation executables have been integrated in the base tool (registry updates, creation of shortcuts), we still need processes run to make database structural changes or data conversions. The users can manually run these routines, but it is something they will need to remember using an InstallShield Express routine.

How do I upgrade to the full version of InstallShield Express?

The quick answer is that you cannot “upgrade” the InstallShield Express – Visual FoxPro Limited Edition to the full version of InstallShield Express. You will need to purchase a full license. InstallShield Express was provided courtesy of InstallShield via Microsoft as a basic method to installing your custom applications, not as a full upgradeable license.

How do I leverage the default Windows directories?

A big advantage to using InstallShield Express (and other commercial install packages) is the ability to place files in different directories. InstallShield also provides a list of Windows folders through generic references. InstallShield converts the references into folders by reading the operating system during the actual installation. This eliminates any hard coding of paths.

Table 1. Optional Windows Folders available in InstallShield Express.

Folder Variable	Description
AdminToolsFolder	Points to the folder where operating system administrative tools are located, obtained from the operating system.
AppDataFolder	Full path to the current user's Application Data folder, obtained from the operating system.
CommonAppDataFolder	Full path to the folder containing application data for all users. An example in Windows XP is C:\Documents and Settings\All Users\Application Data, obtained from the operating system

Folder Variable	Description
CommonFilesFolder	Full path to the Common Files folder for the current user, obtained from the operating system.
DATABASEDIR	Destination for your setup's database files. You can set the initial value for DATABASEDIR and have the end users modify this value during the installation in the Database Folder dialog.
DesktopFolder	Full path to the Desktop folder for the current user unless the setup is being run under NT/2000/XP for All Users, and the ALLUSERS property is set, then the DesktopFolder property should hold the full path to the All Users Desktop folder, obtained from the operating system.
FavoritesFolder	Full path to the Favorites folder for the current user, obtained from the operating system.
FontsFolder	Full path to the Fonts folder, obtained from the operating system.
INSTALLDIR	Destination folder for your setup. You can set an initial value for INSTALLDIR and have the end users modify this value during the installation in the Destination Folder dialog. This property can be set using any of the other system folders.
LocalAppDataFolder	Locally stored application data, obtained from the operating system.
MyPicturesFolder	Full path to MyPicturesFolder, obtained from the operating system.
NetHoodFolderProperty	Full path to the current user's Network Neighborhood folder, obtained from the operating system.
PersonalFolder	Full path to the current user's Personal folder, obtained from the operating system.
PrintHoodFolder	Full path to the current user's Printer Neighborhood folder in Windows NT/2000/XP, obtained from the operating system.
ProgramFilesFolder	Full path to the current user's Program Files folder, obtained from the operating system.
ProgramMenuFolder	Full path to the Program menu for the current user. If the setup is being run under NT/2000/XP for All Users, and the ALLUSERS property is set, then the ProgramMenuFolder property should hold the full path to the All Users Program menu, obtained from the operating system.
RecentFolder	Full path to the current user's Recent folder, obtained from the operating system.
SendToFolder	Full path to the current user's SendTo folder, obtained from the operating system.
StartMenuFolder	Full path the Start menu folder for the current user. If the setup is being run under NT/2000/XP for All Users, and the ALLUSERS property is set, then the StartMenuFolder property should hold the fully qualified path to the All Users program menu, obtained from the operating system.
StartupFolder	Full path to the Startup folder for the current user. If the setup is being run under NT/2000/XP for All Users, and the ALLUSERS property is set, then the StartupFolder property should hold the full path to the All Users program menu, obtained from the operating system.
System16Folder	Full path to the folder containing the system's 16-bit DLLs, obtained from the operating system.
SystemFolder	Full path to the Windows system folder, obtained from the operating system.
TempFolder	Full path to the Temp folder, obtained from the operating system.
TemplateFolder	Full path to the current user's Templates folder, obtained from the operating system.
WindowsFolder	Full path to the Windows folder, obtained from the operating system.
WindowsVolume	Volume of the Windows folder. It is set to the drive where Windows is installed, obtained from the operating system.

