Web Application Builder for Delphi

The Fastest & Easiest Tool to Build Interactive Web Applications

Version 1.01   March 2, 1997

1 Features

- Easily create interactive web sites
- Leverage Delphi's ease-of-use
    - Use Delphi's Form Editor to create your user interface
    - Create standard Delphi code snippets (e.g., events such as OnCreate, OnShow, OnClick, etc.)
- Leverage Delphi's power
    - Full access to Delphi's database engine (BDE) for client/server based web sites
- Using "Local Mode" you can create and easily test your application in the Delphi IDE – no server is necessary
- Automatically converts your Delphi Forms to HTML, and parses the user's response back into the appropriate Delphi component properties, and executes the appropriate handlers
- Full access to the standard user interface components – use TLabel, TEdit, TButton, TListBox, TMemo, TComboBox, TCheckBox
- New HTML components such as HTML Anchor, Header, Image, Table, HiddenText, BodySettings, etc.
- HTML Image component provides easy hotspot creation and handling
- Templates for standard Delphi Table viewing and editing (i.e., DBGrid)
- ISAPI interface is more efficient than CGI – no need to reload an .exe or reconnect to a database

- Persistence is handled automatically by WABD – keep track of user preferences and session variables
- Broken or incomplete sessions are handled by automatic garbage collection or an OnTimeout event
- Supports multiple simulataneous users
- Server performs all processing, clients are lightweight - no need for ActiveX, VBScript, or Java on the client.
- Supports creation of dynamic controls to change your user interface on-the-fly
- Fully functional – not crippled in any way
- The unregistered version prepends "Unregistered" messages to all HTML output

2 Hardware / Software Requirements

- Delphi 2.01 (with the Delphi Internet OCX's)
- Windows 95/NT
- ISAPI (compatible) Web Server (e.g., MS Personal Web Server or Internet Information Server)
- TCP/IP Network setup (e.g., Internet Connection, Dial-up PPP, etc) – Note that "local" mode does not require any network connection
- Client's browser requires forms and tables capability (e.g., Internet Explorer or Netscape Navigator)

3 Introduction

The Web Application Builder for Delphi (WABD) is the fastest, easiest tool to develop interactive web sites on the Internet or on a company intranet. A programmer familiar with Delphi can produce a simple interactive web application in under five minutes using the tools and components provided by WABD.

This tool leverages the full power and ease of use of Delphi. Forms and code "snippets" are created using the Delphi Form Designer and Editor. The programmer is allowed full access to the Borland Database Engine (BDE) to create powerful client/server web applications.

4 Installation

You must create a directory called C:\WABD directory on your C: drive. The demo programs have hard-coded references to the C:\WABD directory. Unzip the WABD zip file (WABD100.ZIP) using a 32-bit zip program (e.g., WinZip) that handles long-filenames. Be sure to enable the option to retrieve directory information. This should unzip several files in the C:\WABD directory, and create two directories C:\WABD\DEMO and C:\WABD\POOLDEMO.

Open Delphi, and select the Menu Component | Install. At the Install Components dialog, hit "Add" and "Browse". Select Files of Type (*.DCU) instead of (*.PAS). Add the following files (repeating the steps as necessary): HTMLAnchor, HTMLFormSection, HTMLHeader, HTMLHiddenText, HTMLImage, HTMLRaw, HTMLTable, and HTMLBodySettings. After you hit OK Delphi will install the components into a section named "HTML".

5 Running the Demos

Open the project WABDDemoLocal.dpr in the C:\WABD\Demo directory. Run the program to test out the capabilities.

Next open the project C:\WABD\PoolDemo\PoolDemoLocal.dpr. Run the program for a demonstration on the database capabilities of WABD.

6 Using the Local Browser

The Local Browser acts just like a Web browser, click on buttons and links as you normally would. There are a few menu options you can use.

File, Open – You can use this to open an HTML file. This will terminate your current logon session.
File, Refresh – This will begin a new logon session. It is exactly what would happen if a user pressed "Refresh" from a Web browser.
View, View HTML Source – This will display the HTML text for the current page.
Options, Run Multithreaded – This is checked by default. This is the best way to test your application, under the same multithreaded conditions as under a web server. If you find it difficult to debug your application in this manner, you may turn it off.

