Publish Paper Manuals No More!: An Introduction to Creating HTML Documentation

Michael Davis, Bassett Consulting Services, Inc., North Haven, Connecticut
Charles Patridge, The Hartford, Hartford, Connecticut

Abstract

So are you still printing paper documentation for your SAS applications while everyone else is surfing the web? Well you don't need to know how to brew a JAVA application to publish user manuals that can be read using any common web browser. Further, HTML (Hyper Text Mark-up Language) allows authors to easily incorporate color pictures of application screens, something that can be costly when documentation is delivered via print. Last, HTML allows authors to create manuals that allow readers to jump to the relevant section at the click of a mouse button. This presentation will cover the following topics:

- introducing HTML and electronic publishing
- contrasting HTML documents with paper publishing
- software that makes it easier to create HTML documents
- how to capture and convert screens into GIF and JPEG files
- design tips for effective HTML documentation
- distributing documentation by disk, file servers, and web servers

While the presenters cannot guarantee that users will read HTML documentation more carefully than they read the paper version, chances are that using HTML will save money. Further, it will be harder for users to say, "I don't know where I put the manual."

Introduction

One of our modern ironies is that while most documents, including this paper, are prepared electronically, they are usually transmitted and read in printed paper format. This paper's authors would not dispute the proposition that it is more convenient to read a paper document, especially while traveling. However, publishing in paper format has certain inherent disadvantages that we seek to avoid.

- cost of reproduction
- cost of distribution
- delays in distribution
- purging obsolete versions
- disposal of obsolete versions

Depending upon the means used, the cost of generating a black and white printed page ranges from about one cent per page to several cents. A color page can easily cost as much as a dollar to reproduce. Multiplied by the many pages often contained in a software manual, reproducing a manual in paper format can be a very expensive proposition!

Paper manuals developed for intra-organization use can often be distributed by hand or intra-organization mail at little additional cost. However, if the manual must be deliveried by the postal carrier or document delivery service, the cost for distributing each copy can easily be several dollars.

In the best circumstances, manuals distributed by a postal carrier or document delivery service can be transmitted to their recipients overnight. Under other circumstances, receipt can take place several days later. If a software user needs the information contained in the manual immediately, the notion that the manual is in the mail is a cold comfort.

Perhaps the most insidious cost of paper documentation is that even after a developer has changed an application and updated the documentation, the users may still rely on obsolete paper copies of the manuals. If only they could be assured that their copy of the manual were the current version!

Last when a paper manual becomes obsolete, it must be discarded. All too often, inventories of virgin manuals, representing many dollars of printing and distribution costs, must be discarded along with copies of the manual put into use. While paper is easily recycled, plastic binders, spines, and tables may need to be discarded as trash.

Contrast the distribution of paper manuals with electronic versions. Reproduction by disk or CD-ROM is much more economical except for the briefest of manuals. Distribution requires no more than a single postage stamp and disk mailer.

When immediate distribution is required, electronic formats can be quickly distributed by e-mail, web servers, FTP, or file servers. Obsolete versions can be sent to the "bit bucket" with no impact on the neighborhood landfill.

In addition to distributing manuals electronically via HTML documents, several other formats are used as vehicles. They include:

- word processor files
- Postscript files
- Adobe Acrobat (PDF) files

Word processing files can be a good distribution format and the authors do use them to create and forward documentation. However, some users may not have access to the version of word processing software used by the author to create the manual document, or be able to read the document file.

The Postscript document format can be characterized as universal format. However there are different flavors of Postscript, which can cause problems for the intended readers. Also, while there are some programs in the public domain that translate Postscript files into a format that can be viewed or printed, many potential readers have difficulty handling Postscript files.

Adobe Acrobat (PDF) files are a good distribution format in that reader software for several computer platforms is available from Adobe at no cost. SUGI and regional SAS conference proceedings are now distributed in Acrobat format on CD-ROM. Also, Acrobat may allow greater control over the appearance of documents than HTML.

However, many potential readers may not have obtained the the Acrobat reader. By contrast, most computers acquired recently probably have a web browser. Also, one must purchase a license for Acrobat Distiller to create manuals in PDF format.

For these reasons, the authors have concluded that at this moment, the most universally accessible format for electronic documentation are HTML files. A brief description of HTML follows.

What is HTML?

