Chapter 8
Release and Post-Release Tips

Release day is finally here! The process of moving an application into production is the end of a long road traveled. Depending on the project, you might spend hours, weeks, months, or even years of blood, sweat, and tears getting ready for this big event. 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 smoothly, or not, depending on the planning and attention to detail.

This chapter outlines some things to verify as you and your clients move the product into production, ideas on supporting a release, and items to consider after a release is completed.

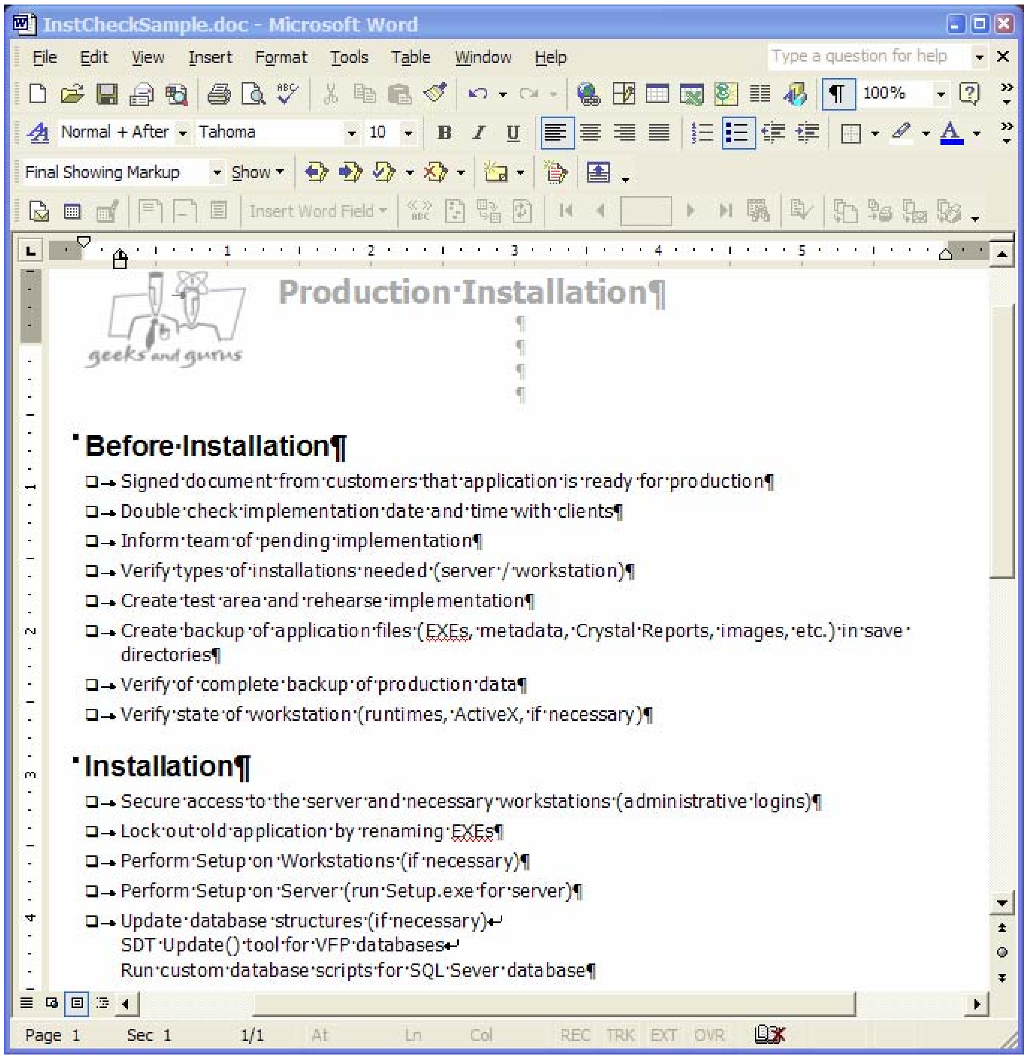
Checklist/Instructions

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

As we noted throughout this book, installation software is much more sophisticated today than the DOS batch files 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 the customer will never read it if they perceive it as overwhelming. Make the instructions concise, but informative. The instructions should include all the steps up to running SETUP.EXE and all the steps needed after the setup is complete and before they start using the new application or update.

We included one sample checklist in the chapter downloads called InstallChecklistSample.doc.

Your checklist (see **Figure 1** for an example) may be different for each customer and for each application. It is also influenced by the technologies you are deploying and by the technologies used to create the installation package. The detail you include is influenced by the person performing the deployment and their experience. If you are performing the install you might only need a bullet point to remind you what is next. If your customer has an internal Information Systems (IS) staff or you have someone from your technical support staff making an on-site visit, you might limit the amount of detail. If the end users are installing the application or update they might need step-by-step instructions with supporting screen shots to make sure they get it right.


InstCheckSample.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
Normal + After
Tahoma
10
100%
Final Showing Markup
Show
Insert Word Field
geeks and gurus
Page 1 Sec 1 1/1 At Ln Col REC TRK EXT OVR

Production·Installation¶

Before·Installation¶

- Signed·document·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·(EXEs,·metadata,·Crystal·Reports,·images,·etc.)·in·save·directories¶
- Verify·of·complete·backup·of·production·data¶
- Verify·state·of·workstation·(runtimes,·ActiveX,·if·necessary)¶

Installation¶

- Secure·access·to·the·server·and·necessary·workstations·(administrative·logins)¶
- Lock·out·old·application·by·renaming·EXEs¶
- Perform·Setup·on·Workstations·(if·necessary)¶
- Perform·Setup·on·Server·(run·Setup.exe·for·server)¶
- Update·database·structures·(if·necessary)↵
  SDT·Update()·tool·for·VFP·databases↵
  Run·custom·database·scripts·for·SQL·Sever·database¶

Figure 1. *This is a sample checklist for installing an application in production.*

Test area

Creating a test area has to be the least used but 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 getting an update, copy the base set of files from the current production directories to provide a true experience of what to expect 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 this is a separate area from the beta test area 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 ensure 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 complete 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 with older runtimes, one with 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 configurations at this time, just confirming the implementation goes smoothly.

