REPORT ON AN AGILE TECHNICAL WRITING INTERNSHIP AT DOVETAIL SOFTWARE

By Melissa Anne Burpo

This report describes my sixteen-week internship with Dovetail Software in Austin, Texas for the Master of Technical and Scientific Communication program at Miami University. My internship lasted from October 23, 2006 to February 2, 2007. In the report, I examine my time at Dovetail and my role within the organization. It contains an overview of my major projects, which included revising user guides, documenting new product functionality, and developing marketing materials as needed. It goes on to describe in detail my first user guide revision and my experience as the first and only technical writer at a small company. Finally, I analyze my time as a technical writer working with an agile team whose development methods included working in short iterations, working in a team room, and continually delivering small slices of functionality.
REPORT ON AN AGILE TECHNICAL WRITING INTERNSHIP AT DOVETAIL SOFTWARE

An Internship Report

Submitted to the
Faculty of Miami University
in partial fulfillment of
the requirements for the degree of
Master of Technical and Scientific Communication
Department of English

by

Melissa Anne Burpo
Miami University
Oxford, OH
2009

Advisor: _______________________________________
Dr W. Michele Simmons

Reader: _______________________________________
Janel Bloch

Reader: _______________________________________
Dr. Douglas Havelka
# Table of Contents

**CHAPTER 1: INTRODUCTION** .................................................................................................................. 1
  ABOUT DOVETAIL SOFTWARE .................................................................................................................. 1
  MY POSITION WITHIN THE ORGANIZATION ............................................................................................ 3

**CHAPTER 2: INTERNSHIP OVERVIEW** .................................................................................................... 6
  OVERHAULING CURRENT USER GUIDES ................................................................................................. 6
    Dovetail Admin .................................................................................................................................... 7
    Dovetail Agent ...................................................................................................................................... 7
    Dovetail SelfService .............................................................................................................................. 8
    Dovetail ArchiveManager .................................................................................................................... 10
  DOCUMENTING NEW PRODUCT FUNCTIONALITY .................................................................................... 10
  DESIGNING A MARKETING BROCHURE ................................................................................................. 11
  CREATING A PRODUCT DEMONSTRATION .......................................................................................... 12

**CHAPTER 3: DETAILED DESCRIPTION OF A MAJOR ACTIVITY** ........................................................... 14
  LEARNING TO USE DOVETAIL ADMIN ................................................................................................. 15
  PLANNING THE PROJECT ....................................................................................................................... 16
    Defining the audience ......................................................................................................................... 16
    Determining the objectives ................................................................................................................ 17
  REWRITING CONTENT ............................................................................................................................. 17
  WRITING NEW CONTENT ....................................................................................................................... 20
  IMPLEMENTING A NEW DESIGN TEMPLATE ............................................................................................ 20
  PREPARING FOR A PRODUCT RELEASE .............................................................................................. 22

**CHAPTER 4: REFLECTION ON MY ACTIVITIES** ..................................................................................... 24
  DEFINING AGILE ................................................................................................................................. 24
  ANALYZING MY WORK AS AN AGILE TECHNICAL WRITER ............................................................... 27
    Adjusting for Agility ............................................................................................................................ 28
    Discovering the Benefits of Agile Technical Writing ........................................................................... 31
    Tips for New Agile Technical Writers ................................................................................................. 33
  CONCLUSION ......................................................................................................................................... 34

**APPENDICES** ...................................................................................................................................... 36

APPENDIX A: DOVETAIL AGENT INSTALLATION COMPARISON .......................................................... 37
  Dovetail Agent Installation section before the revision ........................................................................ 37
  Dovetail Agent Installation section after the revision .......................................................................... 52

APPENDIX B: DOVETAIL MARKETING BROCHURE .................................................................................... 68

APPENDIX C: DOVETAIL ADMIN ORGANIZATION COMPARISON ...................................................... 74
  Dovetail Admin Topics before Reorganization .................................................................................... 74
  Dovetail Admin Topics after Reorganization ....................................................................................... 76

APPENDIX D: SAMPLES FROM THE NEW PARTS SECTION OF THE DOVETAIL ADMIN USER GUIDE .......... 79

APPENDIX E: MY BLOG POST ABOUT PERSONAS .................................................................................... 87

REFERENCES ........................................................................................................................................... 90
LIST OF TABLES

Table 1: Dovetail’s customer service and support tool ................................................................. 2
Table 2: Dovetail’s administration tools .......................................................................................... 2
Table 3: Dovetail’s developer tools .................................................................................................. 2
Table 4: Before and after comparison of topics in the Dovetail SelfService user guide........ 10
Table 5: Three topics created for an agile user story ................................................................... 30
List of Figures

Figure 1: Dovetail Software organizational chart ................................................................. 4
Figure 2: Weekly internship Gantt Chart .............................................................................. 6
Figure 3: Traditional development cycle compared to agile development cycle
(Swaminathan) ....................................................................................................................... 25
Figure 4: Dovetail’s storyboard ............................................................................................. 27
Figure 5: Sample of a Dovetail retrospective artifact ............................................................ 27
Figure 6: Dovetail’s team room ............................................................................................. 27
CHAPTER 1: INTRODUCTION

In this report, I discuss the sixteen weeks I spent as an intern with Dovetail Software, a small Customer Relationship Management (CRM) software vendor. My internship as a technical writer lasted from October 23, 2006, to February 2, 2007, after which time I was hired as a regular employee. The first chapter of this report describes the company I worked with and how I fit within the organization.

ABOUT DOVETAIL SOFTWARE

Dovetail Software, formerly First Choice Software, began as a Clarify technology partner in 1995. Clarify is a leading CRM software vendor. In the beginning, Dovetail focused on developing small custom enhancements for Clarify users, but eventually, Dovetail expanded its focus to include replacement pieces for the Clarify system. Today, Clarify customers can replace their full Clarify CRM suite with Dovetail products that look and behave much like the Clarify system. Dovetail’s value proposition is that it significantly enhances Clarify functionality and performance, while also significantly reducing implementation and operating costs. Clarify users can replace any portion of their system with Dovetail’s products, and still use their existing database. Some of Dovetail’s more notable customers include Best Buy, ConAgra Foods, Kronos, and Motorola.

The Dovetail CRM product suite includes the main customer service and support tool, administration tools, and developer tools, described in tables 1, 2, and 3. Along with selling software, Dovetail also offers a wide range of customer services, including customer support, consulting, custom development, training, and Clarify support.
Dovetail Agent is a web application, most often used in customer service and support centers to manage customer care issues. It includes components that can manage a variety of issues: customer support needs, parts logistics, field operations, repair tracking, quality control, online customer self-service, and knowledge management.

Table 1: Dovetail's customer service and support tool

<table>
<thead>
<tr>
<th>Product</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dovetail Agent</td>
<td>Dovetail Agent is a web application, most often used in customer service and support centers to manage customer care issues. It includes components that can manage a variety of issues: customer support needs, parts logistics, field operations, repair tracking, quality control, online customer self-service, and knowledge management.</td>
</tr>
</tbody>
</table>

With Dovetail Admin, CRM administrators can manage administration tasks for their Dovetail system from an easy-to-use webpage. Functionality includes managing user accounts, business rules, configuration settings, system parts, data restriction, and other administrative concerns.

RuleManager monitors the CRM database and evaluates when certain business rules should execute. It sends out warnings and notifications based on these rules.

Dovetail Email Agent allows users to perform operations within the Clarify database by sending an email to the database. The email is processed based on a set of configurable rules.

ArchiveManager is a command-line driven application that administrators use to import, export, purge, and archive data from their database.

Table 2: Dovetail’s administration tools

With the Dovetail SDK (Software Development Kit), Clarify developers can write Clarify-based customizations and programs.

Developers can modify their Clarify database schema with Dovetail SchemaEditor, allowing them to easily add application features and customizations.

Dovetail Bolt is a web application that gives developers an easy reference to their schema, web forms, business rules, database info, and Clarify license information. Developers can also use it to execute SQL queries against their database.

Table 3: Dovetail’s developer tools
Dovetail’s development group is unique in that it operates as an agile software team. In the popular blog All About Agile, Kelly Waters (2009, para. 10) defines agile in the following way: “Agile development is a different, more collaborative way of managing software projects, where the team delivers products in small steps, allowing customers to make changes as needed.” For Dovetail, this meant working in short two-week iterations to plan, develop, and test single features to completion; forgoing traditional upfront requirements specifications in favor of simple task level user stories; and working in a single team room to engender communication.

Dovetail, like many organizations, adopted agile practices for several reasons. If applied correctly, it can increase speed-to-market; improve the product’s quality through frequent reviews and revisions; help to identify issues early; make it easier to respond to changes in requirements; and build innovative and motivated teams through increased collaboration (Waters, 2009, para. 1). In Chapter 4, I go into greater detail about Dovetail’s agile methodologies and describe how I fit in as a technical writer.

**My Position within the Organization**

At the time of my internship, Dovetail employed thirteen people split into a sales team and a development team, as shown in Figure 1. Before I was hired as an intern, Dovetail had never employed a technical writer, and the developers had been responsible for the product documentation.

Dovetail was not actively seeking a technical writer when I was looking for an internship. I had met a Dovetail board member as part of an earlier contract job for another company. He knew I was looking for an internship and suggested that I contact Dovetail’s CEO, Stephen Lynn, to see if he might be interested. In an email to Stephen, I described my background and the requirements of the internship. He put me contact with Gary Sherman, the VP of Development, who eventually hired me as Dovetail’s first technical writer.
My job duties were at times somewhat nebulous, because the company did not have any experience with technical writers. Gary and I worked together throughout the internship to define the basic requirements for each of my projects, but he also gave me the freedom to figure out on my own how to execute those requirements.

I had to go outside of the company to find an internship mentor. A former MTSC student introduced me to Mary Connor, the Documentation Architect at Advanced Solutions International. Mary and I exchanged emails often and met numerous times throughout my internship, during which time she offered invaluable writing and professional advice.

I faced many challenges as Dovetail’s first technical writer. I was not able to rely on established guidelines for how I should complete my work. Having never worked in this industry, I was often unsure of which practices I should use for my many projects. I depended on my MTSC training, independent research, and
advice from my boss and my mentor to develop my own set of practices. I documented some of these in an internal wiki for the next technical writer at Dovetail. The wiki post included information about how to use Dovetail’s authoring tool to create and modify documentation content; how to generate the user guides for a release; and the file location and a description of each document I created during my time there.

In the remainder of this report, I describe my experience as a technical writer intern with Dovetail. In Chapter 2, I give an overview of the projects I worked on during my internship. Chapter 3 details one of my major projects at Dovetail. Finally, Chapter 4 describes Dovetail’s agile development team and my experience as an agile technical writer.
CHAPTER 2: INTERNSHIP OVERVIEW

As a technical writer for Dovetail Software, I was responsible for overhauling current user guides, documenting new product functionality, designing a marketing brochure, and creating a product demonstration (see Figure 2). I spent the majority of my time working on current user guides. It was not until the end of my internship that my responsibilities expanded to include the new product documentation and marketing work. In this chapter, I describe each of my projects.

Figure 2: Weekly internship Gantt Chart

OVERHAULING CURRENT USER GUIDES

Before I began the internship, it had been the developers’ responsibility to maintain the product user guides. As their main concern was developing the product, documenting the product was usually an afterthought. Gary explained that the current user guides were difficult to use, because they were unclear, disorganized, and in places inaccurate and out-of-date. With his direction, I spent the bulk of my internship updating, rewriting, reorganizing, and redesigning the product documentation. Gary prioritized my work on the user guides by product
usage and upcoming release dates. I checked in my work daily to TortoiseSVN, Dovetail’s revision control software, so that we could keep a history of the work I completed and have a backup if anything went wrong. During my sixteen weeks at Dovetail, I overhauled three user guides and began work on a fourth.

**Dovetail Admin**

Administrators use Dovetail Admin, formerly fcAdmin, to manage aspects of their Dovetail CRM suite. The Dovetail Admin user guide was my first project. It was slated for a release during my internship, and would include bug fixes and new functionality. Gary requested that I make several changes to the document before the release. I go into greater depth about this work in Chapter 3; however, the following is a high level overview of the work I did to complete this project:

- I learned how to use the application through self-guided learning and a training class.
- I used an audience analysis to create project objectives and a project plan.
- I rewrote content using basic technical writing principles.
- I wrote new content with the assistance of a tester on the development team.
- I single sourced the content.
- I designed and implemented a new template that could be published as HTML and CHM (Microsoft Complied HTML Help).
- I prepared the final document for a product release.

I spent six weeks on the Dovetail Admin project. It was the most difficult because it was my first. I was able to apply much of what I learned on this project to my successive work at Dovetail.

**Dovetail Agent**

Dovetail Agent, formerly fcClient, is an agent-facing web application that organizations use for providing customer service and support, managing logistics, supporting field engineers, and submitting requests for product improvements.
and requirements. The user guide suffered from some of the same maladies as the Dovetail Admin document:

- The document did not have a logical structure, which made it difficult to find information.
- Some of the content was inaccurate; updates to the product often did not make it into the documentation.
- The sections describing how to use the product consisted of long paragraphs, making it difficult to figure out how to perform procedures.

I spent seven weeks completing this project. I rewrote the content to fix these problems and moved the content from its original Word document to the new design template I created for the Dovetail Admin user guide.

The greatest challenge I faced during this project was revising the installation steps to be usable. The original installation section was not clear about what order the steps should be performed or which steps were required and which were optional. In addition, the language used in this section seemed at times overly complicated. I needed to simplify this difficult task as much as possible.

It took many meetings with my subject matter experts before I could finally grasp the difficult technical details that were a part of the Dovetail Agent installation process. Once I understood the process, I tried to revise the section as much as I was able so that it was clear from its organization and language how to successfully install the product. See Appendix A (pg. 37) for a before and after comparison of the installation steps.

**Dovetail SelfService**

Dovetail SelfService, formerly fcSelfService, is an online customer-support application through which users can submit inquiries or issues, manage support cases, monitor their products, and seek sales support. I was tasked with updating the user guide to the new template and revising any content as I saw fit.
The major issue that I identified with the original user guide was its lack of task-oriented guidance. The guide described web pages instead of tasks that could be accomplished with the application. With this in mind, I performed a task analysis to identify the tasks most important to the target audience, who included the application administrator and those people responsible for handling user requests and inquiries. The document is not intended for the customers who the company is supporting with this application. Therefore, topics such as CASE.ASP, which described how those customers can create a case, did not belong in this guide.

Table 4 below is a before and after comparison of topics in the two main sections of the user guide. This comparison demonstrates the transition from descriptive topics to procedural topics and shows the removal of topics I deemed unnecessary during the task analysis.

<table>
<thead>
<tr>
<th>User Guide topics before the revision</th>
<th>User Guide topics after the revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration of fcSelfService Registration Modifications to Contact Information Sales Inquiries Support Cases Product Upgrades Session Administration User Manual DEFAULT.ASP LOGON.ASP REGISTER.ASP WELCOME.ASP CASE.ASP NOTES.ASP ATTACH.ASP CLOSE.ASP FINDSOL.ASP QUERYCASES.ASP</td>
<td>Administering Dovetail SelfService Setting up the Administration page Updating metadata cache files Updating schema cache files Ending a user’s session Checking DLL versions Responding to user requests Registering a new user Modifying a user profile Responding to sales inquiries Responding to support cases Providing product upgrades</td>
</tr>
</tbody>
</table>
Dovetail ArchiveManager

I spent my last week as an intern beginning the ArchiveManager project, a product formerly known as DIET. ArchiveManager imports and exports data from Clarify databases and can be used for both Clarify and Dovetail applications. While I was not able to complete this project before the end of my internship, I was able to identify the major issue that required rework.

The original ArchiveManager user guide contained a great level of detail; however, it was almost purely text on a page without any division of information. For example, directions concerning how to run the application from the command line were embedded in paragraphs throughout the entire document. There was no single reference area that described each command line instruction and what it did. This type of disorganization would make it difficult to understand how to use the application. I did not get much further on this project before my internship ended.

DOCUMENTING NEW PRODUCT FUNCTIONALITY

Shortly after I started my internship, Dovetail began developing a new product, internally called Rev2. Generally speaking, it would be a new case management system that helps organizations manage, track, and resolve customer support issues. It was being developed with agile methodologies, which I describe in more detail in Chapter 4.

No documentation existed yet for the product. My assignment was to document what little functionality existed and to develop a document architecture that
would support context-sensitive help. No one on the team was sure how they wanted this type of help to work.

My first order of business was to research how other applications incorporated context-sensitive help. I found that context sensitivity was accomplished in a couple different ways. Some applications incorporate links within the product that open targeted help topics based on the user’s current context. Others insert tooltips throughout the product and mostly forgo traditional help systems.