Because of the interest in the Internet and the World Wide Web, HTML (Hyper Text Mark-up Language) has enjoyed recent widespread interest. If there is a drawback to HTML's new popularity, it is that some erroneously believe that an Internet (TCP/IP) connection is required to put HTML to work. This is not true!

HTML is a way of taking plain text files with control information (tags) to:

- create links within and between other documents
- mark-up existing text with tags to describe how a browser should format it and behave
- hand non-text files, such as graphic images, to helper applications for processing.

While a connection to the Internet is not required to put HTML to work, access to a browser, such as Microsoft Internet Explorer or Netscape Navigator is required. For the purposes of this paper, it does not make much difference which browser you and your documentation readers care to use.

However, for demonstration purposes, we are going to assume that the reader of this paper has access to a personal computer with MS Windows 95 and MS Internet Explorer 4.0. IE4.0 is available for free from Microsoft and comes bundled with MS FrontPage Express, which we will use to handle some formatting tasks.

While there are many exciting "multimedia" features to HTML, such as the ability to embed sound and video, the authors are going to focus on how to tap the HTML features required to create software documentation such:

- turning existing text into HTML
- editing HTML documents
- inserting lines and symbols
- embedding screen images
- creating internal document links
- linking to other files
- creating a table of contents

This paper avoids coverage of some of more exotic HTML topics that are usually used only to jazz up a web site. With this restriction of scope, the techniques that are essential to preparing HTML documentation can be covered within a short paper such as this one. Let's begin by seeing how we might convert some existing text to HTML.

Creating HTML

The "simplest" way to create HTML is to open a text file into the Notepad application and type in the necessary tags. For example, we might have a piece of text such as:

"This is my first HTML document."

After we insert the necessary tags, the complete HTML document might look like:

```
<HTML><HEAD>
<TITLE>This is my first HTML document</TITLE>
</HEAD><BODY>
<P>This is my first HTML document.</P>
</BODY></HTML>
```

Mark-up tags used in HTML consist of a starting tag enclosed by angled brackets (<>) and in most cases, a closing tag. The closing tag matches the opening tag except that it is prefixed by a forward slash (/). Some HTML tags, such the line break <BR> tag, do not require a closing tag.

The <HTML> tag at the beginning of the document and the ending </HTML> tag at the end identifies the text between them as a HTML document. The <HEAD> and </HEAD> tags block out the document's head section.

Within the head section is the document title, marked by the <TITLE> and </TITLE> tags. The title is the information displayed in the banner section at the top of the browser window.

The content of the document is contained in the body section, marked by the <BODY> and </BODY> tags. The paragraph tags, <P> and </P> break the text into sections. If no extra spacing between lines is desired, use the <BR> tag to break line within a paragraph.

Use of the HTML, HEAD, TITLE, and BODY tags is highly recommended for all HTML documents.

While some folks create HTML using text editors, the availability of HTML formatters is making the exercise of marking up text by hand to be an exercise for masochists. Some of the latest available word processors, such as MS Word 97 have the ability to save a document in HTML already built-in.

Older word processors can gain this ability through the installation of a "plug-in". A plug-in is a additional piece of software that adds features to another application, such as a word processor. In some cases, the plug-in can be downloaded at no cost from the vendor's web site. In other cases, the vendor or a third-party offers the plug-in at additional cost.

SAS programs can be handled by the same software used to convert text into HTML. To achieve a better appearance for a block of program code, consider removing the paragraph tags, <P> and </P> separating each line of code. Instead, enclose program code as "pre-formatted" text between <PRE> and </PRE> tags as shown in the following example.

```
<PRE>
proc print data=mydata ;
   var a b c d ;
run ;
</PRE>
```

The following SAS program can be used to save output from a report to an HTML file

```
proc printto file="sashtml.htm" new; run;

/* put HTML tags at beginning of file */
data _null_;
  file print noprint notitles;
  put  " <HTML>" / " <B>" / " <PRE>" ;
run;

/***** place your SAS reports here ****/

/*   put closing tags at end of file  */
data _null_;
  file print noprint notitles;
  put / " </PRE>" / " </B>" / " </HTML>" ;
run;

proc printto;
run;
```

To help in the preparation of HTML documents, SAS Institute provides some macro programs to convert SAS data sets and output into HTML. The macro programs can be found and downloaded from the URL:

http://www.sas.com/rnd/web/format/index.html

Similarly, SAS Institute also provides drivers that can be used within SAS/GRAPH to create GIF and JPEG graphic images files to illustrate HMTL documents. These drivers can be found and downloaded from the following URLs:

http://www.sas.com/rnd/web/driver/GIF/Gif.html
http://www.sas.com/rnd/web/driver/JPEG/Jpeg.html

While HTML formatters can save large amounts of time when preparing HTML documentation, even the best formatters will probably require modification of some HTML code by hand.