The advantages of dynamic paths are obvious. The implementation of the folder variable is handled in InstallShield by entering the value for the property. You can concatenate more than one folder variable if necessary (see **Figure 12.** for DATABASEDIR properties). Just be careful because many of the folders are fully pathed.

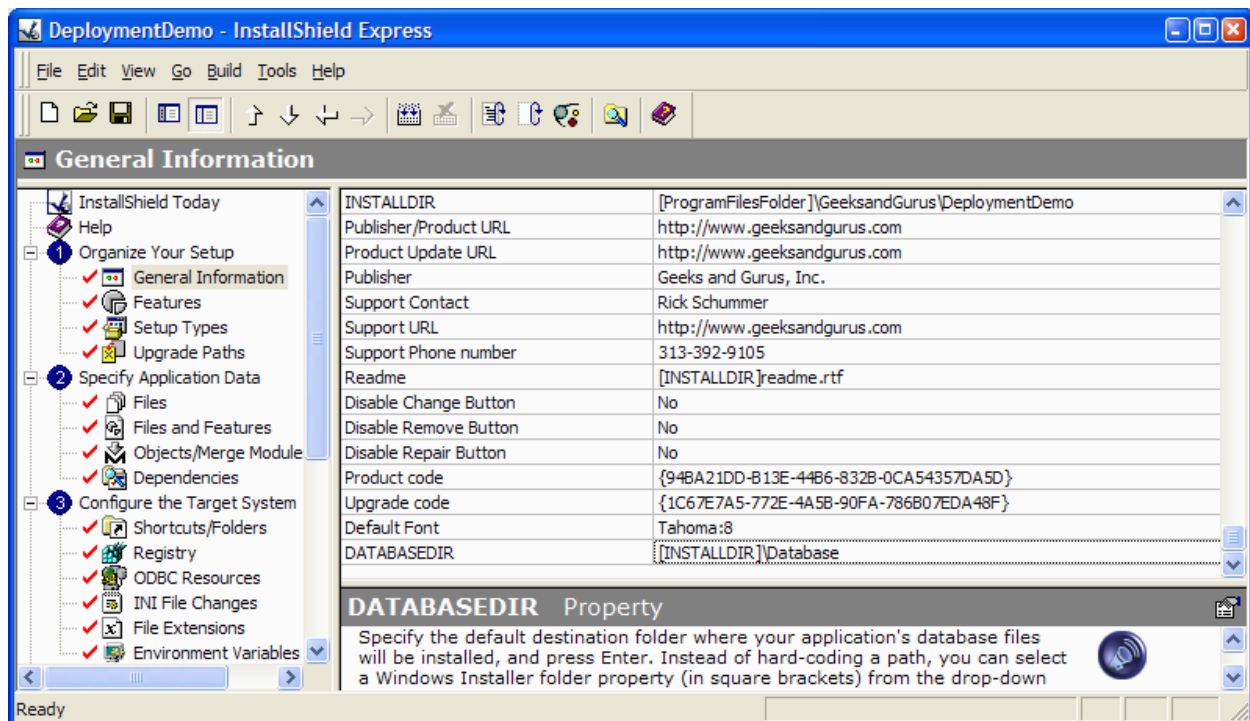


Figure 12. You can leverage multiple Windows Folder properties assigning any directory property.

In the General Information section you can specify the developer defined `INSTALLDIR` and `DATABASEDIR` (defaults for the installation directories, changeable by the installers). The Files section allows you to add any of the Windows folders by right-clicking on the Destination Computer, and selecting the Show Destination Folder. More than one folder can be added by repeating the selection. The Shortcut/Folders section allows you to add shortcuts to the Start Menu, the Programs Menu, the startup folder, the desktop, or a custom menu. The shortcuts have a Target and Working Directory property. These properties accept any of the folder variables. Registry values and the INI file Target property can utilize the folder variables as well.