I was not sure which would be the best for the new product or how I should organize the new document in preparation for this type of help. Luckily my mentor had some advice to offer. Her first point was that depending solely on tooltips greatly limited the amount of help I could provide the users due to space constraints. In addition, managing tooltips within an application is a much more difficult task than managing a help system.

Her recommendation was that I organize the documentation around the main areas of the application, such as cases, users, and workgroups. Then, the developers would only have to manage links within the application to those main nodes. For example, if a user is on a page within the application that deals with cases, the help link would take him or her to the main Case node within the help. This top level Case topic would include a question, such as “What do you need to do?” Under the question, I would then list all of the possible tasks that a user might want to accomplish with cases. She sold me on the idea.

As new functionality for the product was developed, I began building a structure very similar to the one my mentor described in order to support context-sensitive help. Gary approved of the plan. My internship ended before this project got very far, but I learned a great deal about planning for context-sensitive help.

**DESIGNING A MARKETING BROCHURE**

As the sole professional communicator at Dovetail Software, I was at the top of the list when the CEO decided that he wanted someone to complete the
development of a marketing brochure. Several months before I was hired, Stephen had tasked some consultants with the project, but he was not happy with the final design or content. My assignment was twofold. First, take what the consultants had designed and improve it. Second, collaborate with the sales team to revamp the content.

I spent the first part of the project analyzing marketing brochures developed by other companies in the CRM field. I found that many of the brochures shared common elements, such as featuring customer quotes about the companies and the products, incorporating product screenshots, and presenting the information with a clean and simple design. After I nailed down what should be in the brochure, the sales team and I collaborated by email to improve the brochure content. We went through several iterations of the text until we agreed on the final wording, which Stephen approved.

For the final phase of the project, I put together the findings from my competitive analysis with the approved content to develop the final version of the brochure using Adobe Photoshop and Adobe InDesign. Once it was complete and approved, I uploaded the PDF to our website for public consumption. The brochure is included in Appendix B (pg. 68).

**CREATING A PRODUCT DEMONSTRATION**

The sales team wanted to experiment with new ways of presenting Dovetail’s products. They tasked me with creating a simple and short product demonstration that gave a high-level overview of Dovetail SelfService. The final product would be posted on Dovetail’s website as part of the product description.

I had never made a product demonstration before, but I was able to bend my technical writing skills to meet the needs of the project. I designed a plan of attack which included three phases:
Analyze product demonstrations developed by other companies in the CRM space with the goal of identifying common aspects that I should use in our demonstration.

- Write a storyboard script.
- Develop the demonstration using the approved script.

Through my analysis of other CRM product demonstrations, I determined that my script should fit into the following rough mould:

- A brief introduction describing the high level features of the product.
- A walkthrough of one or two realistic scenarios highlighting common use cases.
- A conclusion recapping the benefits of the product and providing contact information.

I used this basic template to write my first draft of the storyboard script. I worked with the sales team to pinpoint realistic usage scenarios that I should use for the product walkthrough. We settled on two. First, I showed how easy it is for customers to logon to Dovetail SelfService and create a case requesting technical support. Next, I showed how a customer could serve himself by logging on and searching the knowledge base for known issues and solutions. I sandwiched these scenarios between brief marketing pitches mostly developed by the sales team. I used Camtasia to capture the screen and audio recordings, and I used Adobe Photoshop to design the introduction and conclusion screens. The final product was under four minutes long.
CHAPTER 3: DETAILED DESCRIPTION OF A MAJOR ACTIVITY

Of all the work I did during my internship, I spent the most time revising current user guides. For this chapter, I have chosen to describe in detail my first project of this nature: the Dovetail Admin user guide. Chronicling my work for this project not only gives a clear picture of how I spent my time as an intern, but it also illustrates my experience as the first and only technical writer at Dovetail.

Administrators use Dovetail Admin to manage their Dovetail CRM system. From this web-based application, they can administer user accounts, business rules, configuration settings, system parts, data restriction, and other aspects of their CRM system. Changes made within Dovetail Admin are effective immediately in the Dovetail CRM system.

Gary defined five major requirements for the completion of the Dovetail Admin project:

- Revise and reorganize the document for clarity and accuracy.
- Update the document to cover new functionality.
- Rebrand the user guide:
  - Change the company name from First Choice Software to Dovetail Software.
  - Replace the old logo with the new logo.
  - Change the product name from fcAdmin to Dovetail Admin.
- Create a new template design for the user guide that can be delivered as HTML and CHM.
- Complete the final user guide in time for the upcoming product release.

A developer wrote the first fcAdmin User Guide in 2002. In later releases, various developers were responsible for updating and maintaining it as new features were
added. Dovetail did not have any guidelines or templates for developing user
guides. This chapter describes how I learned to use Dovetail Admin, planned the
project, rewrote the user guide content, wrote new content, implemented a new
design template, and prepared for a product release.

**Learning to Use Dovetail Admin**

I did not have any experience using CRM software before I started working with
Dovetail. Because this was my first project, I engaged in several activities to
understand how Dovetail Admin worked and to understand its role within
Dovetail’s CRM suite.

The first task Gary assigned was for me to spend a few days playing around with
Dovetail Admin using an isolated database, so that I wouldn’t break anything in
production. This was a great way for me to get started, because I had to use the
current user guide to figure out how to play around. Along with gaining my first
inklings about the product’s functionality, I was also introduced to many of the
problems in the user guide that I had been assigned to revise. I was able to
immediately begin taking notes about possible improvements for the guide.

During the first week of my internship, Gary also gave a casual training class
about basic CRM concepts to another new hire (a software tester) and me. His
training focused on how Dovetail’s products helped organizations manage the
customer model, the parts model, and service level agreement contracts. He also
described how Dovetail Admin interoperated with Dovetail Agent, their support
agent-facing web application.

Gary served as my main subject matter expert for this project, but I also relied on
other members of the development team. My contact with the developers was
usually casual and could take place at any time. I used either email or an instant
messenger to send off questions, I would grab people for quick conversations in
the hallway or the break room, and if I was really stuck, I would schedule a sit-
down to discuss the issue in more depth. Everyone on the team was available to
provide information as needed.
Learning how to use Dovetail Admin was not an activity with a definite end date. I continued to learn about the product throughout the entire life of this project. However, the initial guidance that Gary provided was essential to get me started.

**PLANNING THE PROJECT**

After a week spent reading the original user guide, using the product, and attending the training class, I felt that I was prepared to plan the project. I used the reader-centered writing process described in Paul Anderson’s book, *Technical Communication: A Reader-Centered Approach* (2003), as guidance. I spent the planning phase finding answers to questions about the document’s audience and objectives.

**Defining the audience**

I needed to find out more about the audience so that I could craft the user guide to best meet their needs. After an early discussion with Gary, I knew that Dovetail Admin users were mostly administrators of some sort. After some further research online, I was able to pin down two common job titles for this role: CRM Administrator and Systems Administrator. With these job titles to go on, I used websites like monster.com and careerbuilder.com to search for job postings, which gave me a general idea of the common tasks and responsibilities required for those positions.

CRM and Systems Administrators are responsible for implementing and managing their organization’s CRM system, among other things. They tend to have an IT background in order to handle the technical requirements of the position, and they must communicate with internal stakeholders to understand the organization’s business needs so that they can configure the CRM tool accordingly. The ultimate goal for their management of the CRM tool is to make the organization’s business operations more effective, and by doing so, increase the organization’s profitability.
Determining the objectives

Aside from the broad requirements that Gary defined for this project, I needed to determine objectives specific to my audience, who had a unique characteristic specific to Dovetail Admin.

Dovetail Admin is a replacement application for the administrative portion of the Clarify system. Much of the audience is already familiar with Clarify’s administrative tool: they have been using it for years and have access to user guides that describe in detail the concepts, workflows, and relationships at work within the application. In order to save time and avoid duplication of effort, Gary and I determined that I should try to avoid providing conceptual information that the readers were already familiar with and had access to. Although I was concerned about new users unfamiliar with Clarify, we decided that it was unrealistic for me to address their needs at that time. My objective was to provide procedural direction to help users complete tasks within Dovetail Admin, a tool that only slightly differed from Clarify.

The reader profile also determined my objectives for the structure of the document. Because Dovetail Admin is so similar to Clarify, most users will jump right in and start using the application before opening the user guide. They will probably only refer to it when they run into a problem. Based on these assumptions, Gary and I determined that the organization of the document should mimic the application’s navigation structure, so that users could easily find the topic describing the window with which they were having a problem.

Rewriting Content

I began rewriting the user guide content after a week spent learning about the application and planning the project. I came up with the following set of guidelines for rewriting the content based on conversations with Gary and my coursework in technical writing:
• Organize the content to match the application’s navigation. See Appendix C (pg. 74) for a before and after comparison of the user guide’s organization.

• Chunk topics into smaller pieces.

• Make the content procedural, using numbered steps.

• Use the active voice and address the reader as “you” rather than “the user.”

• Check the content for accuracy and coverage.

• Use consistent styles throughout the document.

My mentor provided another essential guideline. She recommended that I use a sequential sentence structure in the procedures. In this framework, the orientation of the step is presented before the directive:

• [Location], [action]. For example: From the bottom of the page, select New.

• [Goal], [action]. For example: To view existing states, select List.

• Exception: [Goal], [action] [location]. For example: To view the state details, select the state in the list.

After implementing these revisions, the content was much easier to follow and use. For example, consider the “Manage Countries” topic from the original user guide:

This GUI allows you to create and update countries. It is virtually identical to all of the other fcAdmin GUIs. To add a new country, simply enter data in the fields in the bottom section of the page. Then press the Add button. Make sure that the countries name is unique.

To view existing countries, either press the List link at the top of the page (to view all of them), or enter a filter (a “starts with”) in the text box, and press the List link. The selected countries will be displayed in the top grid. You can click on the Currency, Show States, or Show Time Zones hyperlinks to be taken to the appropriate Currency, States or Counties pages. To update a country, simply select a row in the grid, modify the data in the bottom section, and press Update.
To improve this topic, I split it into four subsections so that it is clear at first glance what you can accomplish from this window. In each subsection, I created numbered steps to accomplish specific goals and used the sequential sentence structure that my mentor recommended. Finally, I addressed the reader as “you” and used the active voice. Following is the “Manage Countries” topic after the revision:

From the Countries window, you can add, view, update, and delete countries that appear in your system.

**Add a country**

1. In the Country Details section, enter the new country data. The country name must be unique. 
   Note: If a country requires an unlisted currency type, define it first. Go to the Currencies section for more information.

2. To save, select Add.

**View a country**

1. To view existing countries, select List. 
   Note: You may optionally specify filter criteria before selecting List.

2. To view the country details, select the country in the list.

**Update a country**

1. Select the country in the list.

2. In the Country Details section, enter the updates. 
   Note: If a country requires an unlisted currency type, define it first. Go to the Currencies section for more information.


**Delete a country**

1. Select the country in the list.

2. Below Country Details, select Delete.

Gary advised that I handle the writing and review process incrementally. After I completed a draft of a section, I submitted it to him for review. He would provide
feedback, which I would then integrate into the document. The goal of this incremental development was to catch problems early so that they were easier to fix. For example, during the first review, Gary noticed that I was referring to the product as DovetailAdmin, all one word. Because we caught it early, I was able to fix the few incorrect instances and move forward in the project using the correct name. This correction mitigated what would have been a larger amount of rework at the end of the project if we had waited for a full draft to review.

**WRITING NEW CONTENT**

The upcoming release included new part numbers functionality which I had to document. Gary paired me with our software tester, who had to test the new functionality, so that we could learn how to use the product together. As we tested out how the new functionality worked, we figured out problems together and asked Gary about anything we could not figure out.

After the tester and I both had a good idea of how it worked, we split up. I wrote a draft of the new section following the same guidelines I used for rewriting the content, and he wrote a test plan draft. Next, we exchanged documents. Using my draft, he tested to see that the procedures I described were accurate. I used his test plan to see if I missed anything he was testing. We gave each other feedback, and then we both revised our work accordingly. See Appendix D (pg. 79) for samples from the new parts section.

**IMPLEMENTING A NEW DESIGN TEMPLATE**

In the past, the user guide had been delivered as a PDF, converted from a Word document. It was mostly text, presented in paragraphs with very few headings. It contained old screenshots that were rarely updated, and most considered it cumbersome to use. My goal was to create a new design template that made it easy for users to find information.

Gary chose West Wind HTML Help Builder as the tool I would use to create and implement the new design. It used a combination of templates and style sheets to
generate HTML and CHM files from a single source. The tool dictated the basic structure of the new HTML and CHM design templates. It generated a basic table of contents on the left of the page with the content body on the right. This portion of the project required that I acquire new technical skills, such as writing raw HTML, using CSS to control style, and learning a new help authoring tool.

I came up the following list of goals and strategies for the new design template, which takes into account the needs of users accessing the help content online. Figure 3 shows the final page design (the table of contents is not included in this graphic).

- Use headings that stand out and clearly describe the content that follows.
- Use Dovetail’s corporate colors throughout the guide (blue and green).
- Use CSS to control the styles within the document, so that design changes are easy to implement.
- Use links and text anchors throughout the page so that readers can jump around to the topics they need.
- Encourage feedback from readers by providing a link to

Figure 3: Screenshot of the final Dovetail Admin user guide design
send comments. The link should open a new email, addressed to Dovetail Support, with the current topic title referenced in the Subject line.

- Inform readers about the last time the topic was updated.

I created the new design template incrementally throughout the life of this project. As I made changes, I submitted the design to Gary for review. I used Dovetail’s source control system to check in all of the changes I made, so that I could easily revert the document to a previous version if needed. I made updates to the design all the way up to the final product release.

**PREPARING FOR A PRODUCT RELEASE**

I finished the first full draft of the user guide on November 28, 2006. Because Gary had incrementally reviewed my work throughout the duration of this project, it was ready for a final review. He had only a few final comments to prepare the guide for the upcoming product release.

I had forgotten to include two general pages that all Dovetail user guides contain: the “Product Support” page and the “About Dovetail Software” page. These were not within the scope of my revision project, so I simply copied them from the original documentation into the guide.

In the final stages of the project, the developers had changed some of the file names referred to in the installation and configuration sections. They tried to remove all references to “fcadmin”. I searched for these instances within the document and replaced those portions of the file names with “dovetail_admin”.

I had listed the “What’s New” pages in chronological order from the oldest release to the newest release. Gary requested that I reverse the order so that the newest information was at the top. This made it easier for users to find the release notes.

Gary requested that I add links from the latest “What’s New” page to the new user guide sections. This simple change made it easier for users to dive into the user guide for information about the new features. I incorporated Gary’s changes, and the document was released with the new version of the software the next week.
The Dovetail Admin project was a great way to begin my internship. It was a crash-course in the real world issues that technical writers commonly encounter: learning how to use new and complex technology over a short period of time; planning and working on large scale projects; and figuring out how to get the job done on time.
CHAPTER 4: REFLECTION ON MY ACTIVITIES

The most striking feature of my internship with Dovetail was the introduction to agile development and my attempt to find a place on the agile team as a technical writer. While I spent the first half of the internship in my own office, working in what could be termed a “traditional” manner, I spent the second half working in the team room as a member of the agile development group. In this chapter, I offer a brief introduction to the agile development methodologies employed at Dovetail, compare the traditional technical writing process with my agile experience, and then offer advice to technical writers new to agile.

DEFINING AGILE

The backbone of agile is the Manifesto for Agile Software Development, which was developed in 2001 by seventeen methodologists (The Agile Alliance) seeking an alternative to the traditional, specification driven development process. The Manifesto states:

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

  **Individuals and interactions** over processes and tools

  **Working software** over comprehensive documentation

  **Customer collaboration** over contract negotiation

  **Responding to change** over following a plan

  That is, while there is value in the items on the right, we value the items on the left more.

Many development methodologies follow the values set forth in the Agile Manifesto, including Extreme Programming, Lean development, Scrum, and Crystal. How these methodologies are applied is best described in opposition to the traditional waterfall method, in which software development follows a sequential, siloed path through the following phases (Royce, 1970):
- Write the requirements specifications.
- Create the design specifications.
- Implement the design specifications (coding).
- Integrate the code fragments into a single build.
- Test that it works according to specifications.
- Deploy and maintain the system.

The Agile Alliance identified many problems with this method. It is difficult to respond to change, to gather feedback from project stakeholders, to identify problems early, and to improve team processes throughout development.

Agile approaches the development cycle from another angle. Instead of horizontal slices of development in which each phase is completed before moving to the next, agile produces vertical slices: each feature is planned, designed, coded, and tested to completion in iterations before moving on to the next feature (see Figure 3).

