Web and VB Concepts

The *World Wide Web* is a vast collection of linked servers that are connected using HTTP. Each web server contains pages, which are documents created using HTML

Web pages can be accessed from remote clients using web browsers, such as Microsoft's Internet Explorer and Netscape's Navigator. Browser software uses the formatting information included in the HTML tags to create a page with the correct appearance on the client machine. Browsers also use the scripting information included in the HTML tags to execute instructions and to invoke OLE controls and Java applets.

One common characteristic of web applications is that they contain hyperlinks to other web pages. Links may be connected to pages physically located on the server or to pages located on another remote web server. These hyperlinks are created using an HTML tag combined with the address (or *URL*) of the linked page.

Intranets are private webs. Like the World Wide Web, intranets consist of servers and clients that are linked using HTTP. Browser software located on the client machines decodes HTML pages on the servers and executes any included commands. Intranets can be very effectively used to create client/server applications. For example, everyone in a company could use an intranet to keep track of each other's schedules and workloads.

Visual Basic 5 is the world's most popular programming environment. You probably have at least some familiarity with VB. (...Why else would you have purchased VBnet, which is a VB add-in?) VB applications are largely made up of forms, which, although not comprised of marked-up text, can be analogized to HTML web application pages.

It's the job of VBnet to help you convert Visual Basic forms to HTML pages. This job includes converting Visual Basic code included with your forms to code that can be contained in an HTML page (VBScript). It also involves creating HTML references to OLE controls that can be used in a web application. Other aspects of the conversion that VBnet handles includes the automatic creation of hyperlink tags and the connection of the HTML application to data sources.

Converting Visual Basic Code to VBScript

VBScript is a subset of the Visual Basic language. (Think of it as Visual Basic "lite".)

The runtime library necessary to execute VBScript code is included in Microsoft's Internet Explorer 3.0. It will also be included in future releases of many operating systems.

VBnet 5.0 automatically handles the job of converting Visual Basic code included in your applications to VBScript. Where VBnet 5.0 cannot convert your code--usually because VBScript does not support a particular keyword or syntax--this will be noted in the report generated by VBnet 5.0.

ActiveX Controls

ActiveX controls are a new terminology for OLE controls (OCXs). In order for an ActiveX control to work on a client machine, either the ActiveX must be present on the client machine or on the server (in which case it must be downloaded before it can be used).

Referencing OLE Controls in Web Applications

OLE controls are referenced in an HTML page using a tag and the control's CLSID, or *class ID*. You can find examples of CLSIDs (which are basically incomprehensible Hexadecimal numbers) in your Windows 95 Registry under HKEY_Local_Machine\Software\Classes\CLSID.

VBnet helps you include ActiveX controls in your web applications by automatically supplying tags and the correct CLSID for controls included in a VB application that it converts.

ActiveX Controls are referenced in an HTML application using the <Object> tag. Following the <Object> tag, the Class ID (or CLSID) is placed in the HTML page. For example, if an OLE control is placed on a VB form, the corresponding HTML tag and entry will look something like this:

```
<! -- ActiveX control name: IeTimer1 class:
      IeTimerObjectsCtl.IeTimer -- >

<! -- Begin support for Internet Explorer -- >
<OBJECT classid="clsid:59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
      id=IeTimer1
      width=63
      height=63
      codebase="../Controls/IETIMER.OCX">
          <param name="TabIndex" value="0">
          <param name="_ExtentX" value="1005">
          <param name="_ExtentY" value="1005">
 </OBJECT>
 <! -- End support for Internet Explorer -- >

 <! -- Begin support for NCompass ActiveX plug-in for Netscape -- >
 <EMBED clsid="59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
      id=IeTimer1
      width=63
      height=63
      code="control/Controls/IETIMER.OCX">

<! -- End support for NCompass ActiveX plug-in for Netscape -- >
```

VBnet takes care of automatically generating the tag, CLSID value, and properties list for each ActiveX controls included in your VB application.

In addition, VBnet places a copy of each control that your internet server will need in a separate Controls directory that is part of your VBnet project. (For example, Crescent\VBnet\WebSite\Project1\Controls.)

ActiveX controls and VBScript are supported under Navigator 3.0/4.0 using the NCompass ActiveX Plug-In.