How do I work with Setup Types and Features?

Features are file bundles within the install package from the user's perspective. Setup Types are predefined categories that are assigned features. The user will select a Setup Type and behind the scenes the files associated to the Setup Type through the defined features are installed.

The setup types default to Typical, Minimum and Custom. If the user selects the Custom option they will be able to pick and choose features that you have included. You can change the captions of the setup types by right-clicking on the setup type to bring up the context menu with the rename option. You will not be able to add new setup types. The first one in the list will be the initially selected option, so move the options around if you prefer a different default or different order.

The features represent a function, capability, or component of your application and have much flexibility. You can add new features and subfeatures. The assignment of files to the features is made in the Files and the Files and Features sections. The Always Install feature cannot be removed and is included in every install project. It does not show up on the Custom install either.

It is designed to always run, regardless of which setup type is selected by the user. Here are some sample ideas for ways you can configure features:

- Application (includes the metadata), Data, Runtimes
- Application, Data, Help, Reports
- Executable, Source Code

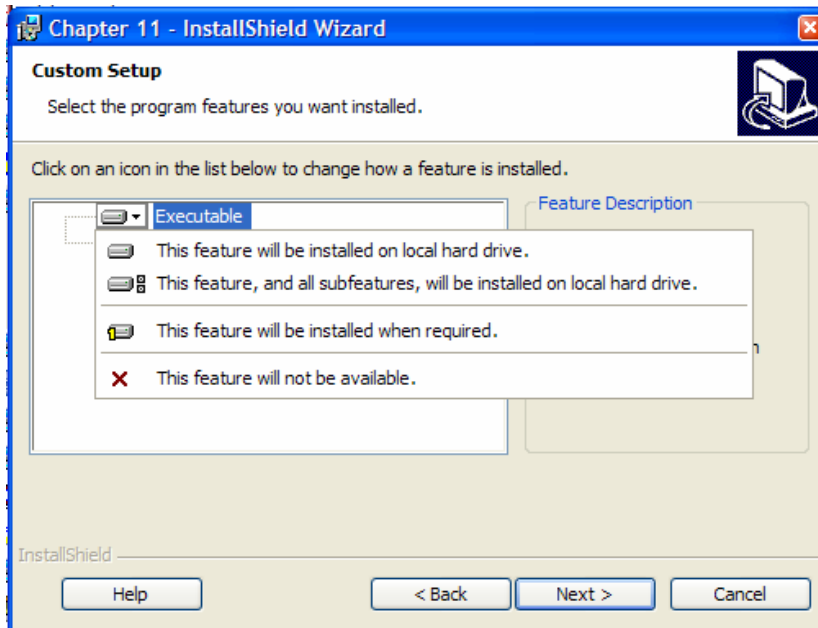


Figure 13. The user will be able to select which features are installed if they pick the Custom setup type.

If the Custom setup is opted by the user they can customize which features are installed and how they are installed (see **Figure 13.**). The user can determine if they want a specific feature installed, not installed, or opt to have it installed at a later time. There is no way to customize the options on the dropdown.

What is a merge module and which do I use for Visual FoxPro installs?

A merge module (MSM file) contains all the files needed to install an application, component, runtime files, or other functionality. All the necessary logic and registry entries are also included to direct the installer routine. These merge modules save you the time of selecting all the individual files included in the merge modules and figuring out the dependencies of other files that they require.

The merge modules are supplied with InstallShield Express. The InstallShield Express product cannot create or alter merge modules. You need the InstallShield Developer edition to create or update the merge modules. Updates to the merge modules will be provided by InstallShield or the other software manufacturers. For instance, Visual FoxPro 7 Service Pack 1 shipped new runtime merge modules.