Web-Page Editors

HTML documents can be edited in Notepad. In fact, when one selects Source from the View pull-down menu in Internet Explorer, the HTML is shown using Notepad. Some experienced developers, such as one of this paper's authors, prefer to create HTML documents using ordinary text editors such as Notepad.

If the source HTML file is larger than can be accomodated by Notepad, it can be edited using Wordpad or the SAS Program Editor. Better still, a large HTML file can be split into smaller, linked files, which will be described in a subsequent section.

Regardless whether one uses Notepad, WordPad, or another text editor or word processor, chances are that the editor cannot automatically supply HTML tags. Some individuals may find it easier to use a web-page editor that can supply the needed tags based upon menu selections. Web-page editors usually show the document under construction in a WYSIWYG ("what you see is what you get") mode, which usually makes editing and layout easier and faster.

Web-page editors often have additional features that make them easier to use for modifying HTML documents than a standard editor. Some editors show the HTML tags in color so they stand out from the content portion of the document. They often include toolbars customized so that common tasks, such as inserting an image or formatting text can be accomplished with a click of the mouse.

Of course, web-page editors can be purchased or licensed from a variety of software vendors. The *PC Magazine* article cited in the Reference section of this paper compares currently available web-page editors. Please note that most of these editors include features that are not necessary for the simpler task of converting software documentation to HTML.

However, there are at least two ways to acquire to web-page editors with the essential features at little or no additional cost. Books on HTML, such as the ones listed in the References section of this paper, often come with a CD-ROM. The CD-ROM will probably include a popular web-page editor.

Also, popular web browsers often come bundled with additional tools. One of the tools often included is a web-page editor. In the case of MS Internet Explorer 4.0, the bundled web-page editor is FrontPage Express, which is a scaled-down version of Microsoft's well-received web-page editor, FrontPage.

MS Internet Explorer 4.0 and Front Page Express can be obtained from several sources. Sometimes it is supplied when you purchase a computer or a copy of Windows 95. It can be purchased separately from Microsoft or a software dealer on CD-ROM.

MS Internet Explorer can be downloaded from the Microsoft web site although it will probably take several hours to complete the task. If you have an ISP (Internet Service Provider), they may have or be able to supply you with Internet Explorer and FrontPage Express, or a comparable web-page editor.

Last, because of the Internet's tradition of free exhange, good web page editors are available for free or very low cost. The *PC Magazine* article cited in the Reference section of this paper lists such editors as:

- AOLpress 2.0
- HTMLtool 1.6
- JMK HTML Author 4.01
- Arachnophilia
- Hippie 97
- WebExpress 3.0

For simplicity of explanation, the authors will assume that the reader has access to FrontPage Express or a web-page editor with similar capabilities.

Editing an HTML Document

Now the reader should start FrontPage or another web-page editor and open an existing HTML document. That HTML can be created by a word processor or one of the SAS formatter macros. Our next step is to reformat and edit the HTML document so that it has the appearance we seek.

The most common editing step is to add or remove paragraph breaks and any extra lines between paragraphs. Within a web page editor, adding a paragraph break is done by hitting the Enter key.

Removing a break is done by hitting the Delete key at the end of the last line of the paragraph or the Backspace key at the left margin of the blank line to be deleted. Adding a paragraph break inserts the </P> and <P> tags. Removing a paragraph break deletes those tags.

If you experiment with your web-editor, you may notice two features of HTML. First, your tags may be in lower case. For example, the paragraph tag may appear as <p>. The tags in HTML are not case-sensitive.

Second, regardless of how many spaces are inserted between words or between sentences, only one space will be shown when the HTML document is displayed on a browser. You can take advantage of this feature to make your HTML code more readable by breaking lines and adding blank lines to split documents into sections.

This hints at a major difference between writing HTML documents and SAS code. When a SAS program is written using syntax that is not quite correct, the usual result is a error message or warning on the SAS Log.