7 Tutorial – Create your own Interactive Web Application (in under 5 minutes!)

The best way to learn how to use WABD is to create your own Interactive Web Application. Follow these directions exactly.

- Select File, New Application.
- Select File, Save All. Move to C:\WABD and create a directory called "Hello". Enter the directory and save "Unit1". Hit save for "Project1."
- Go to Unit1 and add "HTMLForm" to the Uses clause.
- Change TForm1 to derive from THTMLForm instead of TForm.
- Go to the bottom of Unit1, and above the "end." statement type "initialization", new line, and then "RegisterHTMLForm (TForm1)".
- Compile the project to make sure you have not made any errors thus far.
- Switch to the Form, go to your HTML Component palette, and drop an HTMLHeader on the Form.
- Change the caption to "Hello World!"
- Drop an HTMLFormSection immediately below the header, and change its height to 200.
- Go the Standard Palette page, and drop the following components onto the HTMLFormSection: TLabel, TEdit, TMemo, TButton (two), and TCheckbox.
- Set the caption of one TButton to "Submit" and the other TButton to "Log off"
- Double-click on the "Submit" button and enter the following code: Label1.Caption := Edit1.Text;
- Double-click on the "Log off" butotn and enter the following code: EndSession('Goodbye!');
- Go to Project, Add to Project, go back one directory and add LocalBrowser.pas.
- Go to Project, Options and remove "Form1" from AutoCreateForms.
- Hit Run.
- Enter some text into the Edit control, and press Submit.
- Now press "Log Off"
- You just created your first Web Application in under 5 minutes!

To deploy your application using Personal Web Server perform the following:

- Select File, Close All.
- Select File, New. Then select DLL and press OK.
- Select File, SaveAs, go to your C:\WABD\Hello directory, and save the project as "hello.dpr".
- Add the following to your Uses clause: WABD_Isapi, and Unit1.
- Right above the begin end block, enter the following:
```
exports
   GetExtensionVersion,
   HttpExtensionProc;
```
- Select Project, Compile.
- Go to Explorer, and copy the "Hello.DLL" to your scripts directory for your Web Server. On my setup with Personal Web Server I copy it to C:\WebShare\Scripts\Hello.dll.
- Fire up your web browser, and connect to your web server as appropriate, my URL would look something like this: http://204.157.135.52/scripts/Hello.dll. Note that it is useful to use the program "WinIPcfg" to determine your own IP addres if you use a dial-up Internet connection like I do.
- Enter text into the Edit Box and hit "Submit". You have really created an interactive web application!

8 Deployment on the Web Server

You just learned from the tutorial above how to deploy a simple web application on your web server. For more complicated projects there are a few more details to remember.

You must remember to copy over all necessary graphic files and to place them in the correct directory. Remember that the graphic files should not go in the script directory (at least not using Personal Web Server), but rather somewhere beneath C:\WebShare\WWWroot. Also make sure that your project was compiled to correctly reference the graphic files.

If you are testing your project and recompiling the DLL, make sure to shut down the Web server before copying over the new DLL. You may then restart the Web Server and it will use the new DLL.

9 Shareware Registration

WABD is shareware. You are free to copy and distribute this software in unmodified form. The Shareware concept means you have the right to evaluate WABD for a limited time to see if it suits your needs. If you find the program useful and wish to create and deploy applications with WABD, you must register the software. If WABD is to be included in a commercial or government application, it <u>must</u> be registered (i.e., paid for).

The registered version of WABD offers additional benefits: 1) the ability to hide the "Unregistered" message in the "About" box and in the HTML output; 2) free technical support via email or phone (you pay the long distance); 3) free upgrades for all 1.x versions (bug fixes and enhancements); and 4) the "professional" version provides complete source code and a powerful TRACE statement tool (see About the Trace Unit).

10 Pricing and Ordering Information

Registering WABD is U.S. $79 for the "standard" version and $129 for the "professional" version. This provides you with registered versions of the tools, technical support, and source code (professional version only). Note that support is only provided for the <u>unmodified</u> source code. Distribution is by email (attached .ZIP file in MIME format) or by US mail (additional $5 charge).

Send a check or money order to:

WABD Software
Care of: Ben Ziegler
210 E. Fairfax St #624
Falls Church, VA 22046 (USA)

You may do a direct wire transfer to:

Ben Ziegler, Nations Bank, account #8181-8693

Make checks payable to "Ben Ziegler". Be sure to email me if you do a direct wire transfer. When you register I will send you my phone number for technical support. Note that I can only take calls after work.