InstallShield Express comes with a number of merge modules for Visual FoxPro developers to include in their custom installations. The merge modules can be selected in the Objects/Merge Modules section. All you have to do is click on the checkboxes for each of the modules you

would like included in your custom install. This allows you full control over how much or how little is included in the installation outside of the specific files you picked in the Files section.

A minimum install should include the Visual FoxPro 7 runtimes libraries and the Microsoft Visual C++ 7.0 Run-Time Library. There are a number of merge modules to select from for the Visual FoxPro runtimes. Which ones you select will depend on the languages you support. Minimally you will need to check the Microsoft Visual FoxPro 7 Runtime Libraries (VFP7RUNTIME.MSM) and the Microsoft Visual C++ 7 Runtime Libraries (MSVCR70.MSM).

If you support a language other than English then you also need to select the appropriate resource library. The VFP7RUNTIME.MSM file includes the VFP7RENU.DLL, which is used for all English shipping applications. If you want to include support for another localized resource file (VFP7RXXX.DLL), include the merge module containing the localized resource file. For example, include VFP7RDEU.MSM for the German run-time resource file. You will need to look in the merge module description pane (middle, bottom in InstallShield Express) to read the merge module file name.

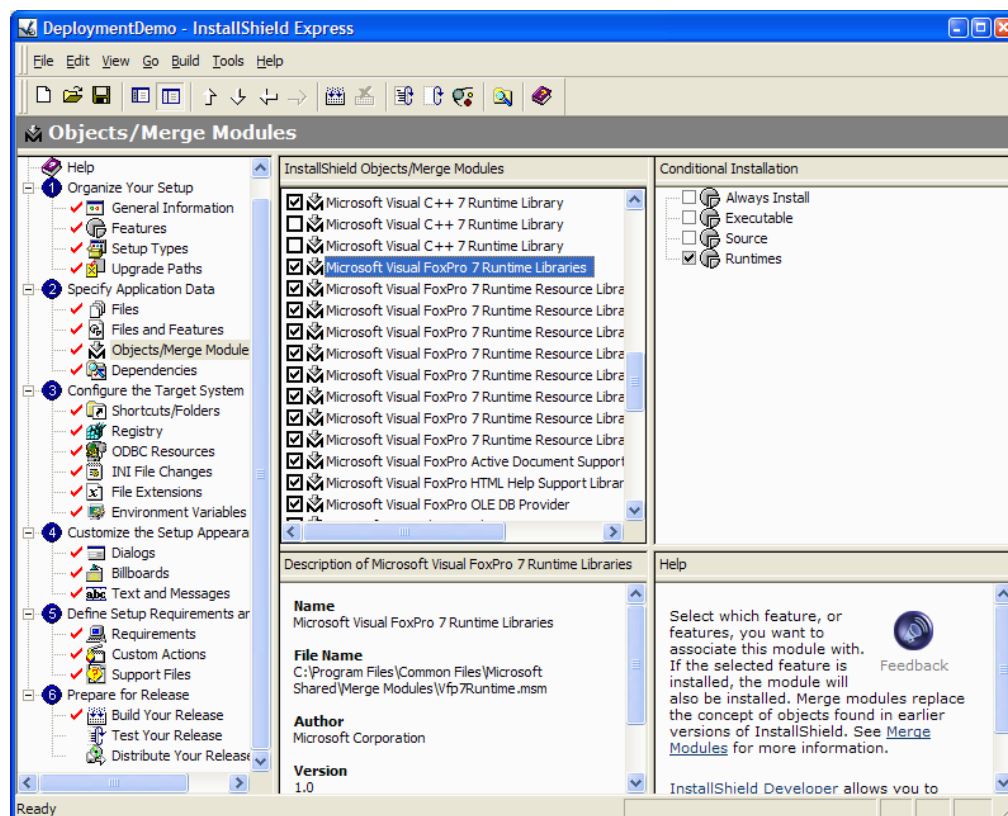


Figure 14. Select Visual FoxPro runtimes and VC++ runtimes for a minimum Visual FoxPro custom application installation.