Backups

Another big mistake we have made in the past is not backing up the files we are changing. This takes an extra few minutes to create a save directory, copy the executables, and any other files you change. These files can later be restored if something goes wrong. This precaution is for those updates you are 110% sure (or possibly less) that the five-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, you have some how destroyed data.

We always feel most comfortable when the user has just backed up the data. If they do not have a current backup we ask they perform a backup before implementation. This is typically a scheduled event so there is plenty of time to ensure it is not only completed but is verified to be a good backup (especially if done to a tape backup). With disk prices as cheap as they are these days we 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 you 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 needed for FoxPro applications to run on computers that do not have a copy of FoxPro installed. The biggest key with the runtime files is they are the same version as the Visual FoxPro version you used to develop the application. This can be trouble if the users have several applications developed over time. Visual FoxPro 6 for instance had new runtimes in several of the five service packs Microsoft shipped. If you developed one application with service pack 3 and another application with service pack 5 you can have different results or behaviors with the two executables sharing the latest runtimes

One issue we did not address is the Windows 2000/XP 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 custom and commercial frameworks have a mechanism to save user preference and application settings to a system table or to the Windows' Registry. New applications do 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 one of the many installation routines we have discussed in Chapter 4: "Setup Tool Roundup" or even the older Visual FoxPro 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 modern 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. If you save settings and configuration items to a table, you need a program to handle the update.

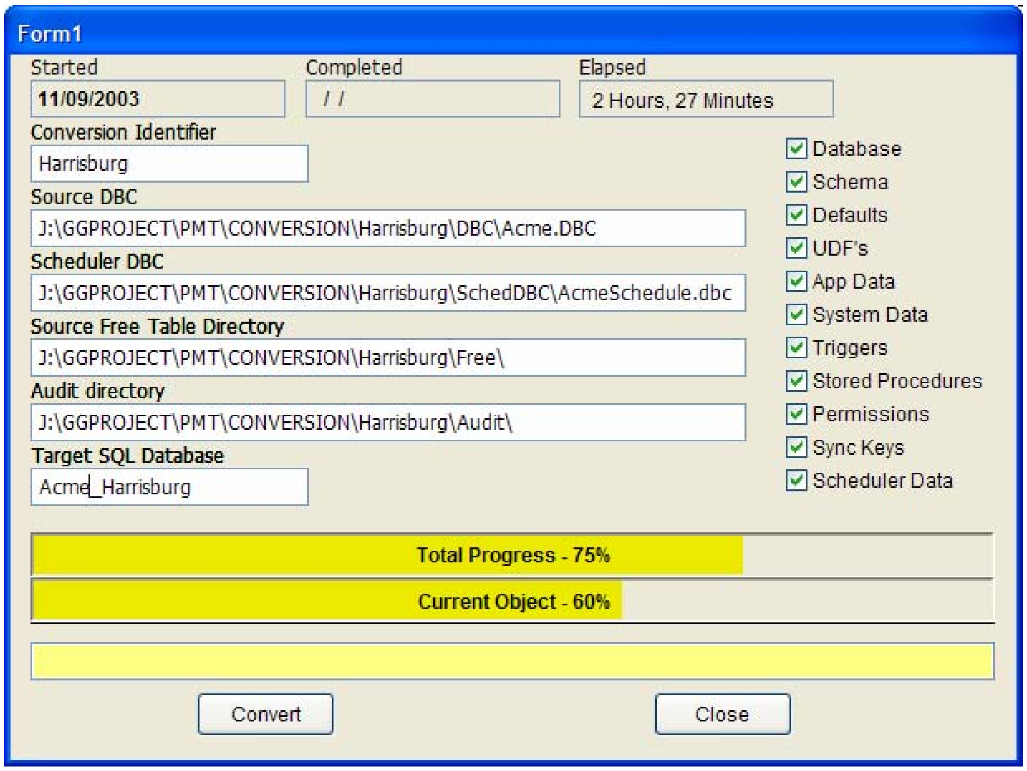
Update structures and indexes

Previous chapters in this book addressed the database updates, table structure changes, and changed/updated indexes. If you are using a Visual FoxPro database container (DBC) you have two basic options. The more difficult way is to track all the changes you made and write a program to make the changes with `ALTER TABLE` and `INDEX ON` commands. You can also write your own routine to 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 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. You need to ensure the users have the ability to run your scripts in the database, or you need to provide your own tool that runs your scripts in the production environment.

Data conversion

Once the table structures are updated, new indexes are added, stored procedures are changed, and various constraints have been made current, you might have to update the data in the database to reflect these changes. You might be installing a brand new empty database. 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 used before the new one was implemented. This conversion is a different and separate step in the process. Obviously this is best handled after the structures are in place.