System Requirements

In order to successfully install VBnet, you will need to have Visual Basic 5 already installed under Windows 95 or Windows NT.

VBnet 5.0 can be used to create client/server data-aware applications using Visual Basic's Data control. (Note that the Data control must be set to access ODBC data sources only.) VBnet also supports the Remote Data Control that ships with the Enterprise Edition of Visual Basic 5.

To view the web applications you have created using VBnet, you will need--obviously--a web browser. You should know that not all features are supported by all browsers. If your application includes VBScript commands and ActiveX controls, to run the application, you will need a browser that supports these features, such as Microsoft's Internet Explorer 3.0 or Netscape's Navigator 3.0/4.0 equipped with the NCompass ActiveX Plug-In. See **Error! Reference source not found.** for information on acquiring the NCompass Plug-In.

To run the server portion of a web client/server application you have created using VBnet, you will need to be connected to the web (or an intranet) and have installed internet server software. For database applications, VBnet 5.0 supports Microsoft's Internet Information Server, O'Reilly's WebSite, and Netscape's Commerce Server.

The Visual Basic Add-in Manager

VBnet 5.0 is a Visual Basic *add-in*. Add-ins are applications that use OLE to manipulate instances of the VBIDE Object. The Visual Basic Add-In Manager is used to enable or disable add-ins. To open the Add-In Manager, select it from the Visual Basic Add-Ins menu.

When an add-in is installed, the add-in is added to the Windows 95 or Windows NT Registry and an entry is included in your Vb.Ini file. For example, you'll find the following entry for VBnet:

```
[Add-Ins32]

VBNETVB5.VBNETCLASS=1
```

The entry indicates that the add-in is active; if you were to uncheck it in the Add-In Manager, it would be de-activated and the entry in Vb.Ini changed to:

```
[Add-Ins32]

VBNETVB5.VBNETCLASS =0
```

Once the VBnet add-in has been made active in the Add-In Manager, you'll find that a VBnet item has been added to your Visual Basic Add-Ins menu. If the Vbnet menu item is checked, the VBnet Toolbar appears on your screen when Visual Basic is open.

VBnet 5.0 Toolbar

There are five buttons on the Toolbar: Make HTML Form, Make Web Application, Run Web Application, Options, and Help. They perform the following functions:

- *Make HTML Form* is used to convert a Visual Basic form into its HTML equivalent.

- *Make Web Application* is used to add database functionality to internet and intranet applications.

- *Run Web Application* is used to view the appearance and functionality of your web applications in your internet browser.

- The *Options* dialog is used to configure VBnet.

- *Help* provides VBnet documentation.

VBnet Options Dialog

The VBnet 5.0 Options Dialog is a tabbed dialog with five panels: Web Page, Web Site, Database, General, and About.

VBnet Options settings will be remembered by VBnet for future VBnet sessions.

Web Page Panel

Selecting a Target Browser

The first panel of VBnet's Option dialog is used to select a target Internet browser for your web application. The choices are Microsoft's Internet Explorer 3.0 or Netscape's Navigator 3.0 or higher. Your web application will be tuned for the browser you select

The primary difference in the converted HTML pages prepared for the two browsers concerns the way the pages are laid out so that they will appear properly, depending on the browser.

Web applications that are designed for Explorer utilize VBScript and ActiveX OLE controls, whereas web applications designed for Navigator use a plug-in to run VBScript and ActiveX controls. Database applications created with Visual Basic are run using JavaScript under both Internet Explorer and Netscape Navigator.

The Web Page panel is also used to determine whether your application will use ActiveX controls (32 bit only) or the intrinsic HTML controls using the <INPUT> tag. Here is an example of using an intrinsic HTML control in a web page:
<INPUT type="TEXT" size=20 name="txtHello" value="Hello!">
In addition, you can use the Web Page to decide whether your application will include the extended Style Sheets supported by Internet Explorer 3.0, or whether it will use standard HTML.

Web Site Panel

Choosing an Internet Server

The second panel of the VBnet Options dialog is used to choose the web server you will be using.

Different internet servers support different web data access methods and techniques. This edition of VBnet creates applications that are designed to run with Microsoft's Internet Information Server, O'Reilly's WebSite, or Netscape's Commerce Server.