<u>Compuserve Registration</u> - GO SWREG account #14744 ($79) for the standard version, and account #14745 ($129) for the professional version. If you need a disk mailed to you, select your home country as outside of the United States to include the $5 charge. Note that even though SWREG may display an older version of WABD, I will send you the most up-to-date version.

11 Additional Information

For the latest information on the Web Application Builder for Delphi (WABD) visit the WABD web site at http://www.radix.net/~bziegler/wabd/wabd.htm. Send your questions, comments, and suggestions to Ben Ziegler at bziegler@radix.net. To make sure I don't miss your email, please put "WABD:" in all caps in your subject line.

12 Acknowledgements

Thanks to Charlie Calvert who's primer on ISAPI programming got me started on this project.
[Your name goes here for bug reports and fixes! ☺]

This concludes the general information. The following sections contain all of the technical details.

13 WABD Objects

This section lists all of the WABD defined objects, its description, properties, methods, and events.

13.1 THTMLForm

All Forms you create in a WABD application must be derived from THTMLForm. Simply create a Form with Delphi, and then change its declaration to = class(THTMLForm).

`property BaseGlobals: TCachedObject;`
BaseGlobals provides a pointer to your Globals object (if you defined and registered a GlobalCreator function). You must cast the BaseGlobals to your Global object type. See the pool demo for examples on how to use and access the global variables.

`procedure EndSession(Msg: string);`
This is similar to Delphi's Application.Terminate method. You can call this method from within any of your Event Handlers. It will terminate the session, free your global variable object, and log the user off. In Local mode it will exit the application. This is the proper way for you to log users off, after they have completed their session.

`procedure ShowFormType(fc: THTMLFormClass);`
This is a very important method in WABD. This is the only method you can use to "Show" new forms. Not that it takes a FormClass, not a Form Object. WABD checks to see if you called this method in your handler, and if so creates the new form and sends it's HTML output to the user. It is common to set a Global variable to a value that the new form can initialize with in it's OnCreate event. For example, your logon form might have code like this:

```
procedure TLogonForm.OnLogonButtonClick(Sender:TObject);
begin
      Globals.UserName := Edit1.Text;
      ShowFormType(TUserMainForm);
end;
```

Designing HTMLForms
You must follow certain rules when placing controls on an HTMLForm. There are only six components you can drop *directly* on an HTMLForm: THTMLHeader, THTMLFormSection, THTMLRaw, THTMLTable, THTMLBodySettings (and an invisible THTMLImage – see THTMLBodySettings). All other controls must be placed in an HTMLFormSection.

All of these controls should have their Align property set to alTop. The HTMLForm will be converted to HTML by processing the controls from top to bottom.

13.2 TCachedObject

TCachedObject is used to create objects that WABD knows about. WABD can delete them if they are not accessed after a certain timeout period. The Globals object you create must derive from TCachedObject.

`property OnTimeOut    : TNotifyEvent;`
This event is called if the object is about to be Destroyed by the automatic garbage collector. You can assign a handler in your Global Object's constructor to do some special processing if there is a TimeOut.

`var   CacheTimeOutSeconds : integer;`
This is a global variable that is used to determine when objects should be deleted by the garbage collector thread. You can set this to any value (an extremely high value will essentially disable the garbage collector, $7FFFFFFF = 472 years!). The default value is (30 * 60) to time-out in 30 minutes.

Deriving from TCachedObject
Your globals object will derive from TCachedObject. This object will be accessed from different threads (each time the user submits the form, a different thread in the Web Server could handle the request). You should not add any member variables that refer to non-thread safe VCL objects.

The only VCL objects that are thread safe are found in the Classes unit (e.g., TList, TStringList, etc), or the non-visual database components (TDataModule, TSession, TTable). If you create a TDataModule, be sure to add a TSession component and assign it a unique name in the global variables constructor, and to have each TTable reference that TSession. See the PoolDemo for an example of this. Using a TSession object will allow any TTables on the DataModule to be thread-safe.

13.3 THTMLFormSection

This is a similar to a TPanel component. All other controls will be placed on this control. You may place multiple HTMLFormSections on your Form if you wish to place HTMLTables between them, etc.