Started: 11/09/2003
Completed: / /
Elapsed: 2 Hours, 27 Minutes
Conversion Identifier: Harrisburg
Source DBC: J:\GGPROJECT\PMT\CONVERSION\Harrisburg\DBC\Acme.DBC
Scheduler DBC: J:\GGPROJECT\PMT\CONVERSION\Harrisburg\SchedDBC\AcmeSchedule.dbc
Source Free Table Directory: J:\GGPROJECT\PMT\CONVERSION\Harrisburg\Free\
Audit directory: J:\GGPROJECT\PMT\CONVERSION\Harrisburg\Audit\
Target SQL Database: Acme_Harrisburg

- ✓ Database
- ✓ Schema
- ✓ Defaults
- ✓ UDF's
- ✓ App Data
- ✓ System Data
- ✓ Triggers
- ✓ Stored Procedures
- ✓ Permissions
- ✓ Sync Keys
- ✓ Scheduler Data

Total Progress - 75%
Current Object - 60%

Figure 2. *This is an example of a database conversion routine running.*

You might implement this as a post setup executable or process (see **Figure 2** for an example), or you may run this even before implementing the new or updated application. If there are several items you need to accomplish after the application files are copied to the appropriate location you need to write a program that handles all the needed steps.

Test

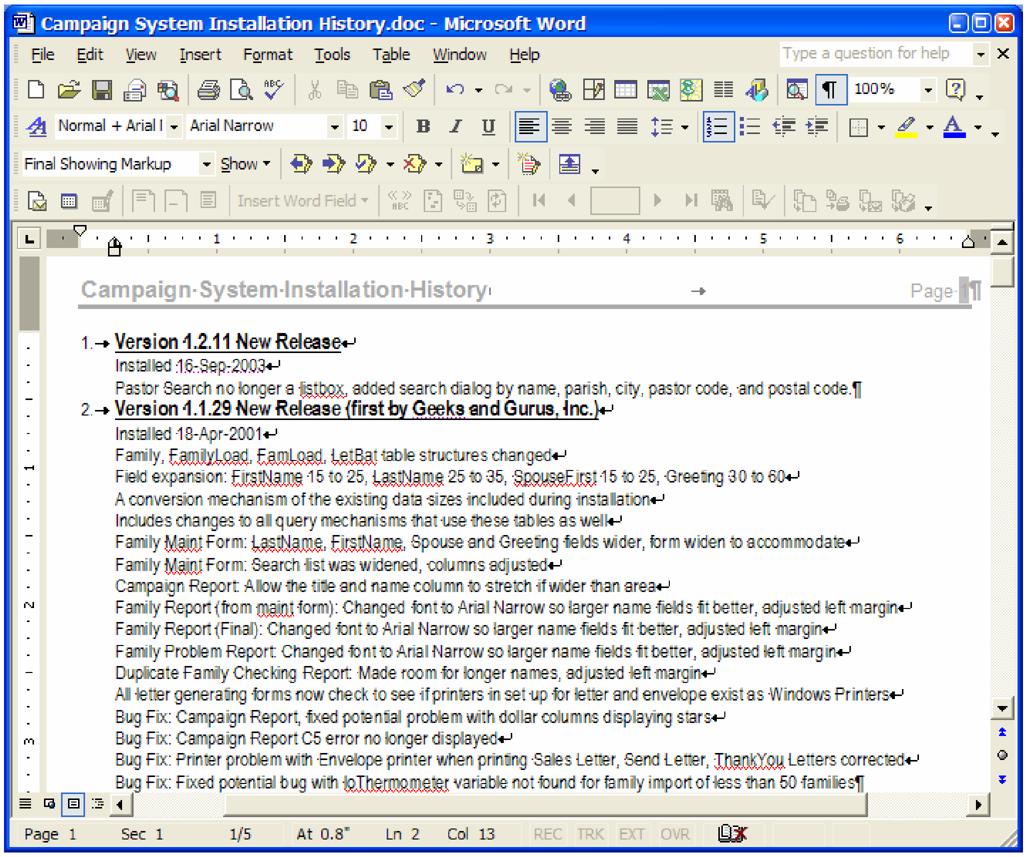
Once the application is loaded and configured it is important for one last test before informing the users the implementation is complete. You are not looking to do a complete system test, just to make sure the executable starts and some primary forms and reports execute correctly, the data converted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

Notify the customer

Finally the moment you look forward to, telling the customer 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 on-site. 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 we open an instance of Notepad and leave a quick message that we are finished with the implementation.

A follow up e-mail and phone call are also important telling them the status of the implementation. Hopefully you inform them all went well and the application is ready for their use. Sometimes things do not go well and you need to tell them 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 we like to provide is a Word document detailing all the changes made, both new features and bug fixes (see **Figure 3**).


Campaign System Installation History.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
100%
Normal + Arial I
Arial Narrow
10
Final Showing Markup
Show
Insert Word Field
Page 1 Sec 1 1/5 At 0.8" Ln 2 Col 13 REC TRK EXT OVR

Campaign System Installation History Page

1. **Version 1.2.11 New Release**
   Installed 16-Sep-2003
   Pastor Search no longer a listbox, added search dialog by name, parish, city, pastor code, and postal code.
2. **Version 1.1.29 New Release (first by Geeks and Gurus, Inc.)**
   Installed 18-Apr-2001
   Family, FamilyLoad, FamLoad, LetBat table structures changed
   Field expansion: FirstName 15 to 25, LastName 25 to 35, SpouseFirst 15 to 25, Greeting 30 to 60
   A conversion mechanism of the existing data sizes included during installation
   Includes changes to all query mechanisms that use these tables as well
   Family Maint Form: LastName, FirstName, Spouse and Greeting fields wider, form widen to accommodate
   Family Maint Form: Search list was widened, columns adjusted
   Campaign Report: Allow the title and name column to stretch if wider than area
   Family Report (from maint form): Changed font to Arial Narrow so larger name fields fit better, adjusted left margin
   Family Report (Final): Changed font to Arial Narrow so larger name fields fit better, adjusted left margin
   Family Problem Report: Changed font to Arial Narrow so larger name fields fit better, adjusted left margin
   Duplicate Family Checking Report: Made room for longer names, adjusted left margin
   All letter generating forms now check to see if printers in set up for letter and envelope exist as Windows Printers
   Bug Fix: Campaign Report, fixed potential problem with dollar columns displaying stars
   Bug Fix: Campaign Report C5 error no longer displayed
   Bug Fix: Printer problem with Envelope printer when printing Sales Letter, Send Letter, ThankYou Letters corrected
   Bug Fix: Fixed potential bug with loThermometer variable not found for family import of less than 50 families

Figure 3. *This is an 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 on site, or by the help desk support staff.