Web Application Destinations

The Web Site panel of the VBnet Options dialog is also used to specify the destination for your web application and the location for your web site's home page.

The default location for your web application is in a directory created under the VBnet\WebSite directory. You can change this location for a particular project anytime you run the VBnet

Options dialog. However, VBnet will always create a directory structure using the name of the Visual Basic project that you are converting, and place this structure under the "WebSite" directory you have designated.
You cannot change the name of the web application directory at this point--it will always be derived from the Visual Basic project name.

For example, if you created a Visual Basic application named Project1, VBnet would create a Project1 directory and four subdirectories.

Since VBnet uses the Visual Basic project name to create the web application destination, it's a good idea to name your VB projects with care.

For each project, VBnet creates the following subdirectories: Controls, Forms, Images and Scripts as shown in the table below. These directories contain the different elements of the web application that VBnet has created.

Web Application Directories Created By VBnet:

Directory	Example	Contains
Controls	Website\Project1\Controls	ActiveX controls referenced in the VB project
Forms	Website\Project1\Forms	HTML files converted from VB forms
Images	Website\Project1\Images	Image files--such as JPEGs--used in the web application
Scripts	Website\Project1\Scripts	Database connectvity script files

Web Site Home Page

The Web Site Home Page text box is used to specify the location of your web site's home page so that VBnet can locate required files.

Database Panel

VBnet 5.0 supports data access via Microsoft's Internet Database Connector (IDC). For information about add-on products that support Microsoft's Active Server Framework and Netscape's Internet Application Framework, please contact Crescent.

The Client Data Access Object checkbox implements a two-tier distributed DAO model suitable for intranets, but not intended for use on the world wide web.

General Panel

You should know that VBnet requires that VB project and form files be saved before VBnet 5.0 can run.

When "Save Before Generation, Don't Prompt" is selected VBnet will automatically save your project files each time VBnet 5.0 is used rather than prompting you to save the files.

If "Save Before Generation, Prompt" is selected, VBnet will prompt you to save the current VB project before VBnet 5.0 runs.

Report VBScript Syntax Errors

If this option is selected, the VBnet 5.0 report will list and Visual Basic code that could not be properly converted to VBScript.

Default Browser Location

Use this browse button to tell VBnet the location of the Internet browser that you wish to view web applications with.

About Panel

This panel contains the information you need to contact Crescent sales or technical support.

The VBnet 5.0 Report

After you select the "Make HTML Form", "Make Web Application", or "Run Web Application" from the VBnet Toolbar, VBnet generates a complete report.

You can view this report on screen, or print it out for later reference.

The statement "VBnet successfully generated your project" at the end of the report means that there were no problems in converting your VB form to an HTML page, your VB project to a web application, or running your application, as the case may be. If you do not see this statement at the end of the report, you should examine the report carefully to determine the nature of the problem.

Note that certain syntax errors or problems may occur that will not be reported. For example, VBnet may be unable to insert an OLE control since that control cannot be used on the web. In such cases, VBnet will still report successful generation.

VBScript Syntax Checking

Provided that you have enabled the VBScript Syntax Check option on the General panel of the VBnet Options dialog, the report will list Visual Basic code that VBnet 5.0 could not convert in compliance with VBScript syntax.

As it goes along, VBnet will generate specific comments indicating errors in the converted code.

VBScript is a subset of Visual Basic, meaning that not all VB keywords and commands are supported by VBScript. For example, Visual Basic's *Select Case* statement is not supported by VBScript.

In addition, VBScript is capable of interacting with exposed object properties of Internet Explorer.

As a simple example of how Visual Basic code is converted to VBScript, open a new VB project and add a command button and a text box to a form. Next, add code to the command button's click event that will display text in the text box:

```
Private Sub Command1_Click()
      txtDisplay.Text = "Greetings from Crescent."

End Sub
```

VBnet 5.0 converts this click handler as follows:

```
<! ------------------------------------------ >
<! -- VBScript derived from Visual Basic code >
<! ------------------------------------------ >

<SCRIPT LANGUAGE=VBS>

Sub Command1_onClick()
    ' Support for Internet browser object model
    Dim Form1
    Set Form1 = document.Form1
    Form1.txtDisplay.value = "Greetings from Crescent
End Sub


</SCRIPT>
```

It's important to understand what VBnet does with Visual Basic code that has no direct VBScript equivalent. We've taken an approach in line with Hippocrates' admonition to "at least do no harm." If there is no clear way to convert Visual Basic code to VBScript, VBnet warns you in its report and adds specific in-line comments. In most cases, this means that if you attempt to load the HTML page containing the converted code in a browser you will receive a syntax error message. The report and in-line comments can be used to manually tune your code.