`property GridX: integer;`
`property GridY: integer;`
These properties deal with the grid alignment of the HTMLFormSection. All controls on the HTMLFormSection will be rounded-off to the nearest grid alignment when converted to HTML. To ensure your Form has no overlapping controls, make sure that all controls are aligned to these values (see the AlignNow property). It is *very* important that dynamically created controls also be set to align on these values, or you might get overlapping controls which will not convert to HTML.

`property AlignNow: TWABDAlignSection;`
This is not a property that you can modify; however, at design time if you double click this property in the Object Inspector, all controls on the THTMLFormSection will be automatically aligned based on the GridX and GridY values.

Notes on GridX and GridY values
The GridX and GridY properties of the THTMLFormSection determine the grid alignment. If you use too small a number, a table with a large number of cells will be used to create the HTML (which will take more bandwidth). If you use too large a number, the controls will be too far apart, etc. Experiment with good values for your form layout.

13.4 THTMLAnchor

THTMLAnchor's are the "jump links" in HTML. You could place a link to other web sites, or other pages on your web site. Note that when the user selects an anchor, they will be essentially logged off from your application, so they should probably only be placed on the last screen, or on a Logon screen before they logon, etc.

`property Destination: string;`
This is the Destination for the "jump link". You can place an absolute or relative URL here. Example: http://www.radix.net/~bziegler/wabd/wabd.htm is an absolute URL, while /Department5/Project1/Intro.htm is a relative URL for your site.

`property Caption;`
This is the text that will appear to the user. This should be an appropriate explanation for the destination URL. For example, you could enter WABD Home Page, and set the destination appropriately.

13.5 THTMLBodySettings

This is a special component that can be placed on an HTMLForm. You should only place **one** on a form. This sets the properties for the HTML <BODY> section.

This component is necessary because certain HTMLForm properties could not be exposed on the Object Inspector due to limitations in Delphi's Form Inheritance.

```
property     TextColor: TColor;
property     LinkColor: TColor;
property     VLinkColor: TColor;
```
These refer to the Text color, Link color, and Visisted Link color. Their default value is "clDefault" which means to let the client's browser decide the colors. If you are using a certain background image, you may need to set these colors for proper contrast. If you double click on these properties, the Color Dialog will appear. Note that you should select a color using the Color Dialog, and **not** using the enumerated colors (e.g., clWindow, etc) which have no meaning on the client machine.

`property     BgrdImage: THTMLImage;`
This property links to an HTMLImage on the form. Note that the HTMLImage can be placed on the Form itself (not in an HTMLFormSection), and the Image will not be converted as a control. It is best to use low-contrast images for background images so that text is easily readable. For more information, see THTMLImage.

13.6 THTMLHeader

HTMLHeader creates an HTML Header which is an emphasized line of text, usually used to break up sections of the HTML form.

`property HeaderNum: integer;`
This can be a number from 1 through 6. A value of 1 is the largest header, while 6 is the smallest.

`property Caption;`
This is the text for the header.

13.7 THTMLHiddenText

This is a useful component to place some "hidden" text in the HTML form that will be returned when the user submits the form. For simple applications, global variables are not necessary and all state information can be passed in Hidden Text components. Note that the HTMLHiddenText component can be dropped anywhere on the form (since they are invisible their position is not important).

WABD uses special HiddenText components to save vital state information, such as the logon session, the global variable object for that session, and the Form's name.

`property     Value: string;`
This can be any string you wish to save with the form.

13.8 THTMLImage

You can place an HTMLImage on a HTMLFormSection (or on an HTMLForm to be used as a background image).

`property IsMap: boolean;`
Set this property to True if you want the user to be able to click on the image. If it is set to false, it is just a "pretty picture" with no user interaction.

`property AltText:string;`
This is the "Alternate text" that a browser can display instead of the image. This is nice to give users some context even if they can't support graphics.

`property OnMouseDown:THTMLMouseDownEvent(Sender: TObject; X, Y: integer);`
This event is called if the user clicks on the image (and the click did not occur on any TPaintBox "hot spots").

`property ImagePath: string;`
The filename of the image to use, e.g., "/wabd/wabd.gif". It is the path where a remote browser would "see" it. This field is required.