When an HTML document contains tags that are unknown to the browser, it ignors those tags. The only certain way that we can verify that our document has the right tags is to view it in our target web browser(s).

Ironically, HTML and SAS share one syntax feature. When the RUN statement is omitted, the following DATA or PROC statement serves as a step boundary. Similarly, if the closing paragraph </P> tag, is omitted, the next paragraph tag closes the previous paragraph block.

The next most common HTML formatting task is to change paragraph and text styles. For example, there are six different heading styles, controlled by the <H1> through <H6> heading tags. Experiment with these tags by viewing the following HMTL code in your web browser.

```
<H1>Heading Style 1</H1>
<H2>Heading Style 2</H2>
<H3>Heading Style 3</H3>
<H4>Heading Style 4</H4>
<H5>Heading Style 5</H5>
<H6>Heading Style 6</H6>
```

Use of heading tags eliminates the need to add paragraph tags.

Controlling text alignment (justification) is a common HTML layout task. Alignment can be changed from the default of left alignment by inserting the align argument in the preceding paragraph tag.

For example

```
<P ALIGN="center">
```

will center the text within the browser window.

Another common task is to change the text style. Text placed between the <STRONG> and </STRONG> tags is shown in bold type. The <EM> and </EM> tags are used to emphasize text by italicizing it.

The <CODE> and </CODE> tags show the text between them in fixed a fixed type face. It is useful for showing program code although the <PRE> and </PRE> tags often give a neater appearance. The <CITE> and </CITE> tags indent the text between them.

One feature common to the text formatting tags that have been described is that they are all logical typeface tags. Logical typeface tags are interpreted by the web browser according to the host computer's display capabilities.

Alternatively, physical attributes can be specified in the font <FONT> tag. For example, if we wanted to specify that some selected text should use the Arial typeface and be 10 point size, the web-page editor would supply the following tags:

<FONT size="2" face="Arial">
selected text </FONT>

The web-page editor translated our font size request to the HTML size=" 2", which approximates 10 point size. The face="Arial" parameter tells the web browser to substitute the Arial font for the Times New Roman font that it would have otherwise used.

Since the desktop computer has the Arial font and our web browser supports the font= parameter in the font tag, our text formatting request is executed as we requested.

However, if the computer viewing our HTML document did not have the requested font, or if an older web browser was being used to view the document page, our font instructions might be igored.

Simlarly, we can specify color as an attribute in the FONT tag. Colors may be specified by name (e.g., <FONT color="red">) or as a six character number (e.g., <FONT color="#FF0000">).

However, colors have this nasty tendency to look different when viewed on different browsers, monitors, and printers. As an experiment, try reproducing the colors red or brown and watch the variations.

One reason for this is that cathode-ray tubes (CRTs) are not able to display primary colors (red, green, and blue) as saturated as we see them in real life. To compensate, computer manufacturers usually show colors darker than their true color values.

Our advice to help insure that documents appear as designed is to:

- use logical text formatting tags instead of physical tags or attributes

avoid using HTML tags that may not be supported by all web browsers

- avoid tags not compatible with HTML 2.0
- test-drive your draft HTML documents on different browsers, especially older ones, to identify problems with layout, color, and unsupported HTML components

Inserting Lines and Special Characters

Sometimes, it is necessary to draw a horizontal line for emphasis. To insert a horizontal line, use the <HR> tag (horizontal rule). Note that the <HR> tag does *not* require a closing tag. The line drawn ends at the right side of the web browser window.

HTML is based upon 7 bit ASCII characters. However, to meet the requirements of copyright lawyers and others, it is sometimes necessary to insert special characters in an HTML document. Web-page editors can greatly ease this task.

For example, we might need to insert the registered trademark symbol while writing a paper about a particular brand of software. We could tell the browser to show the "R" within a circle by using the HTML expression, &reg, followed by a semi-colon.

The ampersand (&) tells the browser that we are specifying a special character. "reg" is the special character for registered trademark. The semi-colon is the delimiter that marks the end of the expression calling for the insertion of the special character.

A sample of other special characters that are available includes:

¬	&not	™	&trade
¶	&para	©	&copy
<	&lt	>	&gt

Inserting Tables

Some types of information is best represented in tabular format. The HTML syntax to display information in a table is straight-forward.

The start of the table, along with any additional attributes is specified by the <TABLE> tag. The </TABLE> tag closes off the table definition.