For example, conditional compilation is unsupported in VBScript. Suppose you included code in a Visual Basic event handler along the lines of:

```
#Const English = True
#If English Then
    MsgBox "Hello, World"

#End If
```

When you used VBnet to convert the application containing this code, the VBnet report would include syntax error warnings as follows:

```
--  Generating VBScript code:
   --  Generating VBScript Sub: Form_Load
       --  VBScript syntax error  --  "#Const"  --  invalid keyword
       --  VBScript syntax error  --  "#If"  --  invalid keyword

       --  VBScript syntax error  --  "#End"  --  invalid keyword
```

In addition, a comment would be placed in the VBScript for each unsupported keyword, for example:

```
<! -- WARNING -- keyword: "#If" not supported by VBScript -- >
```

It would then be up to you to manually adjust the VBScript code as appropriate.

Visual Basic Form Files versus Web Page Files

VB forms are the basis for the appearance of Visual Basic programs, just as HTML pages are the basis of web application appearance.

Visual Basic programmers are used to designing VB forms in the WYSIWYG graphic design environment of the VB Integrated Development Environment (IDE). This means that the act of designing the appearance of a VB form is achieved by dragging and dropping, and drawing controls on a form, and by setting form and control properties. But the underlying structure of a saved VB form file (.Frm file) is an ASCII text file that indicates properties, values and positioning using a coordinate system. (Forms that include graphical elements, such as bitmaps, or any other complex binary information, also have an associated .Frx file that contains the binary information.)

As you probably know, it is also possible to visually edit HTML pages, using a number of more-or-less WYSIWYG editing tools available. However, the underlying structure of the resulting web pages (.Htm, or in Unix environments .Html files) is radically different from that of VB form files. Appearance is manipulated through tags indicated with brackets (e.g., <I am a tag>). There are no tags that act to size or position general elements using a coordinate system.

Unlike Visual Basic forms, web pages--HTML documents--are not based on a coordinate system. This means that translations of VB forms to HTML documents will have an appearance similar to the original VB form. VBnet 5.0 does attempt to approximate control positioning in VB-based web pages. However, HTML pages are different from VB forms. You should, therefore, not expect WYSIWG results when using VBnet to convert VB forms.

The following rules of thumb will help to insure that your HTML pages look as close as possible to the original VB forms:

- Each VB form's ScaleMode property should be set to Twips in the Properties Window.

- The VB frame control is not supported by VBnet. Note, however, that controls contained within a frame will be converted by VBnet and laid out properly.

- For best results, the FontSize property for combo, list and text boxes, and command buttons should be set at 8 points (the Visual Basic default).

- Conversion to HTML of the VB form BackColor property and any VB standard control's ForeColor property settings is supported. The VB form ForeColor property and any standard control's BackColor property are not supported.

- For best results, labels, option buttons and check boxes should be created with borders no larger than necessary. Set the AutoSize property of Label controls to True.

- Option buttons (radio buttons) should be used only in control arrays.

Form-to-Form Hyperlinks

Forms can be connected in Visual Basic in a number of ways. For example, suppose you have a project containing two forms: Form1 and frmLink. You could display frmLink from Form1 by executing a .Show method:

```
    frmLink.Show
```

VBnet automatically converts form invocations in VB that use this formulation to web-style hyperlinks.

Commonly Used HTML Positioning Tags

The tags listed in the table below are often used to position elements within an HTML page. (Note that most tags require a close, indicated with a slash, e.g., <Tag>...</Tag>. Only the material within the block of the tag is formatted by it. The table below omits the tag block close.)

For more information on HTML, have a look at one of the numerous good books available on the topic, or do a web search for "HTML".