The need for support of a release varies 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 on-site and just need to walk into the next room and walk cubicle to cubicle with the bug fix CD or stop in to answer a question. On the other hand, when we were corporate developers our applications were running in 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 connect into the server via the Internet and inspect the situation.

We also encourage users to inform technical support of problems they are experiencing. We cannot tell you how many times we went on-site to review something with a client and when we poke our nose into the application error log we find a slew of records. At this point you can 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 you could ride this excuse and not take blame for a bug, it is important to inform them the software can only get fixed if you know something is wrong.

You also need to set up a schedule of when you are available to perform technical support. It is not uncommon for developers to have applications running 24 hours a day, 7 days a week. You all need sleep. You also need to understand the support structure. You might want to get a time applet like Clock Rack (**Figure 4**) that shows what time it is in the different time zones where your applications are running.

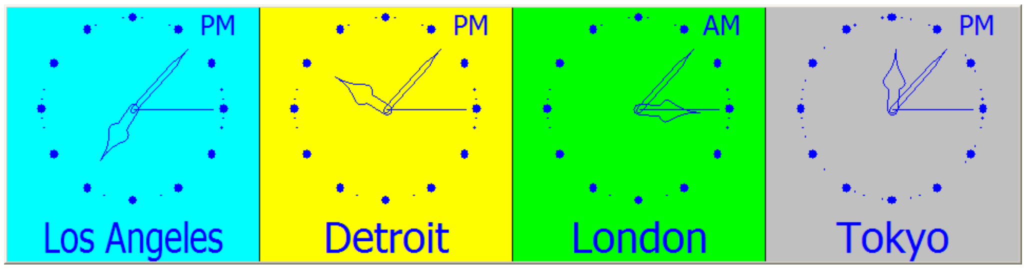


***Figure 4.** You can use a software applet like ClockRack (free to subscribers of PC Magazine's online Utility Library) to show what time it is in different time zones where you have customers running your applications.*

*Clock Rack is available to PC Magazine's online Utility Library subscribers at **http://www.pcmag.com/article2/0,4149,31649,00.asp**.*

Communication mechanisms

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, e-mail, fax machines, and Instant Messaging. Whichever communication mechanism is being used, make sure 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. We have worked for a number of companies both large and small. You might be one of the very fortunate few that never had to be woken up in the middle of the night to handle a support call. It depends on the location and time your customer is using the application. Deployments can be done any time of the day or night, but the reality of the situation is they are done when it is most convenient to the users. Typically this means after hours or during lunch when the application is not used. Sometimes it means working a weekend. This is something you should negotiate long before the setup CD is inserted into the computer and run.

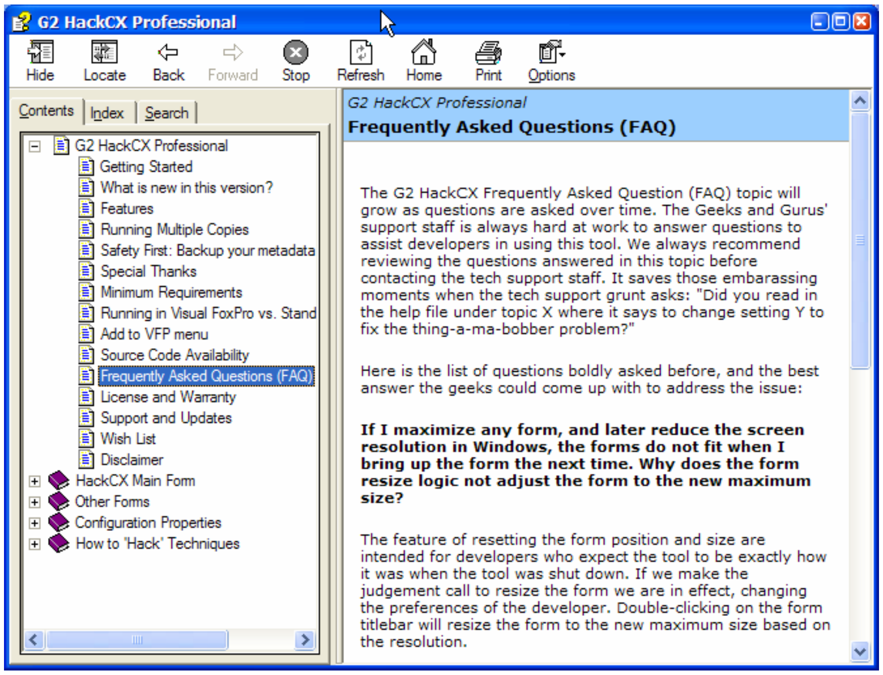
Frequently Asked Questions (FAQ)

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 handled from the customers. This list of questions can be developed as soon as you start collecting specifications, 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 (**Figure 5**), in the readme.txt file, and on the support web site. Larger companies like Microsoft, Symantec, Adobe, Corel, and the like have created a KnowledgeBase of white papers based on issues related to the different applications they support. Who is to say that even the one-person shop cannot leverage these concepts?

Help file

We can already hear the laughter from all corners of the world. Who the heck writes help files for custom software? The reality is many custom projects do not have Help files because they are written directly to the users specifications and 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 might 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 themselves.

However there are advantages to 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 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 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.