<table>
<thead>
<tr>
<th>Traditional Development Cycle</th>
<th>Agile Development Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Analysis</td>
</tr>
<tr>
<td>Design</td>
<td>Design</td>
</tr>
<tr>
<td>Code</td>
<td>Code</td>
</tr>
<tr>
<td>Test</td>
<td>Test</td>
</tr>
<tr>
<td>End-to-End small slice of work</td>
<td>End-to-End small slice of work</td>
</tr>
</tbody>
</table>

*Figure 3: Traditional development cycle compared to agile development cycle (Swaminathan, 2009)*

The vertical slices in the agile development cycle represent user stories and replace the thorough requirements specifications used for traditional development projects. User stories usually include three elements: the user role, what that person wants to accomplish, and why he or she needs to accomplish it.
For example, “As an agent, I want to view a case’s history so that I can track its progress.”

User stories are written by project stakeholders, who can include the product owner, business analysts, developers, and anyone else interested in the product’s development – even technical writers. The stakeholders prioritize each user story based on business value and determine when it should be moved into an iteration. Because the stories are developed iteratively, teams can continuously deliver functional pieces of software to be evaluated by the stakeholders.

No agile team is exactly alike. Dovetail borrowed heavily from Extreme Programming, but also used elements from Scrum and Crystal. At the time I joined the team, Dovetail’s agile methodologies included the following attributes:

- Holding a planning meeting before each iteration to select user stories, develop acceptance tests, break stories into tasks, and estimate the effort to complete stories.
- Working in short two-week iterations to develop and test user stories to completion.
- Using a storyboard to track a user story’s progress throughout the iteration. Stories moved through five phases as work was completed: Iteration Backlog, Development, Testing Backlog, Testing, and Accepted. See Figure 4 for a picture of Dovetail’s storyboard.
- Working in a single room to maximize communication between team members. See Figure 5 for a picture of Dovetail’s team room.
- Maintaining a collaborative relationship with project stakeholders in order to quickly respond to their needs.
- Having a “stand-up” every day around the storyboard, during which time team members share what they worked on yesterday, what they plan to do today, and what, if any, obstacles are in their path.
- Working in pairs on development tasks.
- Integrating and testing the product build continually.
- Having retrospectives at the end of each iteration to reflect on team processes and problems. Figure 6 is a sample artifact of a Dovetail retrospective in which the team listed things we were unhappy about (under the sad face) and things we were happy about (under the happy face).

Dovetail’s agile methodologies changed continually in response to the team’s needs.

Figure 4: Dovetail’s storyboard

Figure 5: Dovetail’s team room

Figure 6: Sample of a Dovetail retrospective artifact

ANALYZING MY WORK AS AN AGILE TECHNICAL WRITER

Halfway through my internship, Gary invited me to move into the agile team room. They had recently begun a new development project, internally called Rev2, and he wanted me to begin documenting the application. At the time, there
was very little information available about agile technical writers, so neither of us was sure what to expect from the experiment. We decided to try it out and see how it worked.

**ADJUSTING FOR AGILITY**

When I began working with the team, I wasn’t sure how I was supposed to adjust my work habits to fit with their agile development methodology. However, as time progressed, I figured out ways that helped me become an agile technical writer.

**Documenting Vertical Slices instead of Horizontal Slices**

The first adjustment I had to make was to shift my perception of the process of technical writing from horizontal slices to vertical slices. My previous work on client projects in the MTSC program and the first projects with Dovetail had already trained me to think about my work in horizontal slices, much like the sequential waterfall process:

1. Plan the project.
2. Write a first draft.
3. Submit the draft for review.
4. Revise my work based on feedback.
5. Deliver the completed project.

I quickly realized that a sequential technical writing process would not work for an agile project. Instead of applying each phase to the entire application, I would have to iteratively apply each phase to individual user stories.

For example, one of the first user stories I documented was “As an agent, I want to create a case so that I can record a customer support issue.” While planning the iteration, the team defined the following acceptance criteria for the story:
A case is created and saved in the system.

- The case contains the following information: summary, contact, type, severity, priority, initial contact method, and case note.

The developers used this information to develop the story, the testers used it to test the story, and I used it to document the story.

First, I planned what I was going to write. I didn’t have to go far to gather the initial information I needed. A user story provides the perfect framework to think about documentation:

- It tells me the user: an agent.
- It tells me the task the agent wants to accomplish: creating a case.
- It tells me why the agent wants to accomplish the task: to record a customer support issue.
- It tells me the application fields the agent will use to complete the task: summary, contact, type, severity, priority, initial contact method, and case note.

I referred to the user personas the team had developed to help me think about my reader. “Adam Agent assists customers by answering questions, solving problems, and providing accurate and timely information. He interacts with customers both personally and anonymously by phone, email, instant-messenger, snail-mail, and internet self-service cases.” See Appendix E (pg. 87) for a blog post I wrote about Dovetail’s user personas.

As the iteration progressed and development was completed, I was able to use the application to create a case. This allowed me to stand in the shoes of the user and develop task information. I also spoke to Gary, who was serving as our business analyst, and he described in further detail the purpose of each field. I planned to use this information to write reference material related to the task. Although there was very little functionality to document at that point, I created three topics
for the user story (see Table 5). Later, as new user stories related to creating cases were developed, I was able to add information to the topics.

<table>
<thead>
<tr>
<th>Topic Type</th>
<th>Topic Content</th>
</tr>
</thead>
</table>
| Concept    | What is a Case?  
A case is a record of a customer's problem, issue, or question. |
| Task       | Creating a Case  
1. From the Create Case page, enter one or more search criteria in the Find a Contact section and click Find.  
2. Select the contact from the list.  
3. Enter the case information.  
4. Click Save. |
| Reference   | Case Fields  
The following table describes the default new case fields.  
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Short description of the case.</td>
</tr>
<tr>
<td>Contact</td>
<td>Contact associated with the case.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of customer issue.</td>
</tr>
<tr>
<td>Severity</td>
<td>Severity of the case according to the contact.</td>
</tr>
<tr>
<td>Priority</td>
<td>Priority of the case according to the caseworker.</td>
</tr>
<tr>
<td>Initial Contact Method</td>
<td>Method of the initial contact, for example, by phone or email.</td>
</tr>
<tr>
<td>Note</td>
<td>Detailed notes about the case. When viewing a case, case notes are listed in reverse chronological order from newest to oldest.</td>
</tr>
</tbody>
</table>

Table 5: Three topics created for an agile user story

The biggest difference between writing in vertical slices compared to horizontal slices concerns the last three steps of the process: review, revise, and deliver. On the agile team, I delivered my documentation every day: I published it as HTML daily and made it available internally for anyone in the company to review.
Likewise, I was always available to revise the documentation based on feedback. Just like the agile development process, agile documentation is iterative.

**Working with a Product in Constant Flux**

Another adjustment I had to make was to figure out how to document a product that was in constant flux. Before I joined the agile team, the only software I had ever worked on was already developed and fully functional. I struggled with the shifting nature of the agile project. Every time a developer made a change, he checked in his code to a shared repository, which kicked off a new build of the application. This meant that the application I was documenting could change several times a day as new work was completed. On top of that, this early in the project the focus was on adding basic functionality instead of developing the user interface. The viewable pieces of the application were bare bone. Everything about the look and feel would change.

It almost seemed like a wasted effort to document something that changed so rapidly. I spoke to the team about my concerns and received a clear answer. One goal of agile development is the ability to deliver a functional product at the end of each iteration. Each story must be developed, tested, and successfully built. Now that I was on the team, they explained, we had to add one more requirement for a story to be complete: documented. Every week, I was responsible for documenting the user stories in the iteration. Even if I expected a certain feature to be updated in the future, I could only document what was accepted into the iteration. Anything more than that went against the nature of an agile project, because that future story might change or be dropped from the project altogether.

**DISCOVERING THE BENEFITS OF AGILE TECHNICAL WRITING**

I had reservations about joining the agile team. I thought the team room might be too distracting, and I wasn’t sure how I would fit in with the developers and testers. However, it didn’t take long for me to discover the benefits of working as an agile technical writer.
A major advantage of working in an agile team room was the proximity to my subject matter experts. Everyone was within sight distance. When I had questions or needed help working something out, they were right there in the same room. I could see when they were busy and when they might have a minute for me. This proximity dramatically changed how I went about my documentation work. It was no longer a solitary task. Instead, I engaged in frequent dialogue with the developers and testers, who provided a constant stream of input that informed my writing tasks.

The developers also became more aware of the work I was doing, because every morning in stand-up we each reported on what we had been working on, what we planned to work on, and what obstacles were in our path. Because everyone was aware of my status, I was more likely to find the help I needed. For example, one morning I reported a major obstacle. I was completely stuck on the ArchiveManager project. The application is run from the command line, an area of computing that completely terrified me. I had no idea how to even begin using the product, which was a problem, because I had to revise its documentation. One of the developers perked up and explained that he had written the entire application, as well as the documentation, and that he would be happy to walk me through the product. Without the stand-up, I might have held back from sharing this obstacle for another few days as I toiled away in my own room. The structure of that daily practice gave me a vehicle to request support.

The proximity also benefited the developers. One day a developer was writing up all of the bug fixes and product enhancements for an upcoming release. This was for a product I had not worked on yet, so he had initially assumed that he had to write it up himself. He realized, however, that I was right there. He called me over to his desk, and we worked on the release notes together. He offered the technical information, and I provided my writing expertise, which made the task easier to complete for both of us.

While I still had a lot to learn about being an agile technical writer, I was able to pick up many new skills by the end of my internship. I learned how to adjust my
work habits to fit an agile development cycle, how to work with applications still under development, and how to collaboratively interact with developers and testers. I was surprised to find how well technical writing fits with agile development.

**TIPS FOR NEW AGILE TECHNICAL WRITERS**

The material available for technical writers working on an agile team is slowly growing. Do a Google search for “agile technical writer” and you’ll find a slew of blogs and articles dedicated to that very subject. When I first started at Dovetail in October 2006, the bucket was much smaller, and I wish that I had had more resources to get started in my new role. What I offer here is by no means a definitive list of best practices. However, the following tips should be useful for new agile technical writers.

- Work in the team room. It can be distracting at first, but the conversations in the room are invaluable. On an agile team, conversations take the place of traditional requirements specifications. You’ll get behind if you’re not there.

- Write iteratively. For each new story, gather information, write a draft, publish it, gather feedback, and then revise. Keep that cycle up until the documentation has been approved and the story is marked complete.

- Internally publish the documentation regularly. The sooner you get it out there, the sooner it can be reviewed and revised. Some agile technical writers use a wiki to internally draft the documentation, so that the team can edit the content directly.

- Keep pace with the stories in development. If a developer adds new code, changes code, or fixes a problem, then update the documentation to reflect that change as soon as you can. You want the team to review your changes while the work is still fresh in their mind, and a story can’t be marked complete until the documentation is updated.

- Speak up! Technical writers are a great asset to an agile team. We are natural user advocates and can raise awareness about user needs early in the
development process. If you see a problem with the user interface, an error message, the navigation structure, or anything else that might be a problem for the user, then you should share it with the team right away.

**Conclusion**

My time at Dovetail provided an excellent introduction to the field of technical communication. I came away from the experience knowing that I had made the right choice when I selected the MTSC program. The position gave me the opportunity to sit at the intersection of writing, editing, design, and usability, and I ended the internship excited about all of the touchpoints of this field.

Without my MTSC background, I would have been lost as the first and only technical writer for the company. Almost every project in the MTSC program was conducted for a real client, which prepared me for my role at Dovetail. For example, I collaboratively put together a website for a waste water group, wrote and designed a procedural reference card for nurses at a local hospital, and wrote a white paper about a local environmental issue for the university. Each project taught me how to practically apply the technical writing theory taught in class to real-world problem solving contexts.

The MTSC program also required that I take supplementary coursework in a field of my choice. My management information systems (MIS) and computer programming classes were the perfect framework for my internship. The MIS course gave me grounding in the business needs that drove Dovetail’s products, and the programming courses introduced me to development processes and concepts employed by Dovetail’s developers. This background made it easier for me to get up and running as a technical writer in the computer software industry.

My internship at Dovetail was an experiment. They had never employed a technical writer before, and they were not sure if they wanted one permanently. Hiring an intern gave them an opportunity to test out the position on a trial basis. Before I began my internship, Dovetail’s product documentation was a mere afterthought. The existing documentation was difficult to use and there had not
been a plan in place for documenting the new product in development. I was able to change Dovetail’s view of documentation during my time there. Along with improving four existing user guides, I also began a user guide for the new product. Furthermore, the development team expanded its definition of a complete user story to include documentation.

The experiment worked. Dovetail offered me a fulltime position after my internship, and I accepted. I worked with them for another year, until I had the opportunity to join my internship mentor and a former MTSC student at Advanced Solutions International where we are currently making the transition to agile.
APPENDIX A: DOVETAIL AGENT INSTALLATION COMPARISON

DOVETAIL AGENT INSTALLATION SECTION BEFORE THE REVISION

Installation Guide

This section provides detailed requirements for fcClient, what files are included in this product, installation and other implementation considerations.

Requirements

This version of fcClient requires the following:

**First Choice Products:**
- fcSDK (First Choice Software Development Kit for Clarify)
- First Choice fcAdmin web application

**Operating System:**
- Microsoft® Windows® XP, 2000, or NT® 4.0

**Clarify Version:**
- 7.0 or later

**Clarify Tools:**
- Data Dictionary Editor (DD Editor)
- Data Exchange (Dataex)
- User Interface Editor (UI Editor)

**Supported Browsers:**
- Microsoft® Internet Explorer® Version 5.5 or later

**Other Tools:**
- Microsoft® ActiveX Data Objects® (ADO) Version 2.6 or later
- Microsoft® Internet Information Services® (IIS) Version 5.0 or later
- SQL query tool (capable of executing SQL statements)
- Unzip utility

**EEO Requirements:**
- Microsoft Outlook Client, CDO (Collaborative Data Objects)
  These must be installed on each client machine that wishes to use EEO.
  AspSmartUpload Component (provided with fcClient)

**Attachments:**
- If you wish to allow users to upload attachments, you must install and configure the AspSmartUpload Component (provided with fcClient)

**FTS Requirements:**
- Clarify FTS Client

**DDE Requirements:**
- Microsoft .NET Framework 1.1 SP1
  If you wish to use fcClient as a DDE server or a DDE client, the client machine must have the .NET Framework installed.

Packaging

fcClient is shipped to you as a zip file.