Tables are defined row by row, in descending order. Within each row, cells are defined, left to right. The following example illustrates how tables are built:

```
<TABLE border="1">
  <TR>
    <TD>&nbsp;Cell 1</TD>
    <TD>&nbsp;Cell 2</TD>
  </TR>
  <TR>
    <TD>&nbsp;Cell 3</TD>
    <TD>&nbsp;Cell 4</TD>
  </TR>
</TABLE>
```

The start of the table is denoted by the Table <TABLE> tag. Within the Table tag, the border="1" attribute specifies a narrow line around cell.

Each row's description starts with the Table Row <TR> tag and ends with the </TR> tag. Each cell's description starts with the <TD> tag and ends with the </TD> tag. The &nbsp pneumonic inserts a non-breaking space between the left border of each cell and text in the cell.

The table that the preceding HTML code generates looks something like this:

Cell 1	Cell 2
Cell 3	Cell 4

The syntax possibilities for tables can be rather complex and confusing, not unlike the SAS TABULATE procedure. However, tables can overcome some of the formatting problems encountered when creating HTML documents.

Capturing Images

Only two types of graphic files, GIF and JPEG can be embedded into HTML documents as "inline" images. The first question that may arise when capturing an image to be inserted into a HTML document is, "Should the image be captured in a JPEG or GIF file?".

JPEG files often (but not always!) take less storage space and downloading time than GIF files. However, JPEG files take longer for browsers to uncompress than GIF files.

JPEG files are capable of reproducing more colors than GIF files (256 verses 16.7 million). However, for all anyone knows, the computer on which the reader's browser is running may not be able to display more than 256 colors.

The best advice that the authors can give is that JPEGs tend to render photographs better than GIFs. For general-purpose illustrations such as line-drawings, GIFs are a better choice.

The choice between JPEG and GIF files becomes academic if you are not able to convert the screen illustration that you captured on the Windows clipboard into one of those formats. The good and bad news is that commercial software is available to capture screen images and to convert other graphic fixes to GIF and JPEG format.

One package that one of the authors has used to convert files to GIF and JPEG format with good success is HiJack Pro from Quarterdeck Software. HiJack Pro comes bundled with HiJack Capture, which competently handles screen capture tasks.

Inserting Inline Images

Some converters that translate word processor files to HTML do not translate the links to GIF and JPEG image files. Inserting links for inline images is one of the tasks that web-page editors handle easily.

To insert an image when using FrontPage Express, position the cursor to the location where the image is to appear. Select Insert,and then Image. Pick the GIF or JPEG file to be inserted and click on the Open button. The image now appears in the document courtesy of the Image <IMG> tag which looks something like:

```
<IMG src="image.gif">
```

The preceding Image tag instructs the browser to look in the same directory as the document for the image file named *image.gif*. If desired, use the align attribute (e.g., align=center) to position and the height and width attributes (e.g., height="50" width="100") to size how the image is displayed.

Inserting Internal Links

The preceding topics covered by this paper relate to text layout activities that parallel paper document formatting. However, one of the strengths of HTML is the ability to move beyond a purely sequential representation of ideas and jump to different topics as the subject and reader may require.

The ability to jump to another portion of a document is also known as a hyperlink. An internal link repositions the web browser to display another portion of the current document page. Such a link consists of the Anchor tags, <A> and </A>.

First, insert the Anchor tags to mark the destination of the link. For example, to set a link to jump to the middle of a page, we would insert the anchor:

<P><A NAME="middle">*any text goes here*</A></P>

Elsewhere on the page, we would insert another anchor tag that might look like this:

```
<P><A HREF= "#middle">go to
middle</A></P>
```

The second anchor tag would appear in the web browser as the text:

go to middle

Clicking on the text causes the browser to jump to the anchor labeled "middle". Most browsers show links in a contrasting color to highlight their availability.

Inserting Links to Other Pages

During the discussion of text and web-page editors, it was observed that HTML documents can get too large to comfortably edit and download. When this occurs, a good practice is to split a document into more than one file and use hypertext links to move among files.