G2 HackCX Professional

Frequently Asked Questions (FAQ)

The G2 HackCX Frequently Asked Question (FAQ) topic will grow as questions are asked over time. The Geeks and Gurus' support staff is always hard at work to answer questions to assist developers in using this tool. We always recommend reviewing the questions answered in this topic before contacting the tech support staff. It saves those embarassing moments when the tech support grunt asks: "Did you read in the help file under topic X where it says to change setting Y to fix the thing-a-ma-bobber problem?"

Here is the list of questions boldly asked before, and the best answer the geeks could come up with to address the issue:

If I maximize any form, and later reduce the screen resolution in Windows, the forms do not fit when I bring up the form the next time. Why does the form resize logic not adjust the form to the new maximum size?

The feature of resetting the form position and size are intended for developers who expect the tool to be exactly how it was when the tool was shut down. If we make the judgement call to resize the form we are in effect, changing the preferences of the developer. Double-clicking on the form titlebar will resize the form to the new maximum size based on the resolution.

Figure 5. *A Frequently Asked Question list provides the users with a common list of questions other users have asked technical support to address.*

Training

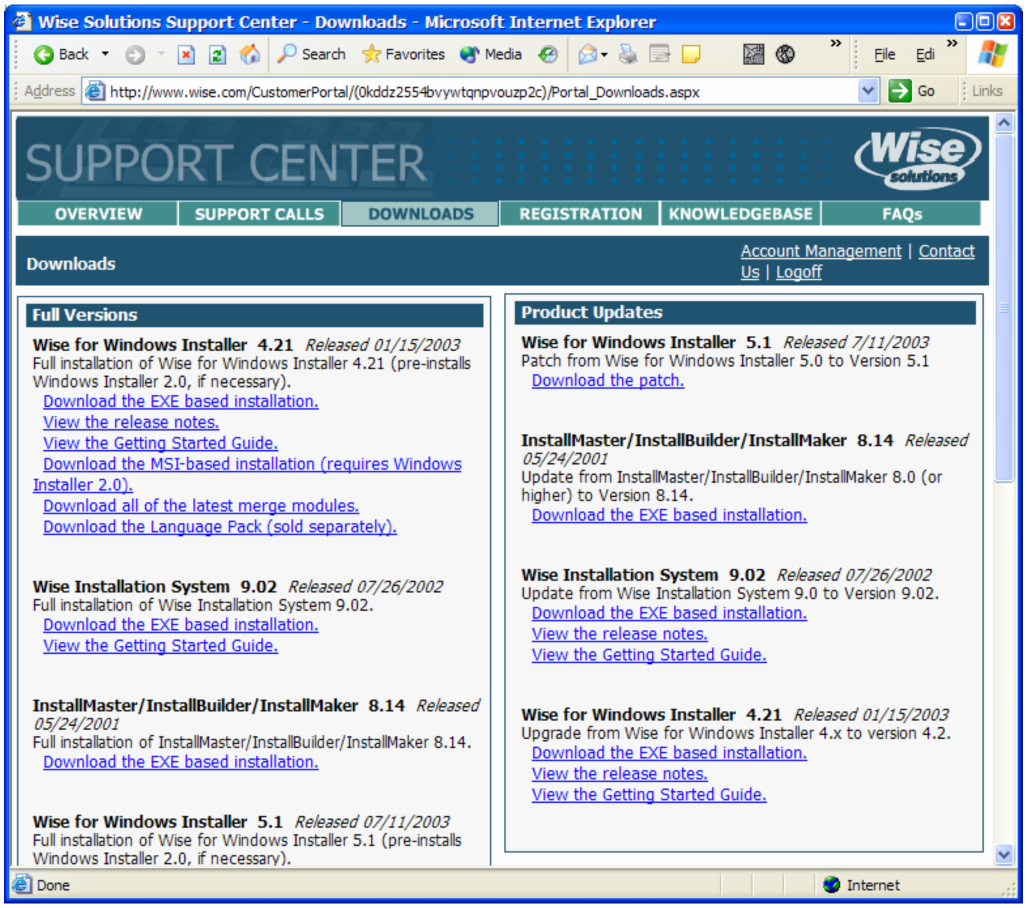
Training can be a one-on-one with the user, or it might be a formal class for all the users, or even a train-the-trainer session. What ever the needs are for the users, training should be handled initially long before the application is moved into production. Ongoing training is sometimes required based 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 us our feelings on the topic of Web sites and the roll they can play for developer shops. We think a Web site 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 the 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 e-mail are all important communication methods to make available.

Users can download installations from your Web site (see **Figure 6**). One of the nice advantages of allowing them to download the package is they decide when they want to take the hit on their bandwidth. If we send it in e-mail, they 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 e-mail order from one of their customers. This is an ideal transfer mechanism for vertical market apps as well because it reduces the distribution costs of duplication and packaging. Web server security also gives you a chance to have clients log in and provide the transfers over a secured socket layer. This also works well for those patches made during the production and post-implementation phases of deployment.


Wise Solutions Support Center - Downloads - Microsoft Internet Explorer
Back
Search
Favorites
Media
File
Edi
Address
http://www.wise.com/CustomerPortal/(0kddz2554bvywtqnpvouzp2c)/Portal_Downloads.aspx
Go
Links
SUPPORT CENTER
Wise solutions
OVERVIEW
SUPPORT CALLS
DOWNLOADS
REGISTRATION
KNOWLEDGEBASE
FAQs
Downloads
Account Management | Contact Us | Logoff
Done
Internet

Full Versions

Wise for Windows Installer 4.21 *Released 01/15/2003*
Full installation of Wise for Windows Installer 4.21 (pre-installs Windows Installer 2.0, if necessary).
Download the EXE based installation.
View the release notes.
View the Getting Started Guide.
Download the MSI-based installation (requires Windows Installer 2.0).
Download all of the latest merge modules.
Download the Language Pack (sold separately).