Installation Tree

The following files are provided with this product:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs/fcClient_user.pdf</td>
<td>This document</td>
</tr>
<tr>
<td>files/fcClient_alias_tables.dat</td>
<td>Import file that includes default tables for Classic Flashes</td>
</tr>
<tr>
<td>files/fcClient_basic_data.dat</td>
<td>Import file that includes permissions that are shared by fcClient products</td>
</tr>
<tr>
<td>files/fcClient_config.dat</td>
<td>Import file that contains configuration items needed for fcClient</td>
</tr>
<tr>
<td>files/fcClient_quality_data.dat</td>
<td>Import file that includes permissions that are specific to fcQuality</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>files\eclient_support_data.dat</td>
<td>Import file that includes permissions that are specific to fcSupport</td>
</tr>
<tr>
<td>files\eclient_identity.dat</td>
<td>Import file that includes permissions that are specific to fcCallCenter</td>
</tr>
<tr>
<td>files\eclient_colors.data</td>
<td>Import file that contains default colors for fcClient</td>
</tr>
<tr>
<td>files\eclient_logistics.data</td>
<td>Import file that includes permissions that are specific to fcLogistics</td>
</tr>
<tr>
<td>files\eclient_web_forms.data</td>
<td>Import file that includes base web form names for Resource Configurations</td>
</tr>
<tr>
<td>files\eclient_sales.data</td>
<td>Import file that includes permissions that are specific to fcSales</td>
</tr>
<tr>
<td>files\eclient_mn_schemes.data</td>
<td>Import file that includes numbering schemes needed by fcClient</td>
</tr>
<tr>
<td>pages\fc_env</td>
<td>File that contains logon information</td>
</tr>
<tr>
<td>pager\global.aspx</td>
<td>Initialization and termination procedures for the application</td>
</tr>
<tr>
<td>pager\defautl.asp</td>
<td>Default Active Server Page for fcClient</td>
</tr>
<tr>
<td>pager\account*.asp</td>
<td>Active Server Pages for displaying and updating accounts</td>
</tr>
<tr>
<td>pager\account\tablesort.txt</td>
<td>Behavior file (htc) for table sorting</td>
</tr>
<tr>
<td>pager\action*.asp</td>
<td>Active Server Pages for displaying and updating action items</td>
</tr>
<tr>
<td>pager\address*.asp</td>
<td>Active Server Pages for displaying and updating customer addresses</td>
</tr>
<tr>
<td>pager\admin*.asp</td>
<td>Active Server Pages for Admin functionality</td>
</tr>
<tr>
<td>pager\bin*.asp</td>
<td>Active Server Pages for product bins</td>
</tr>
<tr>
<td>pager\calendar*.as</td>
<td>Code modules for calendar control</td>
</tr>
<tr>
<td>pager\case*.asp</td>
<td>Active Server Pages for displaying and updating cases</td>
</tr>
<tr>
<td>pager\cep*.asp</td>
<td>Active Server Pages for fcQuery</td>
</tr>
<tr>
<td>pager\code*.js</td>
<td>Javascript files used in building/checking ASP pages and menus</td>
</tr>
<tr>
<td>pager\color*.asp</td>
<td>Active Server pages to modify color choices on the profile form</td>
</tr>
<tr>
<td>pager\commit*.asp</td>
<td>Active Server Pages for displaying and updating commitments</td>
</tr>
<tr>
<td>pager\config*.asp</td>
<td>Active Server Pages for site part configuration management</td>
</tr>
<tr>
<td>pager\console*.html</td>
<td>Pages for console and profile functionality</td>
</tr>
<tr>
<td>pager\contact*.asp</td>
<td>Active Server Pages for displaying and updating contacts</td>
</tr>
<tr>
<td>pager\contract*.asp</td>
<td>Active Server Pages for displaying and updating contracts</td>
</tr>
<tr>
<td>pager\ctrl*.as</td>
<td>Active Server Pages for displaying and updating change requests</td>
</tr>
<tr>
<td>pager\depot*.as</td>
<td>Active Server Pages for displaying and updating Depot Repair parts</td>
</tr>
<tr>
<td>pager\dll*.dll</td>
<td>Client-side DLLs used by fcClient</td>
</tr>
<tr>
<td>pager\eo*.asp</td>
<td>Active Server Pages for EEO (Enhanced Email Out)</td>
</tr>
<tr>
<td>pager\eo*.js</td>
<td>JavaScript code for EEO (Enhanced Email Out)</td>
</tr>
<tr>
<td>pager\eo*.vba</td>
<td>VBScript code for EEO (Enhanced Email Out)</td>
</tr>
<tr>
<td>pager\flasp*.as</td>
<td>Active Server Page for displaying flashes</td>
</tr>
<tr>
<td>pager\fts*.as</td>
<td>Active Server Page for using FTS (Full Text Search)</td>
</tr>
<tr>
<td>pager\generic*.as</td>
<td>Active Server Pages for generic select functionality</td>
</tr>
<tr>
<td>pager\include*.asp</td>
<td>Include files for the Active Server Pages</td>
</tr>
<tr>
<td>pager\include\adodb.inc</td>
<td>ADO constants for JavaScript</td>
</tr>
<tr>
<td>pager\include\inc\vbscript.inc</td>
<td>Common VB Script functions</td>
</tr>
<tr>
<td>pager\include\localid.inc</td>
<td>Sets the Locale Identifier on the ASP Session</td>
</tr>
<tr>
<td>pager\include\blank.htm</td>
<td>Default blank page used for grid frames</td>
</tr>
<tr>
<td>pager\include\loading.htm</td>
<td>Page used when certain other pages are still loading</td>
</tr>
<tr>
<td>pager\interact*.as</td>
<td>Active Server Pages for displaying and updating interactions</td>
</tr>
<tr>
<td>pager\log*.asp</td>
<td>Active Server Pages for displaying and updating email, note, phone and research logs</td>
</tr>
<tr>
<td>pager\login*.as</td>
<td>Active Server Pages for logging in and out of fcClient</td>
</tr>
<tr>
<td>pager\notifier*.asp</td>
<td>Active Server Pages for the fcClient Notifier</td>
</tr>
<tr>
<td>pager\part\num*.as</td>
<td>Active Server Pages for displaying and updating part numbers</td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pages/part_request*.asp</td>
<td>Active Server Pages for displaying and updating part requests</td>
</tr>
<tr>
<td>pages/query*.asp</td>
<td>Active Server Pages for querying cases, subcases and change requests</td>
</tr>
<tr>
<td>pages/sample/sample_contact.asp</td>
<td>Sample Active Server Page for incorporating Attachments Anywhere functionality on the contact page</td>
</tr>
<tr>
<td>pages/sample/sample_site_part.asp</td>
<td>Sample Active Server Page for incorporating Flashes Anywhere functionality on the site part page</td>
</tr>
<tr>
<td>pages/sample/sample_site.asp</td>
<td>Sample Active Server Page for incorporating Flashes Anywhere functionality on the site page</td>
</tr>
<tr>
<td>pages/sample/dde_client.asp</td>
<td>Sample DDE client page, demonstrating how to allow fcClient to function as a DDE client to communicate with other Windows applications that act as a DDE server.</td>
</tr>
<tr>
<td>pages/select*.asp</td>
<td>Active Server Pages for selecting addresses, contacts, contracts, part numbers, products, sites, site parts, interactions, solutions, and part requests</td>
</tr>
<tr>
<td>pages/site*.asp</td>
<td>Active Server Pages for displaying and updating sites</td>
</tr>
<tr>
<td>pages/site_part*.asp</td>
<td>Active Server Pages for displaying and updating site parts</td>
</tr>
<tr>
<td>pages/solution*.asp</td>
<td>Active Server Pages for displaying and updating solutions</td>
</tr>
<tr>
<td>pages/stylesheets*.css</td>
<td>Stylesheet for the Active Server Pages</td>
</tr>
<tr>
<td>pages/subcase*.asp</td>
<td>Active Server Pages for displaying and updating subcases</td>
</tr>
<tr>
<td>pages/territory*.asp</td>
<td>Active Server Pages for displaying and updating territories</td>
</tr>
<tr>
<td>pages/user*.asp</td>
<td>Active Server Pages for displaying and updating users</td>
</tr>
<tr>
<td>pages/workflow*.asp</td>
<td>Active Server Pages for manipulating objects (e.g., activity log, dispatch, change status, etc.)</td>
</tr>
<tr>
<td>pages/zipcode*.asp</td>
<td>Active Server Pages for handling zipcodes</td>
</tr>
<tr>
<td>res/sample/res_schema.sch</td>
<td>Changes to make to the database schema to use baseline data restriction with fcClient</td>
</tr>
<tr>
<td>res/sample/res_data.dat</td>
<td>Data restriction rules to import to use baseline data restriction with fcClient</td>
</tr>
<tr>
<td>schema/fcclient_schema.sch</td>
<td>Schema modifications required for fcClient for use with DD Editor</td>
</tr>
<tr>
<td>schema/fcclient_schemascript.xml</td>
<td>Schema modifications required for fcClient for use with DovetailSchemaEditor</td>
</tr>
<tr>
<td>schema/fcclient_clarify_v8_and_up_schema.sch</td>
<td>Schema modifications required for fcClient for use with DovetailSchemaEditor on Clarify versions 8 and up</td>
</tr>
<tr>
<td>scripts/*.sh</td>
<td>Files used to update the fcClient cache from a script and upgrade classic flashes to Flashes Anywhere</td>
</tr>
<tr>
<td>sql/mysql/*.sql</td>
<td>Triggers and Stored Procedures for Microsoft SQL Server databases</td>
</tr>
<tr>
<td>sql/oracle/*.sql</td>
<td>Trigger and Stored Procedures for Oracle databases</td>
</tr>
<tr>
<td>sql/sybase/*.sql</td>
<td>Triggers and Stored Procedures for Sybase databases</td>
</tr>
<tr>
<td>aspSmartUpload*.dll</td>
<td>aspSmartUpload components</td>
</tr>
<tr>
<td>aspSmartUpload/legal.htm</td>
<td>aspSmartUpload Legal information</td>
</tr>
<tr>
<td>zipcode/zipcode.js</td>
<td>Interface to load zipcode data into the database</td>
</tr>
<tr>
<td>zipfile/zip.dat</td>
<td>Sample mapping files for zipcode interface</td>
</tr>
</tbody>
</table>

**Manual Installation**

Note: It is highly recommended that you first install fcClient on a test system and get familiar with it before installing it on a production system.
The fcClient files should be installed on each server that will act as a web server for fcClient. It is recommended that you unpack the zip file containing fcClient into an fchoice subdirectory. For this document, all examples will use "c:/fchoice/fcclient" as the install folder.

Once you have unzipped the installation package, the following steps must be performed:

- Install the fcSDK
- Update the Clarify schema
- Add the database trigger
- Add the fc_wip_q_counts stored procedure
- Import Data Files
- Import Configuration Items
- Set up necessary web privclasses
- Install/configure a web server
- Edit the .envfile
- Modify the global.asa file
- Install aspSmartUpload
- Setup the FTS Client software (optional)
- Setup Flashes Anywhere (optional)
- Setup Attachments Anywhere (optional)
- Setup Zipcodes (optional)

Install the fcSDK
Install the First Choice Software Development Kit for Clarify (fcSDK). The fcSDK includes FCFL.NET and the API toolkits. Follow the installation instructions included with the fcSDK. Be sure to use the License Installer to install your license key(s), and also be sure to run the Data Verifier to ensure that there are no inconsistencies or bad data that might prevent FCFL.NET from initializing properly. The Data Verifier and License Installer are shipped with the fcSDK.

Updating the Clarify Database Schema
This section details the steps required to update the Clarify database schema to work with fcClient.

Backup the Database
Prior to making schema changes, it is always advisable to back up your current database using standard backup procedures.

Importing the New Schema

The following steps should be followed to update the Clarify database schema.

To perform these changes using Dovetail SchemaEditor, follow these steps:
1. Edit the .SchemaEditor file
2. Set the database connection information
3. Set the inputFilePath to fcclient_schemascript.xsl
4. Preview the changes (SchemaEditor.exe -p)
5. Apply the changes (SchemaEditor.exe -a)
6. If you are using Clarify version 8 or up, repeat the above steps, but set the inputFilePath to 
   &Client_clarify_v8_and_up_schemascript.xml

To perform these changes using Clarify's DD Editor, follow these steps:
Note: These steps differ on recent versions of Clarify, as the Clarify GUI has changed, but the steps still
provide a guideline that you should find useful.

1. Start the Data Dictionary Editor
2. Choose Save To File (or Save) from the File menu
3. Type my_fec.sch in the text box and press the OK button
4. With a text editor, edit the my_fec.sch file, and the supplied feclient_schemasch file. Make all
   changes listed in the feclient_schemasch file (in the my_fec.sch file), and save the changes
5. In DD Editor, Choose Apply Changes from the Actions menu
6. Select the my_fec.sch file, press Open
7. When presented with the question that this operation might be time consuming, press Continue
   (or Proceed)
8. Review the results presented. If any errors are shown, fix them and repeat steps 5-7
9. If there are no errors, press the Apply Changes button
10. On the next form, press the Upgrade button
11. When the upgrade is complete, press the Done button
12. Close DD Editor

Applying Data Restrictions

If you are planning on using Data Restriction with &Client, you must also apply the schema for it. If you
plan to use the sample that is shipped with this product, please repeat the steps above with the
res/sample_res_schemasch file.

Add the Database Trigger

This section details the steps required to add a database trigger necessary for proper operation of the
&Client Notifier.

Adding the Database Trigger for Oracle databases

You should compile the provided trigger (ora_trig.sql). The following is one way of performing the compilation:

```sql
sqlplus sa/<pass>@<sid> @ora_trig.sql
```

where:

- `<pass>` is sa's password
- `<sid>` is the name of the database SID

You should see a message that the trigger was created. If there are any error messages, you
must fix whatever is wrong and recompile.
Adding the Database Trigger for Microsoft SQL Server and Sybase databases
You should compile the provided trigger (mssql_trig.sql or syb_trig.sql). The following is one way of performing the compilation:

```
isql -U<pass> -P<pass> -S<server> -d<database> < <sql_file>
```

where:
- `<pass>` is sa’s password
- `<server>` is the name of the database server
- `<database>` is the name of the database
- `<sql_file>` is the name of the SQL file

If there are any error messages, you must fix whatever is wrong and recompile.

Add the fc_wip_q_counts Stored Procedure
This section details the steps required to add a stored procedure used to count the objects in wipbins and queues.

Adding the Stored Procedure for Oracle databases
You should compile the provided stored procedure (ora_fc_wip_q_counts.sql). The following is one way of performing the compilation:

```
sqlplus sa/<pass> &<sid> &ora_fc_wip_q_counts.sql
```

where:
- `<pass>` is sa’s password
- `<sid>` is the name of the database SID

You should see a message that the stored procedure was created. If there are any error messages, you must fix whatever is wrong and recompile.

Adding the Stored Procedure for Microsoft SQL Server and Sybase databases
You should compile the appropriate stored procedure (mssql_fc_wip_q_counts.sql or syb_fc_wip_q_counts.sql). The following is one way of performing the compilation:

```
isql -U<pass> -P<pass> -S<server> -d<database> < <sql_file>
```

where:
- `<pass>` is sa’s password
- `<server>` is the name of the database server
- `<database>` is the name of the database
- `<sql_file>` is the name of the stored procedure file

If there are any error messages, you must fix whatever is wrong and recompile.
Import the Data Files
There are a few data files provided in the files directory that you should import. They contain the web privileges, configuration and list information that you will need for fcClient. The files are:

- fcclient_basic_data.dat
- fcclient_colors.dat
- fcclient_config.dat
- fcclient_quality_data.dat
- fcclient_support_data.dat
- fcclient_callcenter_data.dat
- fcclient_logistics_data.dat
- fcclient_web_forms.dat
- fcclient_sales_data.dat
- fcclient_num_scheme.dat (only for Clarify versions 7 and below)

In addition, if you are using the presupplied Data Restriction provided with the fcClient you must also import one data file (located in the res subdirectory):

- sample_res_data.dat

If you are adding Flashes Anywhere functionality and converting from Classic Flashes, import this data file to set up the basic alert table objects in the database:

- fcclient_alert_tables.dat

If you wish to import a file, you should first edit it, if necessary. (See next section for updating the fcclient_config.dat file.) Typically you will not need to edit any of these data files. If you do edit one, execute the following command after saving any changes:

```
<path>dataex
    -user_name <user>
    -password <pass>
    -db_server <serv>
    -db_name <db>
    -imp <file>
```

Where:
- <path> is the path to the dataex program
- <user> is the system administrator user
- <pass> is the system administrator password
- <serv> is the database server name
- <db> is the database name
- <file> is the name of the file to import

Note: If the <file> is not in the current directory, you must specify the path to the directory it is in.

Note: Each file should import with 0 errors and 0 warnings. If there are any errors or warnings, the dataex mes file should be investigated for the reason.
## Configuration Items

There are several configuration options that can be set for **fcClient**. These configuration items are shipped in the `fcclient_config.dat` file and may be modified with your favorite text editor. Once the items have been modified, you will import the configuration file into the Clarify database using the Clarify Data Exchange Tool (dataex). See the above section on importing data files for more instructions on using dataex.