So why do you need to include the MS VC++7 Runtime Library? To avoid getting a call from a customer who just installed the latest version of your application. When they fire up the application for the first time they would see a message "msvcr70.dll not found". This is a common first time InstallShield Express installation mistake. It is a problem easily missed unless you are testing on a machine that had no previous Visual FoxPro installation. This is one example of why it is nice to have a clean machine to test your installations.

There are three other specific Visual FoxPro merge modules. The Visual FoxPro OLE DB provider makes it possible for both Visual FoxPro and non-Visual FoxPro applications to access Visual FoxPro data using OLE DB or ActiveX Data Objects (ADO). To install the Visual FoxPro OLE DB provider on the customer's machine, include the Microsoft Visual FoxPro OLE DB Provider (VFPOLEDB.MSM) merge module. The older Visual FoxPro ODBC driver is still available for installation via the VFPODBC.MSM merge module. The Microsoft Visual FoxPro HTML Help Support Library (VFPHTMLHELP.MSM) merge module includes both FOXHHELP.EXE and FOXHHELPPS.DLL files needed to support HTML Help within your custom Visual FoxPro applications. In addition to your application-specific .chm file, you might have to include the core HTML Help viewer files.

If your application uses Web Services or the Simple Object Application Protocol (SOAP), you must include the following merge modules:

- SOAP SDK Files (SOAP_CORE.MSM)
- Visual Basic Virtual Machine (MSVBVM60.MSM)
- Microsoft Component Category Manager Library (COMCAT.MSM)
- Microsoft OLE 2.40 (OLEAUT32.MSM)

There are merge modules for the ActiveX controls that ship with Visual FoxPro and previous versions of Visual Studio, as well as the various Microsoft data access technologies. Third-party control and COM providers may also include merge modules with their products for you to include as part of the installation routine. Updated and new merge modules should be available on the InstallShield Web site.

How do I create shortcuts or folders?

InstallShield Express makes it possible for you to create shortcuts and folders both in the Start menu and on the desktop. In addition, shortcuts can be associated with the features that you defined earlier.

- Navigate to the ShortCuts/Folders section.
- From the Shortcuts TreeView in the center pane, right-click the node where you want to install a shortcut or folder, and click New Shortcut or New Folder.
- Type a name for the item you created, this will be the caption for the shortcut.
- If you created a shortcut, you must specify the Target. In the Shortcut Properties pane (right most), select the Target property, and then select a target folder from the combo box. You can add your own file name to the end of the Target folder selected from the list. The Description property will translate into the tool tip for the shortcut in Windows 2000/ME/XP. You have the option to associate your shortcut with a feature. Select the Feature property, and then select a feature from the combo box. The Icon File can be an ICO or EXE with the number of the stored icon in the EXE selected being saved in the Icon Index property.

How do I create registry keys?

If your application uses registry keys to keep track of user options or application settings, InstallShield Express can add them to the user's machine during the installation. It should be noted that creating registry keys is an optional step in creating a setup program.

Registry entries are created in registry hives that categorize registry entries by function. For example, software options, such as options for Visual FoxPro or your custom application, are contained in the `HKEY_CURRENT_USER` hive under `Software\<CompanyName>\<ApplicationName>` while COM server classes are contained in the `HKEY_CLASSES_ROOT`.

If the registry entries exist on the development machine it is as simple as dragging and dropping the registry entry from the Source Computer's Registry View to the Destination Computer's Registry View. We recommend that you follow the identical hive configuration because it is likely that your custom application will be looking for it in the same place on the destination computer.