Wise Installation System 9.02 *Released 07/26/2002*
Full installation of Wise Installation System 9.02.
Download the EXE based installation.
View the Getting Started Guide.

InstallMaster/InstallBuilder/InstallMaker 8.14 *Released 05/24/2001*
Full installation of InstallMaster/InstallBuilder/InstallMaker 8.14.
Download the EXE based installation.

Wise for Windows Installer 5.1 *Released 07/11/2003*
Full installation of Wise for Windows Installer 5.1 (pre-installs Windows Installer 2.0, if necessary).

Product Updates

Wise for Windows Installer 5.1 *Released 7/11/2003*
Patch from Wise for Windows Installer 5.0 to Version 5.1
Download the patch.

InstallMaster/InstallBuilder/InstallMaker 8.14 *Released 05/24/2001*
Update from InstallMaster/InstallBuilder/InstallMaker 8.0 (or higher) to Version 8.14.
Download the EXE based installation.

Wise Installation System 9.02 *Released 07/26/2002*
Update from Wise Installation System 9.0 to Version 9.02.
Download the EXE based installation.
View the release notes.
View the Getting Started Guide.

Wise for Windows Installer 4.21 *Released 01/15/2003*
Upgrade from Wise for Windows Installer 4.x to version 4.2.
Download the EXE based installation.
View the release notes.
View the Getting Started Guide.

Figure 6. *The Wise Solutions support site is one of the better product download sites we have used. The Web site first has a customer login, and then displays the full versions of the products you registered and any update versions you can download.*

Considerations for small shops

Technical support can be tough for smaller shops (1 to 5 developers) when you are out of the office. Our clients sometimes run our applications 24 hours a day, seven days a week. What happens when we need to take a vacation or attend a conference? In the case of Geeks and

Gurus (a four person shop as of the writing of this chapter) we all plan on attending each of the VFP conferences held in the USA and often speak at various user groups throughout North America. How do we support our clients when we are attending technical sessions and 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. We have even supported our clients from a remote campground in the past so almost anything is possible. We also handle our e-mail 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 you can completely relax. At Geeks and Gurus, for example, we get together to verify all went well for the customer, to verify the internal processes set up for software development at our company were optimized, and to discuss ideas on how to leverage the recent success to bring more business our way.

Some developers believe 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. The rest of this chapter offers some suggestions on what you can do after the application is in production. Feel free to pick and choose 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.

Post-mortem review

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

We believe 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) need 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 teammates? 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 we are just checking to see if you are still paying attention)

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, and 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 our experience that developers 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 key to improvement are noted.

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

There is nothing more demoralizing in a company than spending time on something like the post-mortem, and then have nothing done with the results. We experienced this more times than we 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 this is planned, and resenting the "window dressing" that tries to make the company look like it's progressive and "on the ball." It becomes a joke and more damaging than if you do nothing at all.

Customer follow-up

It is always a good idea to get together for a meeting with your customer to do a follow up survey as well. We like to have these meetings at a 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 lunch 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 e-mails 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 satisfied with the application and that it meets or exceeds their expectations in the production environment. There is nothing better than using the application in the real world to flush out the issues 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? We understand, sometimes the answer is yes depending on the customer, but the majority of the time you want to retain your business.

The second is to determine how well you 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 the user finds hard to work with daily. One of our favorite examples of this was a simple grid entry. We missed the *SelectOnEntry* property of a textbox in one column. It just so happened 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 five minute fix saved them hours a week doing data entry. This feedback also improves your development processes at the same time as you learn common problems your customers reveal.

The review with the client also helps 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 our experience the customers need to be trained over time on what a bug is (a true failing of a feature not meeting the requirements) and what an enhancement request is. These reviews with the clients over a period of time help in establishing this difference.

The last reason for a post implementation meeting depends on how well the implementation and review have gone to this point. There should be no surprises at this point because you 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. Most customers we 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. Some customers are shy about reporting what they think are bugs because they feel they have done something wrong to break the program. Others are going to pull your chain every time they find something wrong.

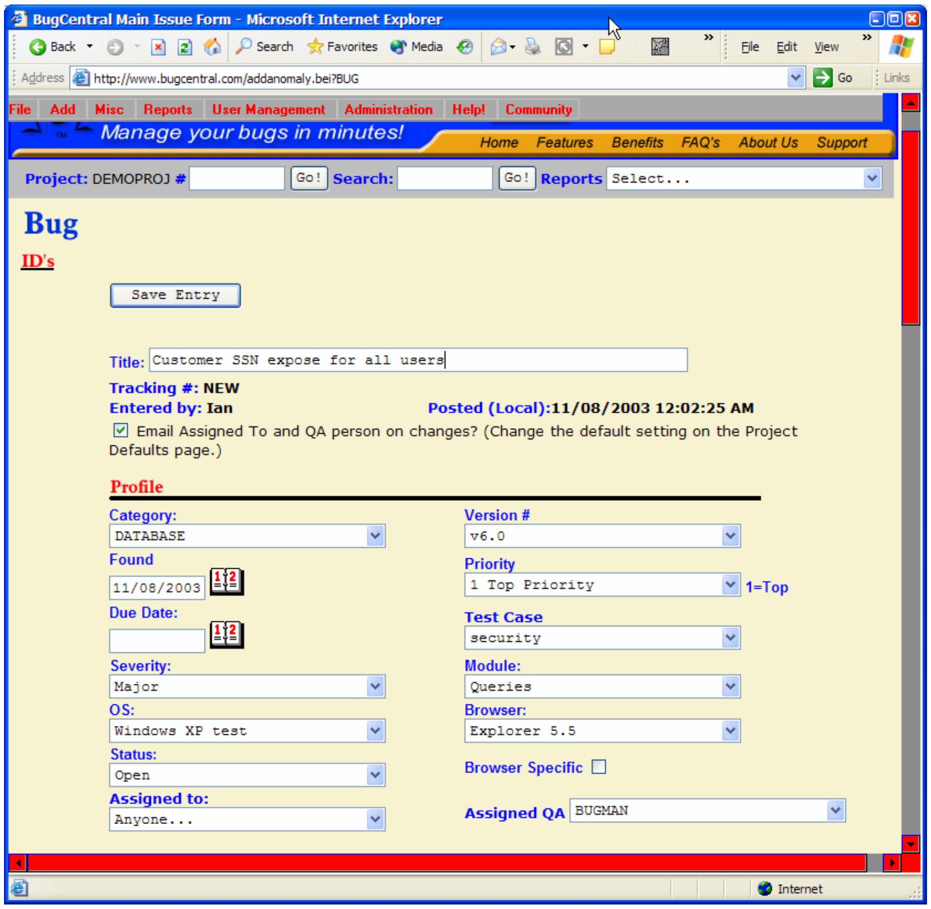
Tracking issues assists you in a number of ways. First, it betters the software you deliver to the customers, which in turn improves customer relations. Second, it provides you a list of things to work on for the various clients. Third, it provides the developers with a training mechanism for better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