<table>
<thead>
<tr>
<th>Item Name</th>
<th>Description and Valid Values</th>
</tr>
</thead>
</table>
| `default_fc_notifier_frequency` | **DEFAULT: i_value = 60**  
  Default fNotifier Frequency - How often to check for new messages (number of seconds) |
| `open_item_from_notifier`  | **DEFAULT: i_value = 0**  
  Should the fNotifier try to parse the message and allow the item to be opened? 0 = no; 1 = yes  
  By setting this config item to 1, the fNotifier will try to parse the message and will display a hyperlink to open the item.  
  The success of this parsing is highly dependent on the notification message.  
  It can successfully parse messages such as:  
  - The Case 9 was yanked from your WIPbin by clarify.  
  - Case 10 has been assigned to you by clarify  
  - RE: Case-Subcase Queue Escalation Case 47 has been sitting in queue Support for 4 hours. Please take appropriate action.  
  It will not be able to parse messages such as:  
  - RE: Case-Subcase Queue Escalation Case 47 has been sitting in queue Support Case for 4 hours. Please take appropriate action.  
  (It will give a hyperlink to Open Case 'for') |
| `fc_wip_q_counts`         | **DEFAULT: i_value = 1**  
  Should we include the count of objects for wipbins & queues in the console leftpane? 0 = no; 1 = yes  
  Note: Due to a limitation/bug in the Sybase OLEDB driver, this functionality is not currently supported on Sybase databases. |
| `query_rowlimit`          | **DEFAULT: i_value = 0**  
  Maximum number of rows to be returned in a query (i.e. case query, subcase query, CR query, select address, select contact, select contract, select part (support, quality), select site, select site_part, select solution, fQuery queries) 0 = no limit |
| `fquery_system`           | **DEFAULT: str_value = <blank>**  
  Which system to query. If blank, it's the current system. Otherwise, use ID, PASS, SERVER, DB, where ID is the user id, PASS is the password, SERVER is the query target server, and DB is the query target database. |
| `attachment_style`        | **DEFAULT: i_value = 0**  
  Which style of attachment handling is used (0 = Classic Clarify, 1 = Attachment Anywhere) |
| `flash_style`             | **DEFAULT: i_value = 0**  
  Which style of flash handling is used (0 = Classic Clarify, 1 = Flashes Anywhere) |
<p>| <code>fc_eeo_alt_replyto</code>      | <strong>DEFAULT: str_value = &lt;blank&gt;</strong> |</p>
<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fc_consoe_maxrecords</td>
<td><code>i_value = 0</code></td>
<td>Maximum number of each object type to display in console. If set to zero, show all. For example, if this is set to five, then the console will only show a maximum of five cases, five subcases, five bugs, etc.</td>
</tr>
<tr>
<td>fc_delete_old_messages</td>
<td><code>i_value = 0</code></td>
<td>fcClient can automatically delete old Notifier messages upon login. The <code>fc_delete_old_messages</code> config item defines how old messages need to be (in days) before they are deleted. For example, if <code>fc_delete_old_messages</code> is set to 14, and messages older than 14 days will be deleted upon login. If this config item is set to zero (or if this config item doesn't exist), then old messages are not automatically deleted.</td>
</tr>
<tr>
<td>cl_cfit_dr_loc</td>
<td><code>str_value = “central stores”</code></td>
<td>This is the Default Inventory Location used in Depot Repair. If Depot Repair is already in use in the Clarify client, this configuration item should already be set in the database. If that is true, either remove this item or set the value to match the current value. If it is not set, change the value to point to the correct inventory location.</td>
</tr>
<tr>
<td>cl_cfit_dr_gl</td>
<td><code>str_value = “exp_gl”</code></td>
<td>This is the Default Expense GL Account used in Depot Repair. If Depot Repair is already in use in the Clarify client, this configuration item should already be set in the database. If that is true, either remove this item or set the value to match the current value. If it is not set, change the value to point to the correct inventory location.</td>
</tr>
<tr>
<td>fc_attach_upload_option</td>
<td><code>i_value = 0</code></td>
<td>On the attachments page, do we give users the option of uploading the file to the attachments server or not? 0=No 1=Yes. If no, attachments will always be uploaded and the attachment (doc_instance and doc_path records) will be added in the Clarify database. If yes, a checkbox will be enabled allowing users to choose if the file is uploaded or not. The attachment (doc_instance and doc_path records) will be added in the Clarify database regardless of the checkbox setting.</td>
</tr>
<tr>
<td>fc_dde_server</td>
<td><code>i_value = 0</code></td>
<td>Should fcClient function as a DDE (Dynamic Data Exchange) server? 0=No 1=Yes</td>
</tr>
<tr>
<td>fc_eeo_dont_prefill_to_list</td>
<td><code>i_value = 0</code></td>
<td>By default, the TO list is pre-filled. For example, when logging email on a case, the TO list will be pre-filled with the email address of the contact for the case. The <code>fc_eeo_dont_prefill_to_list</code> configuration item allows this prefill functionality to be turned off. Setting the <code>i_value</code> to 1 will disable the prefilling.</td>
</tr>
<tr>
<td>fc_config_mgr_search_level</td>
<td><code>i_value = 999</code></td>
<td>On the config manager page, by default, how many levels deep should the site part search go?</td>
</tr>
<tr>
<td>fc_force_site_part_search_filter</td>
<td><code>i_value = 0</code></td>
<td>On the config manager page, should filtering be enforced when</td>
</tr>
</tbody>
</table>
| fc_console_default_refresh_rate | DEFAULT: i_value = 120  
Default Refresh Rate of console. |
|-----------------------------|--------------------------|
| fc_console_refresh_rate_limit | DEFAULT: i_value = 60  
The lower limit for the Refresh Rate of console. This allows admins to prevent too frequent refreshes. |

**Set Up Necessary Web Privclasses**

Web privclasses allow you to decide which fcClient operations a specific user (or group of users) may perform. Using the fcAdmin application, you can turn on (or off) virtually any operation in the thin client.

By default, no operations are initially enabled for any privclasses. So, you must use fcAdmin to grant the proper privileges to the appropriate users.

To modify the web privclasses, you must perform the following steps:

1. Start the fcAdmin in your browser and log in
2. Select Manage Priv Classes from the navigation menu on the left
3. Select the appropriate Privilege Class
4. In the lower frame, check the commands that you wish to be enabled for this privclass.
5. Press the “Save” button

**Configure and Set Up Web Server**

Currently, you must have Microsoft IIS web server (version 5.0 or later) installed on the machine that will be running fcClient. For more information on installing IIS, please visit Microsoft at [www.microsoft.com](http://www.microsoft.com).

After the web server is installed, you will need to configure the web server to recognize fcClient. This is easily performed as follows (Note: the following assumes IIS version 5.0 on Microsoft Windows 2000. Your system may vary slightly):

1. Open the Internet Services Manager (ISM). To do this, click on the Start menu, click Run, type inetmgr, and then click OK. The Internet Information Services (ISM) console opens.

2. Click on your computer host name to expand the top-level node. Right-click Default Web Site, click New, and then click Virtual Directory.

3. In the Virtual Directory Creation Wizard, click Next to continue. In the Alias text box, type a short name (alias) for the directory into which the fcClient product was installed (i.e. fcclient). The alias is what users see as part of the URL path when they are browsing to the fcClient web application (for example, http://www.MyCompany.com/\<fcClient>). Click Next. Enter the path, or browse to the directory which contains the fcClient web application files (the pages folder under the installation directory). Click Next, and set the access permissions to Read and Run Scripts. Click Next again, and then click Finish.

4. You must create two more virtual directories (just like the previous step). They should be called ‘uploads’ and ‘attachments’. The ‘uploads’ is just a temporary directory for attachments uploaded on your web server machine; this directory must have access permissions set to Read, Write and Run Scripts. The ‘attachments’ virtual directory should point to the location where your attachments will be stored; this
directory must have access permissions set to Read and Run Scripts. Note: These names ‘uploads’ and ‘attachments’ must match the values you set in the global.asa file step (see below). You can use other names, as long as they match.

5. In the left pane of the ISM, click Default Web Site, and then press the F5 key to refresh the list of virtual folders. Notice that the virtual folder of your newly created application appear under the alias name you supplied previously. Also note that the icon for your virtual folder appears as a green symbol within a gray box. This means that your application is set up and ready to host ASP pages. You can now close the ISM.

6. Set your web app to run in High Isolation Mode. Using Internet Services Manager, select the properties of the virtual directory. Set the Application Protection to High. This will allow each web app running on the same system to run in its own process.

**Edit the fc.env File**

In the pages directory (of the virtual server directory you created in the previous step) you will find a file named fc.env. This file is quite similar to the clarify.env file you are familiar with. It contains the login information that you need to create the FCApplication object for your web server. It contains entries similar to the clarify.env file.

**You should edit this file with the login information for your database.**

Specifically, you need to edit and change the following information:

1. **db_server** – This is the Server name of the DB server (MSSQL), or the SID name (Oracle)
2. **db_name** – This is the name of the database
3. **login_name** – This is the user name to use in creating the application object. It can be most any valid user name, 'sa' is commonly used.
4. **db_password** – This is the password for the login_name
5. **db_type** – This is either “MSSQL” or “ORACLE”. This is different than base Clarify, as fcClient can, out of the box, support either database system – you don’t need a specific version of fcClient for different database types
6. **midnight** – A time string, such as “12:00:00 AM”. This is only used for Oracle 7 databases. This is part of a workaround for a bug in the OLEDB drivers for Oracle that only affects Oracle 7 databases. If you are not using an Oracle 7 database, do not use this parameter.

In addition, when using the fcSDK, you can define a logging configuration file in the fc.env file. For example:

```
compat_logconfigfile=C:\fcClient\fcclient_logging.config
```

The logging configuration file allows for sophisticated logging configuration. Refer to the fcSDK Documentation for full details on configuring logging.

If you wish to use the Switch User functionality with the fcSDK, you must add the following entry to the fc.env file:

```
fchoice,sessionpasswordrequired=false
```

**Note:** You can modify the login code for fcClient so that it does not use the fc.env file. Please see the documentation in the FCFL_product (application object) for how you can do an “InitializeFull” operation.

**Modify the global.asa File**

The global.asa file for the fcClient is located in the pages folder of your fcClient application. Depending on where you unzipped the fcClient files, its path will be similar to C:\first\fcClient\pages\global.asa
You must modify this file for your environment. It is recommended that you first backup this file to a safe location. You can modify this file with your favorite text editor.

Setting the Working Directory

The working directory of the fcClient application must be defined. This will be the directory where your fc.env file is located. By default, fcClient assumes the working directory to be the root directory for fcClient, typically C:\first\fcClient\pages. If this is true for you (which it will be for most users), you do not have to hard-code the path in the global.asa file. If, however, you wish for your working directory to be different, you can set it in the global.asa file. For example:

```
Application.Contents("FCApp").WorkingDirectory = "C:\first\fcClient\pages"
```

Notice the double-backslashes when using an absolute path. These are important, as all path delimiters must be double-backslashes, not single ones, or the path will not be recognized.

Note: The working directory path, by default, is commented out, using two backslashes. If you want to use a hard-coded path, be sure to also uncomment this line.

Setting Global Application Variables

The following global application variables must also be defined for your environment.

**FCApp object**
Defines whether FCFL or FCFL.NET/fcSDK is used. Refer to the First Choice Platforms section for more information.

**bUseFcSDKToolkits**
Defines whether the fcSDK toolkits are to be used. Refer to the First Choice Platforms section for more information.

**attach_dir**
Permanent Storage Location for Attachments. This is a physical path.

**v_attach_dir**
Virtual directory on the server pointing to permanent attachments directory. This virtual directory allows fcClient users to access existing attachments. This must match the name of the virtual directory you created above. Note: This should be the same directory that the Clarify classic client uses (network shares are permitted) so that the systems will interoperate.

**v_upload_dir**
Virtual directory on the server pointing to temporary upload directory. This is where attachments are first uploaded to. If the attachment has a valid file size, it is moved to the permanent storage location. This must match the name of the virtual directory you created above.

**max_attach_size**
Maximum attachment size (in kilobytes). This allows you to prevent very large file uploads.

**bCaseIdIsNumeric**
Is the Numbering Scheme for Cases an Integer? If so, then cases in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.
For example, if your case IDs are similar to “1234”, then this variable should be set to true. If your case IDs are similar to “C1234-01012002”, then this variable should be set to false.

bSubcaseIdsNumeric
Is the Numbering Scheme for Subcases an Integer? If so, then subcases in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.

Note that only the portion of the ID number after the dash must be an integer. For example, if your subcase IDs are similar to “160-12”, then this variable should be set to true.

bSolutionIdsNumeric
Is the Numbering Scheme for Solutions an Integer? If so, then solutions in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.

bCRIdIsNumeric
Is the Numbering Scheme for Change Requests an Integer? If so, then Change Requests in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.

bDemandIdsNumeric
Is the Numbering Scheme for Part Requests an Integer? If so, then Part Request in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.

bAllIdsNumeric
Is the Numbering Scheme for Action Items an Integer? If so, then Action Items in the fcClient WIPbins and Queues will be sorted numerically, rather than alphabetically.

PASSWORD_REQUIRED
Determines if fcClient requires a password when logging in.
If using Integrated Windows Integrated Security, set this to false. Else, leave this set to true.
Refer to the section on Windows Integrated Security for more information.

Install aspSmartUpload
AspSmartUpload is an ASP component that is used by EEO (Enhanced Email Out) and for uploading Attachments. It is a free product available from ADVANTYS (http://www.aspsmart.com/). To make it easier for you, we have bundled aspSmartUpload with fcClient.

The aspSmartUpload components must be installed on the machine that is running the web server and fcClient. After it is downloaded, you must install it with the following steps:

1. Install aspSmartUpload.dll

You have to register aspSmartUpload.dll on your server. The dll can be registered by using regsvr32.exe.

Register the component using a command like:

```
regsvr32 c:\first\fc\client\aspSmartUpload\aspSmartUpload.dll
(From DOS or the Start/Run prompt)
```

2. Install aspSmartUploadUtil.dll
The objective is to enable the Operating System to access the aspSmartUploadUtil.dll file.

There are two ways to achieve this:

1. Copy aspSmartUploadUtil.dll in the c:\wim\system32 directory. That way, you don’t have to update the PATH property.
2. Copy aspSmartUploadUtil.dll into a directory on the NT server (e.g. c:\mydir). If you do this, you must ensure that the directory used is in the list of directories in the PATH environment variable.

Note: You can (but do not need to) download the current version from ADVANTYS directly from http://www.aspsmart.com.

Note: You only have to install aspSmartUpload if you wish to have your users use EEO or Attachments. If you turn off the EEO and Attachments functionality (using Privilege Classes in fcAdmin), you do not have to perform this step.

Setup FTS Client Software
The fcClient essentially acts as an FTS client. So, you should follow the client installation instructions detailed in the FTS Administration Guide located on your Clarify documentation CD. When setting up the ODBC connector, it is recommended that you create this as a System DSN.

Note: If you are not using FTS, this step is not necessary.

Setup Flashes Anywhere
If you wish to use Flashes Anywhere functionality in fcClient, you must perform the following steps:

1. Make the schema changes associated with Flashes Anywhere. These changes are found in the fcclient_schema.sch file. There are three table modifications for the existing table: alert (table id 4300), two new tables: alert_to_other (table id 4300) and alert_table (table id 4301), and a new view: alert_view (table id 4302). (Note: These changes may have been made already if you are installing fcClient for the first time.)
2. If classic flashes have been used previously in fcClient or in the Clarify client, convert the existing classic Clarify flashes to use them with Flashes Anywhere (see Flashes Anywhere documentation).
3. Configure fcClient for Flashes Anywhere functionality by modifying the flash_style configuration item in the fcclient_config.dat file, and then import this dat file into your database using dataex or DIET.
4. Optionally import the fcclient_alert_tables.dat (located in the files directory) file to quickly create the baseline source objects that Clarify supports.

Note: If you are not using Flashes Anywhere, these steps are not necessary.

Setup Attachments Anywhere
If you wish to use Attachments Anywhere functionality in fcClient, you must perform the following steps:

1. Convert existing Classic Clarify attachments (contact First Choice for assistance)
2. Configure fcClient for Attachments Anywhere functionality by modifying the attachment_style configuration item in the fcclient_config.dat file, and then import this dat file into your database using dataex or DIET.

Note: If you are not using Attachments Anywhere, these steps are not necessary.

Setup Zipcodes
If you wish to use the zipcode feature of feClient, you should upload the zipcode data you have purchased into table_zipcode in the Clarify database. Loading this data is a very simple task, as table_zipcode has a simple format, and no external relations, but is a customized process, depending on the data source you purchase.

The following is the data schema for the crucial fields for table_zipcode:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objid</td>
<td>Long Integer</td>
<td>Clarify unique index field</td>
</tr>
<tr>
<td>zipcode</td>
<td>varchar(30)</td>
<td>Zipcode or postal code</td>
</tr>
<tr>
<td>city</td>
<td>varchar(30)</td>
<td>City for the zipcode</td>
</tr>
<tr>
<td>state</td>
<td>varchar(20)</td>
<td>State for the zipcode</td>
</tr>
<tr>
<td>country</td>
<td>varchar(40)</td>
<td>Country for the zipcode</td>
</tr>
<tr>
<td>time_zone</td>
<td>varchar(20)</td>
<td>Time zone for the zipcode</td>
</tr>
</tbody>
</table>

Please see the section above for more information about running the zipcode interface to load zipcode data into the database.
DOVETAIL AGENT INSTALLATION SECTION AFTER THE REVISION

Note: The final Dovetail Agent user guide was delivered as HTML and CHM files. The documentation has changed since the conclusion of my internship, but the latest documentation is publicly viewable here:

To improve the readability of the installation section for this report, I chose to present it as it appears in a Word document.

INSTALLING DOVETAIL AGENT

Installation Notes and Recommendations

- Before installing Dovetail Agent on a production system, we recommend installing Dovetail Agent on a test system first to get familiar with it.
- Install the Dovetail Agent files on each server that will act as a web server for the program.
- Uncompress the Dovetail Agent zip file into a dovetail subdirectory. In this document, all examples will use "c:\dovetailagent" as the install folder.