Some HTML Tags used for positioning and formatting:

Tag	**Meaning**
<A>	Hyperlink anchor
<ADDRESS>	Format an address section
<BACKGROUND>	Page background graphic
<BR>	Force line break
<EM>	Typographic emphasis, usually italics
<H1>...<H6>	Format six levels of headings
<HR>	Add a line to the page (horizontal rule)
<IMG>	In-line graphic
<P>	New paragraph
<Table>	Information formatted as a table
<Title>	HTML document title

VBnet and Object References

The object model of Internet Explorer is exposed to VBScript.

In the Internet Explorer object model, forms are sub-objects of HTML document objects. In other words, a Visual Basic control property reference such as:

```
    Form1.Text1.Text
```

is converted by VBnet using an exposed document object of the browser:

```
    document.Form1.Text1.value
```

VBScript HTML event handlers are formally slightly different than Visual Basic event handlers for *intrinsic* (meaning VB5 standard) controls.

For example, the Click event in Visual Basic is referred to as onClick in VBScript. Thus, Visual Basic's

```
    Sub Command1_Click
```

becomes VBScript's

```
    Sub Command1_onClick
```

Note that custom ActiveX controls have their own properties, events and methods as defined in the control's type library. Both Visual Basic and VBScript use the control's own name for its events.

The VBScript Language

VBScript supports much of Visual Basic's functionality. The table below shows what Visual Basic syntax is supported by VBScript.

It's important to realize that the only variable type supported by VBScript is variant. VBnet converts all variable declarations with this in mind. Thus Visual Basic's

```
Dim MyVariable As String
```

becomes, in VBScript

```
Dim MyVariable
```

The Visual Basic syntax supported and not supported by VBScript:

Functionality	***Statement***	***Supported by VBScript?***
Arrays	Array function	Yes
	Declaration (Dim, Static, etc.)	Yes
	Erase	Yes
	LBound	Yes
	Declaring arrays with Lbound<>0	No
	Option Base	No
	ReDim	Yes
	UBound	Yes
Assignment	=	Yes
	Let	Yes
	Set	Yes
Calling DLLs	Declare (calls to external libraries are not allowed in web applications)	No
Classes	Clipboard object	No
	CreateObject	Yes
	Dim x As New TypeName	No
	Set x=New TypeName	No
	If TypeOf x Is TypeName	No
	With...End With	No
Code Control	Line continuation character (_)	Yes
	Line separation character (:)	Yes
Collections	Add	No
	Count	No
	Item	No
	Remove	No
	Collection access operator (!)	No
Comments	Rem, '	Yes
Conditional compilation	#Const #If...Then...#Else...#End	No
	If	No
Constants	Many VBA constants are not supported	No

Control flow	AppActivate	No
	Beep	No
	Command function	No
	DoEvents	No
	Do...Loop	Yes
	Environ function	No
	For...Next, For Each...Next	Yes
	GoSub...Return	No
	GoTo	No
	If...Then...Else	Yes
	Line numbers and labels	No
	OnError...GoTo	No
	Select Case	No
	SendKeys	No
	Shell	No
	While...Wend	Yes
Conversion functions	Ans	Yes
	Asc, Chr	Yes
	CBool, CByte	Yes
	CCur	No
	CDate, CDbl, CInt	Yes
	Chr$, Hex$, Oct$	No
	CLng, CSng, Cstr	Yes
	CVar, CVDate	No
	CVErr	Yes
	DateSerial, DateValue	Yes
	Fix, Int, Sgn	Yes
	Format, Format$	No
	Hex, Oct	Yes
	Str$, Str, Val	No
Data types	Data typing is not allowed, meaning all variables are variant. Type suffixes are not used. User-defined classes are not allowed. The Me keyword is not allowed.	No
Date and time	Date and time functions	Yes
	Date and Time statements	No
	Date$, Time$	No
	Day, Month, Weekday, Year	Yes
	Hour, Minute, Second	Yes
	Now	Yes
	Timer	No
	TimeSerial, TimeValue	Yes
DDE	DDE is not supported	No
Error trapping and debugging	Erl, Error, Error$	No
	Err object	Yes
	On Error...Resume	No
	On Error Resume Next	Yes
	Resume, Resume Next	No
File I/O	Most file access is not allowed	No
Financial functions	Many financial functions are not supported	No
Graphics functions	Most graphics and printing functions are not supported	No