Fatal error bugs, the unexpected ones or the ones your customers find ways of producing, are fairly easy to track. Using the new **`TRY…CATCH…FINALLY`**, the object's *Error* method, or a global error handler expose a chance to collect specific application state information. Typical information we collect in our error trapping includes: 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 datasessions, 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 e-mail and have it sent to your support e-mail address. The most recent idea we have is to transform the error information tracked in the error table into XML and have the XML sent via e-mail. 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 the user reports that allegedly does not meet the requirements or when the application fails to operate as they expect. It is important the user specifies a number of key elements. The elements we 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 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 reveals more than a dozen different bug tracking solutions. The BugTrackingSoftware topic on the Fox Wiki links to more than a half dozen available products. Visual Studio 97 (with which VFP 5 was included) includes a solution called Anomaly Tracking System (ATS) written in Visual FoxPro (**Figure 8**). Visual Studio 6 includes a web version called ATSWeb. One nice thing about being database developers is you can augment the ATS products or build a solution that meets your needs if a satisfactory solution cannot be found. Some products like IssueView (**Figure 9**) from a company called IssueView.com has both a desktop component (based on a SQL Server or MS Access database) and an Internet component, which uses Active Server Pages with full, built-in e-mail notification.

There are other Web-based options that can be used including the new Customer Service Center from F1 Technologies (makers of Visual FoxExpress), BugCentral.com (based on VFP and WebConnect, see **Figure 7**), and Steven Black's Wiki (another VFP and WebConnect implementation). The advantage of using a Web-based bug tracking application is your customers can directly enter in reports, geographically diverse development teams can access a common set of reported bugs, and customers can access reports to see the status of reports they made as well as reports from other users.



Project: DEMOPROJ **#** **Search:** **Reports** Select...

Bug

ID's

Title: Customer SSN expose for all users

Tracking #: NEW
Entered by: Ian
Posted (Local): 11/08/2003 12:02:25 AM

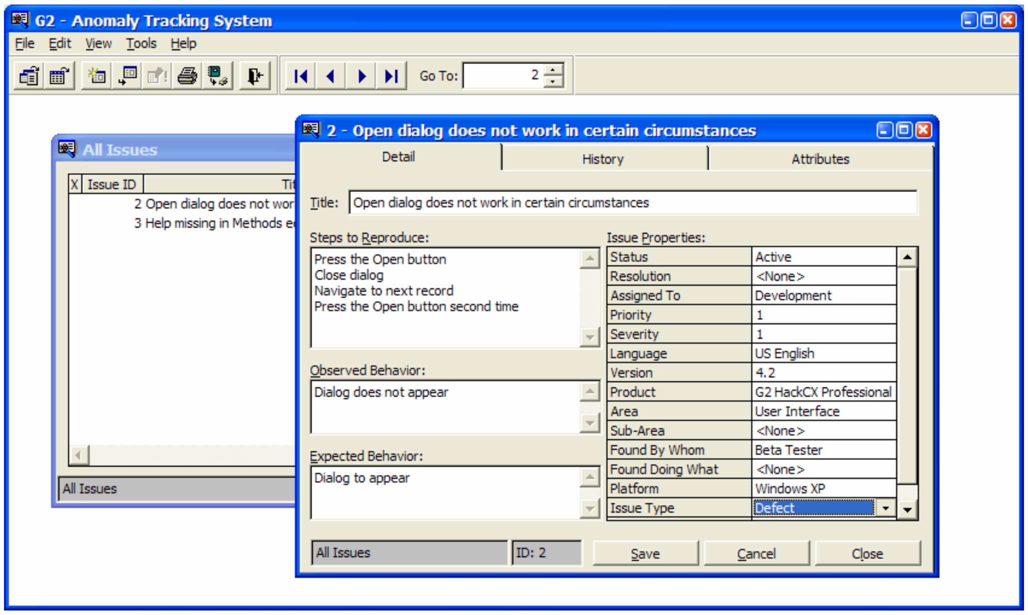
☑ Email Assigned To and QA person on changes? (Change the default setting on the Project Defaults page.)

Profile

Category: DATABASE	Version # v6.0
Found 11/08/2003	Priority 1 Top Priority 1=Top
Due Date:	Test Case security
Severity: Major	Module: Queries
OS: Windows XP test	Browser: Explorer 5.5
Status: Open	Browser Specific ☐
Assigned to: Anyone...	Assigned QA BUGMAN

Figure 7. *BugCentral.com is one example of a Web-based bug tracking application.*

Microsoft products automatically send statistics back to Microsoft via the Internet whenever a GPF or other serious error occurs via the Dr. Watson Error Reporting. Are we the only developers in the world that get some pleasure each time this happens and hoping some Microsoft Web server is getting bombarded with the same reports? There are many ways to get the reports programmatically via e-mail, a Web service, and faxing.