To complete the installation, follow the steps in this section.

STEP 1 - INSTALL THE DOVETAIL SDK

1) Install the Dovetail SDK. The Dovetail SDK includes FCLNET and the API toolkits. Please see the Dovetail SDK documentation (http://www.dovetailsoftware.com/resources/docs.aspx?product=DovetailSDK) for more details.

2) Use the License Installer to install your license key(s).

3) Run the Data Verifier to check that there are no inconsistencies or bad data that might prevent FCL.NET from initializing properly. The Data Verifier and License Installer are shipped with the Dovetail SDK.

STEP 2 - UPDATE THE CLARIFY SCHEMA

You can change the Clarify schema with either Dovetail's SchemaEditor or Clarify's DDEditor.

Prior to making schema changes, backup the current database.

To make these changes using Dovetail SchemaEditor:

1) Edit the .SchemaEditor file
   - Set the database connection information.
   - Set the inputFilePath to fclient_schemascript.xml

2) Preview the changes (SchemaEditor.exe -p).

3) Apply the changes (SchemaEditor.exe -a).

4) If your Clarify version is 8.0 or higher, repeat steps 1-3 using fclient_clarify_v8_and_up.schemascript.xml as the inputFilePath.
If you plan to use data restriction with Dovetail Agent, you must also apply that schema. If you plan to use the sample that is shipped with this product, repeat the steps above with the real/FcClient_data_restrictions_schemascript.xml file.

To make these changes using Clarity DDEditor:
1) Start the **DD Editor** program.
2) To save a copy of the database schema to disk, select **Save**.
3) Edit the saved schema file.
4) Edit the fclient_schemas.sch file. Follow the directions in that file to cut/paste the schema changes into the saved schema file. Use the **Apply Changes menu item** to compile the new saved schema file.
5) Exit **DD Editor**.

If you plan to use data restriction with Dovetail Agent, you must also apply that schema. If you plan to use the sample that is shipped with this product, repeat the steps above with the real/sample_fcs_schemas.sch file.

---

**STEP 3 - ADD THE DATABASE TRIGGER**

Adding the database trigger is required for the Dovetail Agent Notifier. The steps to add the database trigger vary depending on your type of database and are described in this section.

**ADDING THE DATABASE TRIGGER FOR ORACLE DATABASES**

Compile the provided trigger (`ora_trig.sql`).

- One way of performing the compilation:
  
  ```sql
  sqlplus sa/<pass>@<sid> &ora_trig.sql
  where
  <pass> is sa's password
  <sid> is the name of the database SID (security ID)
  ```

  When finished, you should see a message that the trigger was created. If there are any error messages, fix the error and recompile.
ADDING THE DATABASE TRIGGER FOR MICROSOFT SQL SERVER AND SYBASE DATABASES

Compile the provided trigger (mssql_trig.sql or syb_trig.sql).

- One way of performing the compilation:
  
  isql -Usa -P<pass> -S<server> -d<database> < <sql_file>
  
  where:
  
  <pass> is sa's password
  
  <server> is the name of the database server
  
  <database> is the name of the database
  
  <sql_file> is the name of the SQL file

When finished, you should see a message that the trigger was created. If there are any error messages, fix the error and recompile.

STEP 4 - ADD THE FC_WIP_Q_COUNTS STORED PROCEDURE

Stored procedures are used to count the objects in wipbins and queues. The steps for adding the fc_wip_q_counts stored procedure vary depending on your type of database and are described in this section.

ADDING THE STORED PROCEDURE FOR ORACLE DATABASES

Compile the provided stored procedure (ora_fc_wip_q_counts.sql).

- One way of performing the compilation:
  
  sqlplus sa/<pass>0/<sid> ora.fc_wip_q_counts.sql
  
  where:
  
  <pass> is sa's password
  
  <sid> is the name of the database SID (security ID)

When finished, you should see a message that the stored procedure was created. If there are any error messages, fix the error and recompile.

ADDING THE STORED PROCEDURE FOR MICROSOFT SQL SERVER AND SYBASE DATABASES

Compile the appropriate stored procedure (mssql_fc_wip_q_counts.sql or syb_fc_wip_q_counts.sql).

- One way of performing the compilation:
  
  isql -Usa -P<pass> -S<server> -d<database> < <sql_file>
  
  where:
  
  <pass> is sa's password
  
  <server> is the name of the database server
  
  <database> is the name of the database
  
  <sql_file> is the name of the stored procedure file

If there are any error messages once completed, fix the error and recompile.
**STEP 5 - SET CONFIGURATION ITEMS**

Review and set the configuration items in the *fcclient_config.dat* file, which can be modified with any text editor. After saving your changes, import the configuration file into the Clarify database as described in the *Import Data Files* (see "Step 6 - Import Data Files" on page 36) section.

<table>
<thead>
<tr>
<th>Configuration Item Name</th>
<th>Description and Valid Values</th>
</tr>
</thead>
</table>
| `default_fc_notifier_frequency` | **DEFAULT: i_value = 60**  
Default fcNotifier Frequency - How often to check for new messages (number of seconds) |
| `open_item_from_notifier`    | **DEFAULT: i_value = 0**  
Should the fcNotifier try to parse the message and allow the item to be opened? (0 = no; 1 = yes)  
By setting this config item to 1, the fcNotifier will try to parse the message and will display a hyperlink to open the item. The success of this parsing is highly dependent on the notification message.  
It can successfully parse messages such as:  
- The Case 9 was yanked from your Wipbin by Clarify.  
- Case 10 has been assigned to you by Clarify  
- RE: Case-Subcase Queue Escalation Case 47 has been sitting in queue Support for 4 hours. Please take appropriate action.  
It will not be able to parse messages such as:  
- RE: Case-Subcase Queue Escalation Case 47 has been sitting in queue Support Case for 4 hours. Please take appropriate action. (It will give a hyperlink to Open Case For) |
| `fc_wip_q_counts`            | **DEFAULT: i_value = 1**  
Should we include the count of objects for wipbins & queues in the console left pane? (0 = no; 1 = yes)  
**Note:** Due to a limitation/bug in the Sybase OLEDB driver, this functionality is not currently supported on Sybase databases. |
| `query_rowlimit`             | **DEFAULT: i_value = 0**  
Maximum number of rows to be returned in a query (i.e. case query, subcase query, CR query, select address, select contact, select contract, select part (support, quality), select site, select site_part, select solution, fcQuery queries) 0 = no limit |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcquery_system</td>
<td><strong>DEFAULT: str_value = blank</strong></td>
<td>Which system to query. If blank, it's the current system. Otherwise, use: ID</td>
</tr>
<tr>
<td></td>
<td>where:</td>
<td>Where: ID = the user id, PASS = the password, SERVER = the query target server, DB = the query target database</td>
</tr>
<tr>
<td>attachment_style</td>
<td><strong>DEFAULT: i_value = 0</strong></td>
<td>Which style of attachment handling is used? (0 = Classic Clarify, 1 = Attachment Anywhere)</td>
</tr>
<tr>
<td>flash_style</td>
<td><strong>DEFAULT: i_value = 0</strong></td>
<td>Which style of flash handling is used? (0 = Classic Clarify, 1 = Flashing Anywhere)</td>
</tr>
<tr>
<td>fc_eeo_alt_replyto</td>
<td><strong>DEFAULT: str_value = blank</strong></td>
<td>Reply-To address for EEO to use. If blank, EEO will use the current sender's email address. For example, if you want replies to an EEO email to go to your support email alias, set this value to be &quot;<a href="mailto:support@mycompany.com">support@mycompany.com</a>&quot;. This is a global setting for all users.</td>
</tr>
<tr>
<td>fc_console_maxrecords</td>
<td><strong>DEFAULT: i_value = 0</strong></td>
<td>Maximum number of each object type to display in console. If set to 0, all will be shown. If, for example, it is set to 5, then the console will only show a maximum of 5 cases, 5 subcases, 5 bugs, etc.</td>
</tr>
<tr>
<td>fc_delete_old_messages</td>
<td><strong>DEFAULT: i_value = 0</strong></td>
<td>Dovetail Agent can automatically delete old Notifier messages upon login. The fc_delete_old_messages configuration item defines how old messages need to be (in days) before they are deleted. For example, if fc_delete_old_messages is set to 14, then messages older than 14 days will be deleted upon login. If this configuration item is set to 0 (or if this config item doesn't exist), then old messages are not automatically deleted.</td>
</tr>
<tr>
<td>cl_dift_dr_loc</td>
<td><strong>DEFAULT: str_value = &quot;central stores&quot;</strong></td>
<td>This is the Default Inventory Location used in Depot Repair. If Depot Repair is already in use in the Clarify client, this configuration item should already be set in the database. If that is true, either remove this item or set the value to match the current value. If it is not set, change the value to point to the correct inventory location.</td>
</tr>
<tr>
<td>Variable</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cl_dflt_dr_gl</td>
<td>DEFAULT: <code>str_value = &quot;exp_gl&quot;</code></td>
<td>This is the Default Expense GL Account used in Depot Repair. If Depot Repair is already in use in the Clarify client, the configuration item should already be set in the database. If that is true, either remove this item or set the value to match the current value. If it is not set, change the value to point to the correct inventory location.</td>
</tr>
<tr>
<td>fc_attach_upload_option</td>
<td>DEFAULT: <code>i_value = 0</code></td>
<td>On the attachments page, are users given the option of uploading the file to the attachments server or not? 0=No 1=Yes. If no, attachments will always be uploaded and the attachment (doc_inst and doc_path records) will be added in the Clarify database. If yes, a checkbox will be enabled allowing users to choose if the file is uploaded or not. The attachment (doc_inst and doc_path records) will be added in the Clarify database regardless of the checkbox setting.</td>
</tr>
<tr>
<td>fc_dde_server</td>
<td>DEFAULT: <code>i_value = 0</code></td>
<td>Should Dovetail Agent function as a DCE (Dynamic Data Exchange) server? (0 = no; 1 = yes).</td>
</tr>
<tr>
<td>fc_eeo_dont_prefill_to_list</td>
<td>DEFAULT: <code>i_value = 0</code></td>
<td>By default, the TO list is pre-filled. For example, when logging email on a case, the TO list will be pre-filled with the email address of the contact for the case. The <code>fc_eeo_dont_prefill_to_list</code> configuration item allows this prefill functionality to be turned off. Setting the i_value to 1 will disable the prefilling.</td>
</tr>
<tr>
<td>fc_config_mgr_search_level</td>
<td>DEFAULT: <code>i_value = 999</code></td>
<td>On the configuration manager page, by default, how many levels deep should the site part search go?</td>
</tr>
<tr>
<td>fc_force_site_part_search_filter</td>
<td>DEFAULT: <code>i_value = 0</code></td>
<td>On the config manager page, should filtering be enforced when searching for site parts? (0 = no filtering required, 1 = enforce filtering).</td>
</tr>
<tr>
<td>fc_console_default_refresh_rate</td>
<td>DEFAULT: <code>i_value = 120</code></td>
<td>Default Refresh Rate of console in seconds.</td>
</tr>
<tr>
<td>fc_console_refresh_rate_limit</td>
<td>DEFAULT: <code>i_value = 60</code></td>
<td>The lower limit for the Refresh Rate of console. This allows administrators to prevent refreshes that are too frequent.</td>
</tr>
</tbody>
</table>
Import these data files found in the files directory, which contain web privileges, configuration, and list information needed for Dovetail Agent:

- fcclient_basic_data.dat
- fcclient_colors.dat
- fcclient_config.dat [see Set Configuration Items (see "Step 5 - Set Configuration Items" on page 32) for more information]
- fcclient_quality_data.dat
- fcclient_support_data.dat
- fcclient_calcenter_data.dat
- fcclient_logistics_data.dat
- fcclient_web_forms.dat
- fcclient_sales_data.dat

**Import these additional data files if...**

- If using Clarify version 7 or below: fcclient_num_scheme.dat
- If using the pre-supplied data restriction provided with Dovetail Agent: sample_res_data.dat
- If adding Flashes Anywhere functionality and converting from Classic Flashes, import this data file to set up the basic alert table objects in the database: fcclient_alert_table.dat

**To import the data files using Dovetail ArchiveManager (Diet)**

Execute the following command:

```
<path>\diet.exe
-user_name <user>
-password <pass>
-do_server <serv>
-do_name <db>
-import <file>
```

where:
- `<path>` is the path to the dataex program
- `<user>` is the system administrator user
- `<pass>` is the system administrator password
- `<serv>` is the database server name
- `<db>` is the database name
- `<file>` is the name of the file to import. If the `<file>` is not in the current directory, specify the path to the directory it is in.
To import the data files using dataex

Execute the following command:

```
<path>\dataex
-user_name <user>
-password <pass>
-db_server <serv>
-db_name <db>
-imp <file>
```

where:
- `<path>` is the path to the dataex program
- `<user>` is the system administrator user
- `<pass>` is the system administrator password
- `<serv>` is the database server name
- `<db>` is the database name
- `<file>` is the name of the file to import. If the `<file>` is not in the current directory, specify the path to the directory it is in.

**Note:** Each file should import with 0 errors and 0 warnings. If there are any errors or warnings, examine the dataex.err file for possible problems.

---

**STEP 7 - SET UP WEB PRIVCLASSES**

Web privclasses allow you to decide which Dovetail Agent operations a specific user (or group of users) may perform. Using Dovetail Admin, you can turn on (or off) virtually any operation in the thin client.

By default, no operations are initially enabled for any privclasses. You must use Dovetail Admin to grant privileges to users.

**To modify the web privclasses**

1. Start Dovetail Admin in your browser and log in.
2. From the navigation menu on the left, select **Privilege Classes**.
3. Find and select the Privilege Class you want to modify.
4. In the lower frame, check the commands to be enabled for this privclass.
5. Select **Save**.

---

**STEP 8 - CONFIGURE THE WEB SERVER**

Configure the web server to support Dovetail Agent:

**Note:** The following assumes IIS version 5.0 on Microsoft Windows 2000. Your system may vary slightly.

1. To open the **Internet Services Manager (ISM)**, select **Start** and **Run**.
2) Enter inetmgr, and then select OK. The Internet Information Services (ISM) console will open.

3) Click on your computer host name to expand the top-level node.

4) Right-click Default Web Site and select New and Virtual Directory.

5) In the Virtual Directory Creation Wizard, select Next to continue.

6) In the Alias text box, type a short name (alias) for the directory into which the Dovetail Agent product was installed (i.e. agent). The alias is what users see as part of the URL path when they are browsing to the Dovetail Agent web application (for example, http://www.MyCompany.com/<agent>).

7) Select Next.

8) Enter the path, or browse to the directory which contains the Dovetail Agent web application files (the pages folder under the installation directory).

9) Select Next and set the access permissions to Read and Run Scripts.

10) Select Next and then Finish.

11) To create two more required virtual directories, use the Virtual Directory Creation Wizard as you did in steps 5-10. These directories should be named ‘uploads’ and ‘attachments’. The names ‘uploads’ and ‘attachments’ must match the values you set in the global.asa file step (see the Modify the global.asa file (see “Step 10 - Modify the global.asa File” on page 41) section for more information). You can use other names, as long as they match. The ‘uploads’ directory is just a temporary directory for attachments uploaded on your web server machine; this directory must have access permissions set to Read, Write, and Run Scripts. The ‘attachments’ virtual directory should point to the location where your attachments will be stored; this directory must have access permissions set to Read and Run Scripts.

12) In the left pane of the ISM, select Default Web Site, and then press the F5 key to refresh the list of virtual folders.

Note: Notice that the virtual folder of your newly created application appears under the alias name you supplied previously. Also note that the icon for your virtual folder appears as a green symbol within a gray box. This means that your application is set up and ready to host ASP pages.

13) Close the ISM.

14) Set your web app to run in High Isolation Mode by using the ISM to select the properties of the virtual directory.

15) Configure Application Isolation

   a. If using Windows XP or Windows 2000, set the Application Protection to High. This will allow each web app running on the same system to run in its own process.

   b. If using Windows 2003, create a new application pool and place the Dovetail Agent application in this application pool.

      • Right-click, new Application Pool, name it appropriately, such as “Dovetail Agent pool”

      • Right-click on the root of your Dovetail Agent application, choose properties, in the application pool drop-down list, select the new application pool you just created, and choose OK.

60
STEP 9 - EDIT THE FC.ENV FILE

In the pages directory of the virtual server directory you created in Step 8 (see "Step 6 - Configure the Web Server" on page 38), you will find a file named fc.env. This file is very similar to the datify.env file that you are familiar with. It contains the login information that you need to create the FCApplication object for your web server.