`property LocalPath: string;`
The root path to prepend to the ImagePath in local mode (or design time). You can set this to the root directory of yor Delphi project. It is actually best to set it to the root directory for your Web Server, so that you only need 1 copy of your image files. Personal Web Server uses "C:\WebShare\WWWroot". The physical location of the image file should be LocalPath+ImagePath.

Image Path & Local Path Notes
The image to display is determined by the ImagePath property (and in local mode, also by the LocalPath property). If you set the ImagePath to "\dinosaur.gif", then that .gif must exist in your sites root directory. You could also set it to "\wabd\wabd.gif" to have it access the wabd directory for the image. When running WABD in local mode, it is important to properly set the LocalPath property also, or the HTMLImage will probably look in the wrong place. The LocalPath property is simply the directory that is used "as root" for the Web Server. For example, Personal Web Server sets the directory "\WebShare\WWWroot" to root.

For example, if I had a web site called www.ziegler.com, then I could create a directory C:\WebShare\WWWroot\WABD, and when users wanted to access it via the web server, they would simply access www.ziegler.com/WABD. (Thoroughly confused yet?). Basically, in local mode the HTMLImage looks for LocalPath+ImagePath, and during HTML operation it just uses "ImagePath" (because the server is automatically adding in the LocalPath).

Hot Spots
WABD provides easy support for "hot spots". Hot spots are useful for images that represent menu choices, etc. For example, an image may have several objects in it, and if the user clicks on that object he is taken to a different form. To use hot spots, simply place TPaintBox components inside the THTMLImage, size the TPaintBox over the appropriate image region, and create an OnClick handler for the TPaintBox. If the user doesn't click inside a TPaintBox, then the HTMLImage's OnMouseDown event is called (with the X,Y location of the mouse click).

Viewing .GIF & .JPG Images at Design Time
In order to view images during design time, you must create a companion image in .BMP format for each .GIF or .JPG file, otherwise Delphi will not be able to display the image. Simply put the companion image in the same directory as the compressed image. If the HTMLImage component can not find a companion .BMP file, it will display a white box.

13.9 THTMLRaw

This component can be placed on an HTMLForm or an HTMLFormSection. You can enter raw HTML text into the memo field. This allows full access to HTML. Use this component with caution, and only if you are familiar with

HTML syntax. Remember that the HTML you enter in this component will be placed within a <BODY> statement, a <FORM> statement, and if its on an HTMLFormSection, a <TABLE> statement.

`property Lines: TStrings;`
This is the raw HTML text.

13.10 THTMLTable

This is derived from TStringGrid, and has the usual ColCount, RowCount, and Cells properties. It can be placed on an HTMLForm or an HTMLFormSection. If it is a long table (> 10 rows), you should probably just place it directly on the HTMLForm to reduce overhead.

```
property     ColCount: integer;
property     RowCount: integer;
```
The number of Rows and Columns in the table.

```
property    Cells[row,col: integer]: string;
```
The text for an individual cell in the table. The row and col index are zero-based.

```
property     CellBorder: integer;
```
This determines if the Table will have line borders when it is converted to HTML. The higher the number, the thicker the borders. If Cellborder=0 then there will be no line borders.

```
procedure    Fill_N(Data: TDataSet; Num: integer);
procedure    Fill_All(Data: TDataSet);
```
These functions fill the HTMLTable with data from a TDataSet (which can be a TTable or TQuery). If it is a TQuery it should not be uni-directional. With the Fill_N method, you can tell it how many records to fill (using the current record as the starting point).

13.11 Notes for Standard Components

TLabel – You can set the font style to a combination of Bold, Italic, and Underline for special effects.

TMemo – You should set the scroll bars to “vertical” to look as it will in the browser.

TButton – Set the Font to “System” to look as it will in the browser. Also set the Width to just slightly larger than the Button caption.

TEdit – You can set the password char to “*” for password fields.

TComboBox – set the style to “csDropDownList”.

TRadioButton – Due to limitations in the NetManage HTML OCX, TRadioButton’s exhibit strange behaviour in local mode, but work fine in a “real” browser,

TListBox / TComboBox - HTML form processing submits listboxes and comboboxes by value, not by index. Therefore, it makes no sense to provide users with a listbox (or combobox) with duplicate items in the listbox, because only one of the values can be returned.

14 About the Trace Unit

The TraceUnit unit creates a file called TraceFile.txt in the current working directory of the process. It sends “TRACE” statements to this file. This is useful for debugging purposes.