If the keys do not exist on your development machine, you can either create them by hand using RegEdit or the InstallShield interface, or programmatically with Visual FoxPro or another tool. These registry entries will be available in the top pane (source computer). If you want to manually create the registry entries in InstallShield, here are the steps to follow:

- Right-click the registry hive on the Destination Computer's Registry View (bottom pane).
- On the Context menu, click New | Key and type a name for the key.
- Right-click the new key.
- On the Context menu, click New, and then select the type of value you want to add to the key.
- Double-click on the new value and enter in the initial value for the entry.

How can I limit the hardware configurations the app will install?

InstallShield provides a number of configuration checks that will stop a user from installing your application if their computer does not conform to the required specifications.

The first check is checking to make sure the operating system (OS) meets requirements. This option allows you to pick all operating systems (not placing restrictions), or pick and choose which operating systems are acceptable. There is one problem with the initial release of InstallShield Express – Visual FoxPro Limited Edition, if you select the operating systems, there is no option for Windows XP. Guess what, it will not allow an install on XP unless you allow all OSes.

The processor option allows you to select all processors, 486, or Pentium or higher. We know that Visual FoxPro will not work on a 486, so we encourage this option to be set at Pentium or higher.

The RAM options allow you to specify the lowest amount of RAM that allows the application to be installed. We recommend that you set this to the Visual FoxPro minimum, which is 64MB.

The screen resolution and color depth options are very personal settings. We have known users over the years that refuse to move past 640x480 no matter how large a monitor they use.

Restricting the screen resolution should be negotiated in advance since there are laws that regulate accessibility issues in many countries.

How do I have the install files registered for all users of the computer?

There is a quirk in the initial release of InstallShield Express – Visual FoxPro Limited Edition that does not automatically load the installed application for all users on Windows. This happens if the installation does not use the Customer Information Dialog.

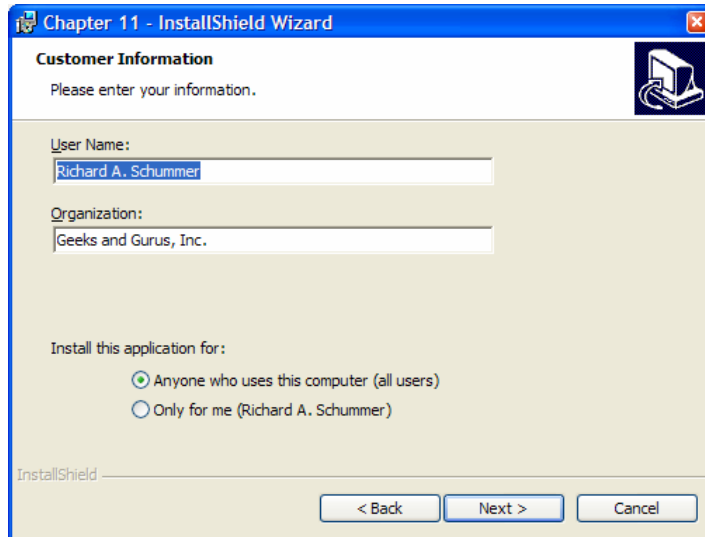


Figure 15. The Customer Information Dialog allows the user to determine if the install is completed for all users on the computer or just for the user that installs it.

The natural workaround is to include the Customer Information Dialog in the install routine. If you run into installs that were built and shipped and it is not cost effective to re-release the application you can still work around this issue by adding a parameter to the Setup.exe:

```
setup.exe /V"ALLUSERS=1"
```

There are three values that can be set for the ALLUSERS property. ALLUSERS=NULL (default value) will install the package for the current user. ALLUSERS=1 will install the package for all the users on the machine provided the user has administrative privileges. If the current user running the setup on Windows NT/2000/XP does not have admin privileges, then the setup will error out and abort. ALLUSERS=2 will check the user's privileges to see if they have admin rights. Pending the outcome of this check, it will install for all users if the user has enough admin privileges otherwise just install it for the current user.

NOTE: There are a number of excellent tips like this one available on

<http://support.installshield.com/kb/>, <http://support.microsoft.com/kb/> and <http://fox.wikis.com/>