Edit this file with the login information for your database.

- **db_server** - The Server name of the DB server (MSSQL), or the SID name (Oracle)
- **db_name** - The name of the database
- **login_name** - The username to use when creating the application object. It can be most any valid username, 'sa' is commonly used.
- **db_password** - The password for the login_name
- **db_type** - This is either "MSSQL" or "ORACLE". This is different than base Clarify, because Dovetail Agent supports either database system. - you don't need a specific version of Dovetail Agent for different database types
- **midnight** - A time string, such as "12:00:00 AM". This is only used for Oracle 7 databases. This is part of a workaround for a bug in the OLEDB drivers for Oracle that only affects Oracle 7 databases. If you are not using an Oracle 7 database, do not use this parameter.

To define a logging configuration file

When using the Dovetail SDK, you can define a logging configuration file in the fc.env file. For example:

```
compat_logconfigfile=C:\dovetail\agent\fcclient\logging.config
```

The logging configuration file allows for sophisticated logging configuration. Refer to the Dovetail SDK documentation (http://www.dovetailsoftware.com/resources/docs.aspx?product=DovetailSDK) for more information about configuring logging.

To use the Switch User functionality

To use the Switch User functionality with the Dovetail SDK, add the following entry to the fc.env file:

```
fchoce.sessionpasswordrequired=false
```
**STEP 10 - MODIFY THE GLOBAL.ASA FILE**

Before modifying the `global.asa` file for your environment, we recommend that you first back it up to a safe location. It can be modified with any text editor.

The `global.asa` file for Dovetail Agent is located in the `pages` folder of your Dovetail Agent application. Depending on where you unzipped the Dovetail Agent files, its path will be similar to:

```
C:\dovetail\agent\pages\global.asa
```

Define the following global application variables for your environment:

<table>
<thead>
<tr>
<th>Application Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>attach_dir</code></td>
<td>Permanent Storage Location for Attachments. This is a physical path.</td>
</tr>
<tr>
<td><code>v_attach_dir</code></td>
<td>Virtual attachments directory on the server pointing to the permanent attachments directory. This virtual directory allows Dovetail Agent users to access existing attachments. This virtual directory was created in the <strong>Configure the Web Server</strong> (see “Step 8 - Configure the Web Server” on page 38) step.</td>
</tr>
<tr>
<td><code>v_upload_dir</code></td>
<td>Virtual uploads directory on the server pointing to the temporary upload directory. This is where attachments are first uploaded to. If the attachment has a valid file size, it is moved to the permanent storage location. This virtual directory was created in the <strong>Configure the Web Server</strong> (see “Step 8 - Configure the Web Server” on page 38) step.</td>
</tr>
<tr>
<td><code>max_attach_size</code></td>
<td>Maximum attachment size (in kilobytes). This allows you to prevent very large file uploads.</td>
</tr>
</tbody>
</table>
| `bCaseIdsNumeric`    | Is the Numbering Scheme for Cases an integer?  
  - True = Cases in the Dovetail Agent Wipbins and Queues will be sorted numerically  
  - False = Cases in the Dovetail Agent Wipbins and Queues will be sorted alphabetically  

For example, if your case IDs are similar to "1234", then this variable should be set to true. If your case IDs are similar to "C1234-01012002", then this variable should be set to false.  

---

Note: This should be the same directory that the Clarify Classic Client uses (network shares are permitted) so that the systems will interoperate.
<table>
<thead>
<tr>
<th><strong>bSubcaseIdIsNumeric</strong></th>
<th>Is the Numbering Scheme for Subcases an Integer?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True = Subcases in the Dovetail Agent Wipbins and Queues will be sorted numerically</td>
</tr>
<tr>
<td></td>
<td>False = Subcases in the Dovetail Agent Wipbins and Queues will be sorted alphabetically</td>
</tr>
</tbody>
</table>

*Note*: Only the portion of the ID number after the dash must be an integer. For example, if your subcase IDs are similar to "100-12", then the variable should be set to true.

<table>
<thead>
<tr>
<th><strong>bSolutionIdIsNumeric</strong></th>
<th>Is the Numbering Scheme for Solutions an Integer?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True = Solutions in the Dovetail Agent Wipbins and Queues will be sorted numerically</td>
</tr>
<tr>
<td></td>
<td>False = Solutions in the Dovetail Agent Wipbins and Queues will be sorted alphabetically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bCRIIdIsNumeric</strong></th>
<th>Is the Numbering Scheme for Change Requests an Integer?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True = Change Requests in the Dovetail Agent Wipbins and Queues will be sorted numerically</td>
</tr>
<tr>
<td></td>
<td>False = Change Requests in the Dovetail Agent Wipbins and Queues will be sorted alphabetically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bDemandIdIsNumeric</strong></th>
<th>Is the Numbering Scheme for Part Requests an Integer?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True = Part Requests in the Dovetail Agent Wipbins and Queues will be sorted numerically</td>
</tr>
<tr>
<td></td>
<td>False = Part Requests in the Dovetail Agent Wipbins and Queues will be sorted alphabetically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bAIIIdIsNumeric</strong></th>
<th>Is the Numbering Scheme for Action Items an Integer?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True = Action Items in the Dovetail Agent Wipbins and Queues will be sorted numerically</td>
</tr>
<tr>
<td></td>
<td>False = Action Items in the Dovetail Agent Wipbins and Queues will be sorted alphabetically</td>
</tr>
</tbody>
</table>

| **PASSWORD_REQUIRED** | Determines if Dovetail Agent requires a password when logging in. If using Integrated Windows Integrated Security, set this to false. Else, leave this set to true. Refer to the section on *Integrated Windows Security* (on page 54) for more information. |
**OPTIONAL - INSTALL ASPSMARTUPLOAD**

**Note:** Only install aspSmartUpload if your users use EEO or Attachments. If you turn off the EEO and Attachments functionality (using Privilege Classes in Dovetail Admin), then installing aspSmartUpload is not necessary.

aspSmartUpload is an ASP component used by EEO (Enhanced Email Out) and for uploading Attachments. It is a free product available from ADVANTYS, which we have bundled with Dovetail Agent for your convenience.

**Note:** Install the aspSmartUpload components on the machine running the web server and Dovetail Agent.

1) Install aspSmartUpload.d1l.
2) Register aspSmartUpload.dll on your server. The dll can be registered by using regsvr32.exe.  
   **For example:** Register the component using a command like:  
   regsvr32 c:\dovetail\agent\aspSmartUpload\aspSmartUpload.dll
3) Install aspSmartUploadUtil.dll. The objective is to enable the Operating System to access the aspSmartUploadUtil.dll file. There are two ways to achieve this:  
   - Copy aspSmartUploadUtil.dll in the c:\winnt\system32 directory. That way, you don't have to update the PATH property. -OR-
   - Copy aspSmartUploadUtil.dll into a directory on the NT server (e.g. c:\anydir). If you do this, make sure that the directory used is in the list of directories in the PATH environment variable.

**OPTIONAL - SET UP FULL TEXT SEARCH (FTS) CLIENT SOFTWARE**

**Note:** If you are not using FTS, this step is not necessary.

Dovetail Agent essentially acts as an FTS client. Follow the client installation instructions detailed in the FTS Administration Guide located on your Clarity documentation CD. When setting up the ODBC connector, it is recommended that it is created as a System DSN.

**OPTIONAL - SET UP FLASHES ANYWHERE**

**Note:** Setting up Flashes Anywhere functionality is optional.

To set up Flashes Anywhere:

1) Make the schema changes associated with Flashes Anywhere. These changes are found in the provided schema files and may have been made already if you are installing Dovetail Agent for the first time. Three table modifications for the existing table are required:
alert (table id 4300)

two new tables: alert_to_other (table id 4300) and alert_table (table id 4301)

a new view: alert_view (table id 4302).

2) If classic flashes have been used previously in Dovetail Agent or in the Clarify client, convert the existing classic Clarify flashes to use them with FlashesAnywhere (see FlashesAnywhere documentation).

3) Configure Dovetail Agent for FlashesAnywhere functionality by modifying the flash_style configuration item in the client_config.dat file, and then import this dat file into your database using dataex or ArchiveManager.

4) Optionally import the fclient_alert_tables.dat (located in the files directory) file to quickly create the baseline source objects that Clarify supports.

**OPTIONAL—SET UP ATTACHMENTS ANYWHERE**

**Note:** Setting up Attachments Anywhere is optional.

To set up Attachments Anywhere:

1) Convert existing Classic Clarify attachments (contact Dovetail Software (see "Getting Help" on page 32) for assistance).

2) Configure Dovetail Agent for Attachments Anywhere functionality by modifying the attachment_style configuration item in the client_config.dat file, and then import this dat file into your database using dataex or ArchiveManager.

**OPTIONAL—SET UP ZIP CODES**

**Note:** Setting up Zip codes is optional.

Dovetail Agent supports automatic zip code lookup. This section describes how to set up zip codes in Dovetail Agent.
Purchasing Zip Code Data

Dovetail Agent does not ship with predefined zip code data. This data can be inexpensively purchased from many different sources.

One company that Dovetail has used in the past for zip code data is zipinfo.com (http://www.zipinfo.com/). They have good prices and support. Their product, ZIPLIST5, contains all of the data needed for the zip code feature in Dovetail Agent.

After the zip code data is purchased, import it into your system. If you purchased the product listed above, you can use the interface provided with Dovetail Agent. See the Load Zip Code Data (see "Loading Zip Code Data" on page 45) section to find out how to use the interface. If you have purchased a different data source, you can edit and modify the supplied interface to load the data into your system.

Loading Zip Code Data

Run the zip code interface supplied with Dovetail Agent to load zip code data into table_zipcode

The interface is located in the zip code directory and consists of two mandatory files:

- fe.env - This is a fe.env file for FCFL. Edit it with values to connect to your database.
- load.zip.js - This is the interface program. It is a FCFL-based javascript program. It will read the input file and load the zip code records into your database.

Three other files can be created to assist with the interface: the country, state, and time_zone mapping files. See Creating a Mapping File (on page 46) for more information.

To run the interface:

1) Edit the fe.env file with the correct login parameters.
2) Copy the zip code data file to the zip code directory.
3) Rename the file zipcode_in, or edit the interface program and change the name of the input file in the program to match the file you have supplied.
4) Execute the interface. The command line to execute the interface is:
   cscript /nologo load_zip.js <delete_old>
   where <delete_old> is either true or false. If <delete_old> is set to true, all of the records in table_zipcode will be deleted before the new data is loaded. The default value is false.
The program prints out its status as it runs. The interface also generates several output files you can read for more detailed information:

- **status_zip.log** - This file contains the status of the execution with summary statistics on the number of records processed and bypassed and the number of errors and warnings encountered.
- **error_zip.log** - Any significant errors are listed in this file, including the line number of the input line in the data file that caused the error.
- **warn_zip.log** - Any data rows that create warnings are reported in this file, along with the input file line number.
- **badrec_zip.log** - If there are any warnings reported (for example, a data row is missing the required zip code field), the entire data row will be copied into this file. This allows you to go into the bad record file with an editor, edit the rows, and use this file as the input for another run of the interface.

**CREATING A MAPPING FILE**

It is often the case that the data source may have different names than what is stored in the Clarify database. For example, your data load may list a country as "United States." In Clarify, it's listed as "USA."

Create a mapping file to help with this process. The mapping file default names, which can be changed in the interface, are:

- **country_map.dat**
- **state_map.dat**
- **timezone_map.dat**

Each of these three files has the same format: each line in the file is a mapping. Each line consists of a "from" name and a "to" name, separated by a comma. For example, the following line could be used for country mapping:

- United States, USA
- UK, United Kingdom

These files can be as long as needed, and a file can be created for any of the three. If the file does not exist in the interface directory, no mapping will occur.

**OPTIONAL—ADD INDEXES FOR THE INTEGRATED NOTIFIER**

**Note:** Adding these indexes is optional.

If you plan to use the Integrated Notifier, adding the following optional indexes will optimize Dovetail Agent's performance:

<table>
<thead>
<tr>
<th>Table</th>
<th>Column(s)</th>
<th>Index Name</th>
<th>Unique?</th>
</tr>
</thead>
<tbody>
<tr>
<td>fc_message</td>
<td>user_name</td>
<td>fc_message_user_name_index</td>
<td>No</td>
</tr>
<tr>
<td>message</td>
<td>user_name</td>
<td>fc_message_user_name_index</td>
<td>No</td>
</tr>
</tbody>
</table>
Dovetail CRM
Taking Customer Service to the Next Level

Dovetail SOFTWARE
Dovetail delivers high performance, enterprise web solutions that are easy to implement, maintain, extend, and integrate

“Therma-Wave selected Dovetail CRM because it delivers high performance even in low-bandwidth situations, because it is easy and inexpensive to maintain and upgrade, and because of its consistent development platform. Service representatives in remote global offices can reliably connect to the database using locally purchased machines, and we can quickly and safely adapt the application to meet their changing needs.”

Michael Campbell, Therma-Wave Chief Information Officer

If your Amdocs/Clarify CRM system is being outpaced by your quickly changing business and competitive challenges, now is the time to implement Dovetail CRM. Dovetail CRM delivers a comprehensive package that includes customer service and support, logistics management, web services, customer self service, and knowledge management capabilities. We are experts at closing the gap between where you are and where you need to be. Our web-based, customer-driven solutions are easy to customize, highly scalable, and can seamlessly integrate into existing enterprise applications.

Today, when investing in technology solutions and services, it’s all about product agility with a low TCO (total cost of ownership) and a measurable ROI (return on investment). Dovetail CRM delivers this and more. Our solutions will increase productivity, lower costs, improve customer satisfaction, and extend both the life and usefulness of existing Amdocs/Clarify deployments. This, combined with our impeccable professional services, translates to no risk solution to your legacy Amdocs/Clarify CRM problems.
Dovetail is the answer for companies that want to take customer service and support to the next level while leveraging their Amdocs/Clarify investment

"The implementation is an important element of our strategic CRM plan. It ensures we are leveraging the best possible return from our CRM investments. The enhanced .NET application server offers a significant performance improvement over an already solid Dovetail application platform. We could not be more pleased to have this strategic partnership and the quality support that Dovetail's staff provides."

Shawn Zudal, MICROS Vice President of MIS

Dovetail offers the next generation CRM solutions for Amdocs/Clarify customers. The transition from Amdocs/Clarify to Dovetail is smooth and simple, and it has many benefits:

- **Dramatic Cost Reductions**: Dovetail’s purchase, implementation, and maintenance costs are considerably lower than Amdocs/Clarify, ensuring a lower Total Cost of Ownership (TCO) and a rapid ROI.
- **High Performance**: Dovetail’s browser-based solutions offer quick response times and are written from the ground up for rapid and scalable deployment on the .NET platform.
- **Agility**: Dovetail easily extends to and integrates with other platforms. Our large library of APIs combined with Dovetail WebServices offers low-risk, quick ways to create custom applications in-house.
- **More Clarify than Clarify**: Dovetail has proven Amdocs/Clarify expertise. For over a decade, we have provided reliable and flexible enhancement solutions for Amdocs/Clarify implementations. Dovetail products offer support for all baseline Amdocs/Clarify functions, along with a complete set of administrative tools. Our applications look and behave just like traditional Amdocs/Clarify products, so there’s a zero learning curve and no re-training cost.
- **Professional Services**: Dovetail knows that organizations often don’t have the time or resources available to manage or customize their Amdocs/Clarify CRM environment. Dovetail’s professional service consultants can work with your organization to increase efficiency or provide a turnkey solution, freeing up valuable resources for other major initiatives.

Dovetail CRM Site Part Configuration
Dovetail Services will rejuvenate your customer relationships

"As we have implemented converged voice, video and data services over a single connection throughout our systems, Dovetail Software’s ticketing generation system was the logical choice. Dovetail Software offered us a cost-effective and efficient solution that seamlessly integrates with our IFO initiative and Fast-Track Services."

Barbara Switzer, Global Crossing’s senior manager of applications programming

With Dovetail Services, you will gain a wealth of expertise that can help you jumpstart a project no matter what your Amdocs/Clarity CRM strategy or initiative may be.

**Implementation:** Dovetail offers a continuum of implementation options from ad hoc development and business analysis resources to complete turnkey, managed implementations. Our experienced staff follows a proven methodology that ensures a timely, trouble-free deployment.

**Managed Services:** With the number of accomplished Clarify practitioners becoming more scarce and expensive, outsourcing the management of your Clarify or Dovetail system to Dovetail has never made more sense. We can structure agreements that cover anything from a complete outsourcing of the day-to-day upkeep and enhancement of the entire CRM system to agreements that only cover part-time staffing under your management.