The professional version of WABD comes with a Trace Server. If TraceUnit can find the Trace Server, it executes it (in a different process), and sends the Trace statements to the Trace Server. This allows you to view the Trace statements in real-time. Having the Trace Server in a different process is useful if your application terminates, you can still browse the Trace messages.

15 General Notes

The **event order** is different when you are using WABD. Whenever the user submits some input (by logging on initially or pressing a button or image), your form is created, its handler called, and then it is destroyed. Therefore the "OnCreate" event is called for every user "click". The "OnShow" event is called when the user first logs on, and each time you call ShowFormType. The only event that gets called just once is the Create method of your Global vairables (see Multithreading notes for more about Global variables).

You should **handle all exceptions** within each button-click code snippet (see PoolDemo's TableEditForm LocalWrap method). An unhandled exception will terminate the user's login session, and force them to logon again.

If you create **dynamic components**, set their Tag property to a non-zero value so that WABD will be sure to "remember" your components when the form is recreated on the fly. Also, make sure you place them in the proper grid alignment (based on their Parent THTMLFormSection), and that controls do not overlap. Use the "AlignNow" property of the THTLMFormSection to test for these conditions.

Do not use **HTML reserved tokens** in your button captions, labels, memo text, etc. The tokens you must not use are & (amperstand), < (less than), and > (greater than). If necessary you can user their escape codes: &amp, &lt, &gt.

Forms are placed in tables – if the user tries to select text on the form, it will look funny because non-breaking spaces are required to fill in blank table cells, and they will be selected.

A possible source of trouble that you should watch for: the **user might hit the back button**, change values, and then press a form button. This could cause errors in your program logic. If this could cause serious problems, keep track of a Transaction number in your global variables (and in a THTMLHiddenText component), and make sure that your global variables and the user's form are in synch.

The **NetManage HTML OCX's provided with Delphi do exhibit some strange behaviour**. The HTML control does not support Netscape tables 100%. It has problems with COLSPAN > 1 or ROWSPAN > 1 with skipped cells. The HTML control incorrectly displays Delphi ListBoxes as HTML Comboboxes (with normal dropdown height), and Delphi Comboboxes as HTML Comboboxes (with height of one – looks bad).

If you register the professional version of WABD, enter this as the first line in your implementation section:
{$DEFINE WABD_REGISTERED}
for ConvHTML.pas and WABD_About.pas to remove the "Unregistered" notices.

16 Multi-threading Notes

Since your application will be running underneath a web server, it will have to be able to operate in a multi-threaded environment.

When using the Borland Database Engine, be sure to create a TSession object in your TDataModule. Assign it a unique name in your Global's constructor, and make sure that all TTable's reference that TSession. You may now edit tables and perform queries safely in your thread.

The Visual Component Library (VCL) is not thread safe. The only thread safe objects are in the Classes unit (TList, TStringList, etc), and the non-visual database components (TDataModule, TSession, TTable, TQuery). Do not add any other types of VCL objects to your Global object, because they will not operate correctly in new threads (especially TForm). (This is the main reason why WABD re-creates and destroys your Form for each button-click – the form would be corrupt outside of its main thread)

You can not use true global variables. Howerver, WABD provides full support for “Global” variables that exist just for a given logon session (and throughout the lifetime of that logon session). You just need to define a GlobalCreator function and register it, and WABD will create a Global object that you specify in your function. Place any session initialization in the Create method of your Global’s constructor (see the Pool Demo for more information). A pointer to this Global object is automatically copied to all HTMLForm’s in that session (property BaseGlobals).

Garbage collection is performed automatically on your Global variables. A background thread is created to check each Global variable for the time spend since its last access. If this time is greater than 30 minutes, the variable is automatically destroyed (your destructor will be called, as well as an OnTimeOut event if you declared one). If the user begins a session, and then waits 35 minutes to make his/her next selection, they will receive an error. You may change the value of the timeout to any value (CacheTimeOutSeconds is a global variable). Garbage collection is performed to avoid perpetual resource leaks as users log-off incorrectly.

17 Version History

Version 1.01 March 2, 1997

- Changed TraceUnit.pas to not look for the TraceServer without a $DEFINE
- Updated install notes (to look for *.DCU instead of *.PAS)
- Added direct wire transfer to payment options in documentation

Version 1.00 February 27, 1997

- No bugs (yet!)