2 - Open dialog does not work in certain circumstances

X	Issue ID	Ti
	2	Open dialog does not wor
	3	Help missing in Methods e

Go To: 2

Title: Open dialog does not work in certain circumstances

Steps to Reproduce:
Press the Open button
Close dialog
Navigate to next record
Press the Open button second time

Observed Behavior:
Dialog does not appear

Expected Behavior:
Dialog to appear

Issue Properties:

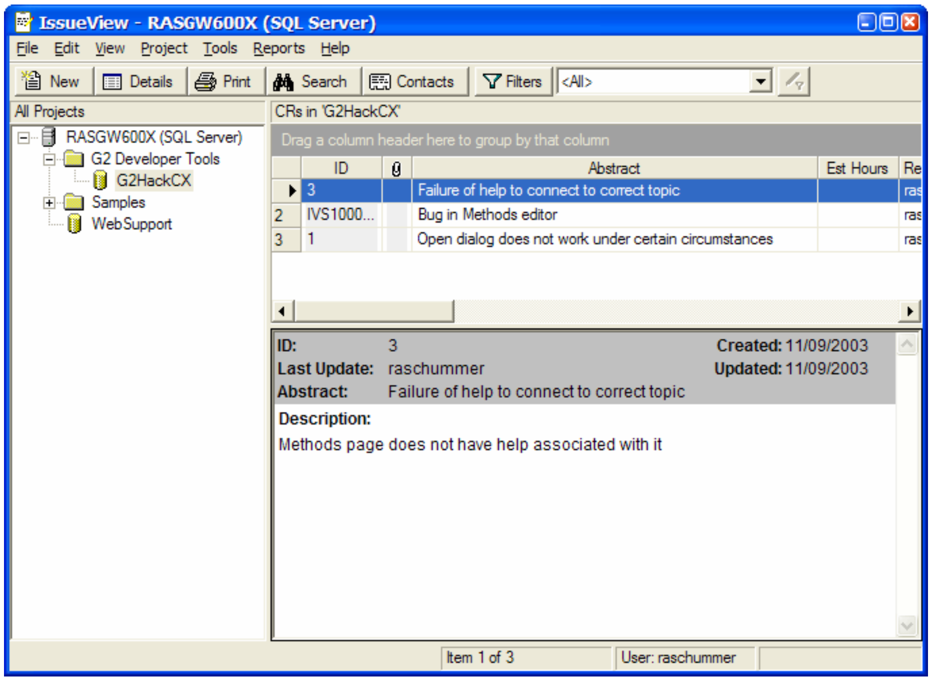
Status	Active
Resolution	<None>
Assigned To	Development
Priority	1
Severity	1
Language	US English
Version	4.2
Product	G2 HackCX Professional
Area	User Interface
Sub-Area	<None>
Found By Whom	Beta Tester
Found Doing What	<None>
Platform	Windows XP
Issue Type	Defect

All Issues | ID: 2

Figure 8. *Some bug tracking packages like Microsoft's older Anomaly Tracking System have just a desktop component.*

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 the priority assigned to the issue. This sets the order the bugs are to be corrected. The reported issues 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.

We have evaluated a number of solutions over the years and always found something in the products lacking. This does not mean you will, but it is uncommon to find developers who are completely satisfied with a canned solution. This usually leads developers to create their own solution.


IssueView - RASGW600X (SQL Server)
File Edit View Project Tools Reports Help
New Details Print Search Contacts Filters <All>
All Projects
RASGW600X (SQL Server)
G2 Developer Tools
G2HackCX
Samples
WebSupport
CRs in 'G2HackCX'
Drag a column header here to group by that column
Item 1 of 3
User: raschummer

	ID		Abstract	Est Hours	Re
▶	3		Failure of help to connect to correct topic		ras
2	IVS1000...		Bug in Methods editor		ras
3	1		Open dialog does not work under certain circumstances		ras

ID: 3 **Created:** 11/09/2003
Last Update: raschummer **Updated:** 11/09/2003
Abstract: Failure of help to connect to correct topic
Description:
Methods page does not have help associated with it

Figure 9. *Several bug tracking packages like IssueView have both a desktop and Internet component.*

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. We have always believed a performance review should never reveal any surprises. It should be 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 contributions a developer made to the project. Also have them provide constructive criticism. You might learn from the feedback something that helps you determine the share of the bonus pool (discussed later), determine internal training opportunities, and see what new techniques have been learned.

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. We find inviting customers to a get together is a great way to build partnerships.

These parties can be held 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 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 you have for them, you in turn need to exceed their expectations in the complete compensation package. One way to do this is to give a bonus to the people who made it all happen by exceeding expectations.

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 one 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 understand you appreciate the sacrifices that the families have made as well.

You can also be creative. 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 the author's partners). Sending the staff (and family) on a weekends away at a nice resort is a great way for people to recharge their batteries before returning to work. One of my favorite ways to reward people is to grant them more vacation time. Top gun salaried developers typically work late into the evenings and on weekends; why not give them back their time without any additional taxes to pay? (Consult your accountant to see if this is okay in your particular situation.)

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. I 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 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 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 we 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 21 cents an hour for my overtime effort over the length of the project. Guess where my loyalties went after this? Do not make this same mistake.

Conclusion

Just as planning is critical to a successful deployment, paying attention to the details during the implementation and after the implementation is important. Hopefully we provided a number of ideas for your consideration in this regard.

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