**Custom Development:** Not every business runs the same. The Dovetail team is keenly aware of this fact, and if needed, we offer custom development services that include seamless integration with other business applications.

**Training:** We regularly offer hands-on training programs, taught by expert-level developers, designed to provide the necessary skills for customization and integration of Dovetail solutions. You will gain the critical knowledge you’ll need to maintain and enhance our solution as you move forward.

**Customer Support:** We provide responsive, convenient, and reliable support. When you contact Dovetail, you can expect a highly skilled engineer to give you the necessary information to resolve your problem quickly.

**Clarify Help Desk:** Dovetail’s extensive expertise with Amdocs/Clarity solutions and our responsive team makes us a trusted resource to answer your Amdocs/Clarity questions. Our customers agree that working with Dovetail provides clear, concise, responsive, affordable support to help you run your business.
Dovetail CRM is the key to your customer’s success

“Increased visibility of Dovetail in the company has brought forward other business cases for using Dovetail, and as a consequence, the use of Dovetail will be expanded from just the Field Service organization to Customer Support, where it will be used to manage hardware and software escalations. Dovetail’s thin client is so easy to link with other web applications, other groups in the company are asking how they can integrate. They want to add customer lists, case notes, and other Clarify information to their own web pages and reports.”

Kristina Pereyra, Thema-Wave Enterprise Wide Programmer Analyst

With Dovetail CRM, you can take care of customers instead of taking care of your CRM software. The Dovetail CRM suite consists of these web-based applications that are easy to deploy, manage, and use:

**Dovetail Support**, as part of your total CRM solution, leverages innovative web technologies for enterprise-class performance that results in shorter wait times, fast and accurate responses, and happier customers. Dovetail Logistics streamlines your customer order and fulfillment process.

**Dovetail FieldOps** provides the tools you need to manage all support activity when Field Engineers are dispatched to customer locations.

**Dovetail DepotRepair** is an optional Logistics component designed to track parts through the repair process.

**Dovetail Quality** provides workflow and management facilities to track product change requests, requirements, or improvements.

**Dovetail SelfService** empowers your customers to resolve problems, manage products, and seek sales support online. This application eliminates a large number of basic requests traditionally routed to your Customer Care staff, which will ultimately result in substantial cost savings.

**Dovetail WebServices** enables fast, straightforward integration with external systems or web applications.

**Dovetail Admin** is a family of powerful applications designed to replace all Clarify administrative tools.

Dovetail CRM Dashboard
Dovetail Software has a solid reputation for providing powerful, cost-effective products and exceptional customer support.

Dovetail Software specializes in enhancing and replacing Amdocs/Clarity deployments. Our engineers are the heart of our culture, and they are the core advantage we bring to our customers. We focus on delivering the best CRM solutions tailored to your business needs.

The support we give to our customers is total, and we have a clear development path that legacy Amdocs/Clarity owners can count on. Our products are rock-solid reliable, very flexible, and built around user needs. Contact us today to find out how we can help you to enjoy increased profitability, improved customer satisfaction, industrial-strength reliability, and significant cost savings.

Dovetail Satisfied Customers:

www.dovetailsoftware.com
APPENDIX C: DOVETAIL ADMIN ORGANIZATION COMPARISON

DOVETAIL ADMIN TOPICS BEFORE REORGANIZATION

WHAT'S NEW
   Before You Upgrade to a New Version
   What's New in Version 2.3
   What's New in Version 2.2
   What's New in Version 2.1
   What's New in Version 2.0

USER GUIDE
   Overview
   First Choice Platforms

USING FCADMIN
   Login Form
   Manage Contacts Page
   Manage Privilege Classes Page
   Manage Strings Page
   Manage Users Page
   Manage Status Codes
   Manage Application Lists
   Manage Web Commands Page
   Manage Restriction Groups
   Manage Restriction Tables
   Manage Restriction Clauses
   Manage Web Forms
   Manage Web Form Versions
   Manage Resource Configurations
   Manage Queues
   Manage Configuration Items
   Manage Autonumbering Schemes
   Manage Stored Procedures
   Manage Colors Page
   Manage Currencies
   Manage Countries
   Manage States
   Manage Time Zones
   Manage Quality Transitions
   Manage Logistics Transitions
   Manage ZipCodes
Manage fcQuery Objects
The Path Editor
Manage Flash Tables
Manage Business Rule Properties
Manage Business Rules
Manage User Defined Popup Lists
Manage Part Catalogs
Requirements
Packaging
Installation Tree
Manual Installation
Limitations
Performance
DOVETAIL ADMIN TOPICS AFTER REORGANIZATION

INTRODUCTION
    Getting Help

UPGRADE GUIDE AND NEW FEATURES
    Upgrade Dovetail Admin
    Version 2.5
    Version 2.4
    Version 2.3
    Version 2.2
    Version 2.1
    Version 2.0

INSTALLATION GUIDE
    Before You Begin
        Requirements
        Installation Tree
        Limitations
        Performance
    Install Dovetail Admin
        Step 1 - Install Dovetail SDK
        Step 2 - Compile the Clarify Schema Changes
        Step 3 - Import the Data File
        Step 4 - Create a Virtual Directory
        Step 5 - Modify the fc.env File
        Step 6 - Test the Connection
        Step 7 - Grant Permissions

MAIN ADMIN PAGE
    Update Metadata Cache Files
    Update Schema Cache Files
    End a User’s Session
USERS & PRIVILEGES
Users/Employees
Web Users
Privilege Classes
Web Commands

LISTS
Status Codes
Application Lists
User Defined Lists

BUSINESS RULES
Business Rules
Business Rule Properties

CONFIGURATION
Configuration Items
Numbering Schemes
Queues

PARTS
Part Numbers
  Details
  Revisions
  Bill of Materials
  Alternate Parts
  Catalog Inclusions
Part Domains
Part Classes
Part Catalogs

DOVETAIL FEATURES
Colors
fcQuery Objects
Flash Tables
Stored Procedures
Strings
WEB FORMS
   Web Forms
   Web Form Versions
   Resource Configurations

TRANSITIONS
   Quality Transitions
   Logistics Transitions

DATA RESTRICTION
   Groups
   Clauses
   Tables

GEOGRAPHY
   Countries
   States
   Time Zones
   Zip Codes
   Currencies

PATH EDITOR

INDEX
APPENDIX D: SAMPLES FROM THE NEW PARTS SECTION OF THE DOVETAIL ADMIN USER GUIDE

Note: The final Dovetail Admin user guide was delivered as HTML and CHM files. The documentation has changed since the conclusion of my internship, but the latest documentation is publicly viewable here: http://dovetailsoftware.com/resources/docs.aspx?product=DovetailAdmin

To improve the readability of the new parts sections for this report, I chose to present it as it appears in a Word document.
## PART NUMBERS

### DETAILS

#### VIEW A PART NUMBER

1) Select **Part Numbers**.

2) To view existing part numbers, enter at least one search filter (one or more characters) under the **List part numbers whose section**.

3) To view the part number information, select the part number in the list and select **Open**, or double-click the part number row.

#### ADD A PART NUMBER

1) Select **Part Numbers**.

2) From the bottom of the page, select **New**.

3) At the top of the page, enter the new part number data.

4) On the **Details tab**, enter all applicable information.

5) Select **Add**.

#### UPDATE A PART NUMBER’S DETAILS

1) Select the part number in the **list** (see "View a Part Number" on page 69).

2) On the **Details tab**, enter the updates.

3) Select **Update**.

#### COPY A PART NUMBER

1) Select the part number in the **list** (see "View a Part Number" on page 69).

2) Select **Copy**.
**Note:** Selecting Copy will add a '_rev' to the part number. To change this and other part number details, enter the changes and select Update.

**LIST BOM INCLUSIONS**

1) Select the part number in the **List** (see “View a Part Number” on page 69).
2) On the **Details** tab, select **Where Used**.

**Note:** From here, you can click a row and Open to view or make changes to any BOM part numbers listed. Select the part number in the

---

**REVIEWS**

**VIEW A REVISION**

1) Select **Part Numbers**.
2) To view existing part numbers, enter at least one search filter (one or more characters) under the **List part numbers whose** section.
3) To view the part number information, select the part number in the list and select **Open**, or double-click the part number row.
4) Select the **Revisions** tab.
5) To view the revision details, select the revision in the list.

**ADD A REVISION**

1) Select the part number from the **List** (see “View a Revision” on page 70).
2) On the **Revision** tab, enter the new revision data. Select a revision date with the button.
   - If a revision row is already selected, select **Clear**.
   - The **Replaced by PN/Rev** fields cannot be altered on this screen. These fields are populated once the information is entered for the replacement part number.
3) Select **Add Revision**.

**UPDATE A REVISION**

1) Select the part number from the **List** (see “View a Revision” on page 70).
2) On the **Revision** tab, select the revision in the list.
3) Enter the updates. Select a revision date using the button.
   - The **Replaced by PN/Rev** fields cannot be altered on this screen. These fields are populated once the information is entered for the replacement part number.
4) Select **Update Revision**.
COPY A REVISION

1) Select the part number from the list (see "View a Revision" on page 70).
2) On the Revision tab, select the revision in the list.
3) Select Copy Revision.
   ▶ Selecting Copy Revision will add a '_rev' to the part number. To change this and other part number details, enter the changes and select Update.

BILL OF MATERIALS

VIEW A PART'S BOM

1) Select Part Numbers.
2) To view existing part numbers, enter at least one search filter (one or more characters) under the List part numbers whose section.
3) To view the part number information, select the part number in the list and select Open, or double-click the part number row.
4) Select the Parts BOM tab.
5) To view the BOM details, select the BOM in the list.

ADD BOM TO A PART

1) Select the part number from the list (see "View a Part's BOM" on page 72).
2) On the Parts BOM tab, enter the new BOM part number data.
   ▶ To search for part numbers, select the button. After selecting List and choosing a part number to add, select Use/Done.
3) To save, select Add.

REMOVE BOM FROM A PART

1) Select the part number from the list (see "View a Part's BOM" on page 72).
2) On the Parts BOM tab, select the BOM part.
3) Select Remove.
4) Select OK in the dialog box.

LIST BOM INCLUSIONS

1) Select the part number from the list (see "View a Part's BOM" on page 72).
2) On the Parts BOM tab, select the BOM part.
3) Select Where Used.
**Note:** From here, you can click a row, and Open to view or make changes to any part numbers listed.

### ALTERNATE PARTS

#### VIEW AN ALTERNATE PART

1) Select **Part Numbers**.

2) To view existing part numbers, enter at least one search filter (one or more characters) under the **List part numbers whose** section.

3) To view the part number information, select the part number in the list and select **Open**, or double-click the part number row.

4) Select the **Alternates tab**.

5) To view the alternate part details, select the part in the list.

#### ADD AN ALTERNATE PART

1) Select the part number from the **list** (see "View an Alternate Part" on page 74).

2) On the **Alternates tab**, enter the new alternate part number data.

   - To search for part numbers, select the **button**. After selecting List and choosing a part number to add, select **Use/Done**.

3) To save, select **Add**.

#### REMOVE AN ALTERNATE PART

1) Select the part number from the **list** (see "View an Alternate Part" on page 74).

2) On the **Alternates tab**, select the alternate part.

3) Select **Remove**.

4) Select **OK** in the dialog box.

#### LIST ALTERNATE BOM INCLUSIONS

1) Select the part number from the **list** (see "View an Alternate Part" on page 74).

2) On the **Alternates tab**, select the alternate part.

3) Click **Where Used**.

**Note:** From here, you can click a row and Open to view or make changes to any part numbers listed.
CATALOG INCLUSIONS

VIEW THE CATALOGS A PART IS IN

1) Select Part Numbers.
2) To view existing part numbers, enter at least one search filter (one or more characters) under the List part numbers whose section.
3) To view the part number information, select the part number in the list and select Open, or double-click the part number row.
4) To view the part's catalog inclusions, select the Catalogs tab.

ADD A PART TO A CATALOG

1) Select the part number from the list (see "View the Catalogs a Part is in" on page 76).
2) On the Alternates tab, select the button to search for catalogs.
3) To view all catalogs, select List.
   
   Note: You may optionally specify filter criteria before selecting List.

4) Select the catalog in the list.
5) Select Use/Done.
6) To save, select Add.

REMOVE A PART FROM A CATALOG

1) Select the part number from the list (see "View the Catalogs a Part is in" on page 76).
2) On the Catalogs tab, select the catalog
3) Select Remove.
4) Select OK in the Window's dialog box.

PART DOMAINS

ADD A PART DOMAIN

1) In the Part Domain Details section, enter the new part domain data.
2) To save, select Add.

UPDATE A PART DOMAIN

1) Select the part domain from the top list.
2) In the Part Domain Details section, enter the updates.
3) Select **Update**.

**DELETE A PART DOMAIN**

1) Select the part domain from the top list.
2) In the **Part Domain Details section**, select **Delete**.

**PART CLASSES**

**VIEW A PART CLASS**

1) To view existing part classes, select **List**.
   
   **Note**: You may optionally specify filter criteria before selecting List.

2) To view the part class details, select the part class in the list.

**ADD A PART CLASS**

1) In the **Part Class Details section**, enter the new part class name. Adding a description is optional.
2) To save, select **Add**.

**UPDATE A PART CLASS**

1) Select the part class from the **list** (see "View a Part Class" on page 79).
2) In the **Part Class Details section**, enter the updates.
3) To save, select **Update**.

**DELETE A PART CLASS**

1) Select the part class from the **list** (see "View a Part Class" on page 79).
2) In the **Part Class Details section**, select **Delete**.

**PART CATALOGS**

**VIEW A CATALOG**

1) To view existing catalogs, select **List**.
   
   **Note**: You may optionally specify filter criteria before selecting List.

2) To view the catalog details, select the catalog in the list.
ADD A CATALOG

1) In the Part Catalog Details section, enter the new catalog data.
2) To save, select Add.

UPDATE A CATALOG

1) Select a catalog from the top list (see "View a Catalog" on page 81).
2) In the Part Catalog Details section, enter the updates.
3) Select Update.

DELETE A CATALOG

1) Select a catalog from the top list (see "View a Catalog" on page 81).
2) In the Part Catalog Details section, select Delete.
   
   Note: You can only delete inactive catalogs.

ADD A PART NUMBER TO A CATALOG

1) Select a catalog from the top list (see "View a Catalog" on page 81).
2) In the Part Catalog Details section, select Include PN.
3) To view part numbers, select List.
   
   Note: You may optionally specify filter criteria before selecting List.
4) Select Use/Done.
APPENDIX E: MY BLOG POST ABOUT PERSONAS


Don't lose your users in abstraction

In our agile environment, we write user stories to define the features and requirements of the product – specifically, these one to two sentence cards briefly describe how a user can use it.

I’m the new tech writer at Dovetail, and I sit right next to the user story board in the programmer’s bullpen. When I look at the story cards, I see statements like, “As a case worker, I can resolve a case to complete the rendering of service to the customer,” and “As a system administrator, I can provision user accounts in order to grant access to the system.” But who is the case worker, and who is the system administrator? How will they really use the system?

We came to the conclusion that these abstract “users” aren’t the complete story. If we get lost in these abstract ideas of our users, we might also lose sight of what their real needs might be. Instead of talking about some abstract entity like the “case worker,” we should talk about a specific person, in a specific situation, with a specific need.

To this end, we developed user personas that will be the basis for the user stories. So, instead of talking about a case worker, we’ll talk about Adam Agent, and we’ll know what his job duties are, who he works for, and who some of his customers are. And maybe we’ll catch some stories that might otherwise have been lost.

And now for some introductions:
The Company - EndoTech
EndoTech is a medical devices company that develops, manufactures, and services orthopedic reconstructive products and implant software systems. The EndoTech Customer Service department supports medical professionals, medical organizations, device recipients, and caregivers through a 24-hour call center and a self-service web application.

The Administrator – Ally Admin
As part of her common administration tasks, Ally Admin works with a team to install, configure, and maintain EndoTech’s internal business applications.

The Agent – Adam Agent
In the EndoTech Customer Service department, Adam Agent assists customers by answering questions, solving problems, and providing accurate and timely information. He interacts with customers both personally and anonymously by phone, email, instant-messenger, snail-mail, and internet self-service cases.

The Customer – Dr. Ortho
Dr. Ortho is an orthopedic surgeon who specializes in joint replacement procedures. He relies on EndoTech for detailed product information, academic support, and orthopedic resources.
The Customer – Patricia Patient

Patricia Patient’s hip was recently replaced with an EndoTech device, and she likes to stay abreast of any new developments. She also enjoys sharing her experiences with other orthopedic patients and candidates.
REFERENCES