The Anchor tag is used with the href attribute to set up this type of link. For example, to set up a link to a document named *pagetwo.htm*, the following Anchor tag would be inserted:

```
<P><A HREF= "pagetwo.htm">go to
Page Two</A></P>
```

If it is desired to position the link destination to a portion other than the beginning, a name anchor can be used. Building on an earlier example where a link to the middle of a document was created, the Anchor tag might be similar to the following:

```
<P><A HREF= "pagetwo.htm#middle">go to
Page Two</A></P>
```

Using Lists to Create a Table of Contents

A helpful feature to add to a lengthy or complex HTML document is a table of contents. One way to do this is combine hypertext links with HTML lists.

For a table of contents, use an unordered list. At the beginning of the list, place a <UL> tag and at the end of the list, a </UL>. Each list item (each topic in the table of contents) starts with an <LI> tag and ends with a </LI> tag.

The HTML code needed for a table of contents listing might be similar to the following:

```
<UL>
   <LI><A href="Intro.htm">Introduction</A></LI>
   <LI><A href="Section1.htm">Section 1</A></LI>
   <LI><A href="Section2.htm">Section 2</A></LI>
   <LI><A href="End.htm">End Section 2</A></LI>
</UL>
```

The preceding HTML code would create a table of contents that would appear as the following:

- Introduction
- Section 1
- Section 2
- End

If desired, multi-level table of contents can be represented by using sublists as shown in the following example:

```
<UL>
   <LI><A href="Intro.htm">Introduction</A></LI>
   <LI><A href="Section1.htm">Section 1</A></LI>
        <UL>
        <LI><href="S1PartA.htm">Part A</A></LI>
        <LI><href="S2PartB.htm">Part B</A></LI>
        </UL>
</UL>
```

The resulting multi-level table of contents would look similar to the following:

- Introduction
- Section 1
    - o Part A
    - o Part B

At the end of each linked HTML file, consider adding the following additional navigational links:

- Previous Section
- Next Section
- Return to Table of Contents

As a final touch, consider using the name attribute so that when the user returns to the table of contents, the browser is positioned on the list item link to the previously viewed file. The table of contents list items can be named in sequence, starting with name="TofC1".

By Disk, File Servers, and Web Servers: Distributing HTML Documents

One of the biggests misconceptions about HTML is that one needs a web server to distribute documents and that the reader needs an Internet connection to read them. This is not true!

In many cases, HTML documents and the accompanying GIF and JPEG illustrations will fit on a floppy disk or CD-ROM. Thus HTML documents can be distributed on the same media used to distribute SAS applications.

File Servers are often used to distribute SAS applications and their data. Regardless of the brand of network operating system used, file servers can also be used to distribute web documents.

After establishing a connection to the file server containing the HTML document, start their browser and select Open (or Open File) from the File menu. After entering the path and HTML file name where the URL usually appears, hit the Enter key to view the document.

When should a web server or FTP (File Transfer Protocol) be employed? When the readers cannot easily receive documents on removal media, or when it is not possible to share a file server, a web server reachable by an Internet (TCP/IP) connection may be an attractive alternative.

Conclusion

A visit to a bookstore will reveal shelf after shelf of books on HTML and related subjects in the computer books sections. The authors recommend that those motivated to create HTML documentation consider acquiring one or more books for reference.

However, the amount of knowledge of HTML and related subjects required to create attractive online documents is relatively small. The authors hope that with the encouragement provided by this paper, SAS developers will use HTML documents to instruct users on how to use their applications.

References

Brown, Mark, Jung, John, and Tom Savola (1996) *Special Edition Using HTML, Second Edition,* Indianapolis: Que Corporation

Evans, Tim (1996) *10 Minute Guide to HTML 3.2, Second Edition,* Indianapolis: Que Corporation

Mendelson, Edward (January 20, 1998), "Web Authoring Tools," *PC Magazine,* 17(2), 152-187

Tittel, Ed and James, Stephen N. (1997), *HTML for Dummies, Third Edition,* Foster City: IDG Books Worldwide, Inc.

Acknowledgments

SAS and SAS/GRAPH are registered trademarks of SAS Institute, Cary North Carolina

The authors may be contacted at:

Michael L. Davis
Bassett Consulting Services, Inc.
10 Pleasant Drive
North Haven CT 06473-3712
Telephone: (203) 562-0640
Fax: (203) 498-1414
E-Mail: Bassett.Consulting@worldnet.att.net

Charles Patridge
172 Monce Road
Burlington CT 06013
Telephone: (860) 673-9278
E-Mail: TVJB41A@prodigy.com
Website: http://pages.prodigy.com/SASCONSIG