Literals	Empty	Yes
	Nothing	Yes
	Null	Yes
	True, False	Yes
	User-defined numbers or variable names	Yes
	User-defined real numbers using scientific notation dates, or trailing type characters	No
Mathematical functions	Atn, Cos, Sin, Tan	Yes
	Exp, Log, Sqr	Yes
	Randomize, Rnd	Yes
Object and method references and functions	AddItem, RemoveItem	No
	Arrange, ZOrder, SetFocus	No
	Dot operator (.), for example, document.form1	Yes
	Drag	No
	Hide, Load, Move, Show, Unload	No
	InputBox$	No
	PaintForm, Refresh	No
Operators	All operators except Like	Yes
Procedures	Declaring procedures:	
	Function	Yes
	Property Get/Set/Let	No
	Specifying Public/Private	No
	Sub	Yes
	Calling procedures:	
	Call	Yes
	Exiting procedures:	
	Exit Function	Yes
	Exit Property	No
	Exit Sub	Yes
	Parameters for procedures:	
	ByVal, ByRef	Yes
	ParamArray	No
	Optional	No
Strings	Asc, AscB, AscW	Yes
	Chr, ChrB, ChrW	Yes
	Fixed length strings	No
	Format	No
	Instr, InStrB	Yes
	Len, LenB	Yes
	LCase, UCase	Yes
	Left, Right	Yes
	LeftB, MidB, RightB	Yes
	Mid function	Yes
	Mid, LSet, RSet statements	No
	Space function	Yes
	StrComp	Yes
	StrConv	No
	String Function	Yes
	Trim, LTrim, RTrim	Yes
Structures	Type...End Type	No
	LSet, RSet	No
User-interface	InputBox	Yes

	MgBox	Yes
Variable scoping	Module-level:	
	Const	No
	Dim	Yes
	Private	Yes
	Public, Global	No
	Procedure-level:	
	Const	No
	Dim	Yes
	Static	Yes
Variant functions	IsArray	Yes
	IsDate	Yes
	IsEmpty	Yes
	IsError	Yes
	IsMissing	No
	IsNull	Yes
	IsNumeric	Yes
	IsObject	Yes
	VarType	Yes

We think you'll pretty much get the picture after studying this table. VBScript is familiar, so it is easy for Visual Basic programmers to use. However, VBScript has been crafted so that it runs remotely in a stream. In addition, security concerns compel some limitation of VBScript control over distributed environments.

Between the various concerns, you'll find that VBScript--while capable in many ways--has been intentionally designed with limitations. These limitations function to:

- Cripple the ability to invoke external libraries or executables which might not be available to a distributed web application.

- Limit control over distributed environments, implying little printing, graphics or file manipulation capability.

Please visit Microsofts VBScript website *http://www.microsoft.com/vbscript/* to obtain a list of Visual Basic 5.0 commands that are supported by VBScript.

Configuring Your System

Visual Basic ODBC database applications built using the Data control can quickly and easily be converted to client/server web applications using the VBnet 5.0.

The following restrictions apply to client/server projects that are created in VB and converted by VBnet to HTML:

- All data access statements--e.g., SQL statements--must be set in control properties, not in code

- The application will only run under a JavaScript-enabled browser such as Netscape Navigator 3.0 or Internet Explorer 3.0

To create a VB client/server application that can be converted to a web application, you must first add an ODBC driver for the data source to your system.

From the Windows control panel, select the 32bit ODBC applet. The 32bit ODBC applet in the control panel is used to add a data source to your system.

Select the Add button on the right of the Data Sources dialog.

The Add Data Source dialog will appear. Select an installed ODBC driver and click OK. Use the ODBC Setup Dialog to select a database and give the connection a name.

This dialog is used to select a database, and give a name to the connection to that database for use with the Data control. Note that the name you choose for your data source must also be the name used when you use the ODBC system DSN Setupdialog to create an ODBC data source for your Windows NT server.

Creating a VB Client/Server Application

To create a simple Visual Basic client/server application, drop a Data control onto your form by double-clicking on the Data control in the VB toolbox.

If the Data control is not present in the toolbox, check that it has been added to your project by selecting Custom Controls from the VB Tools menu. For the control to be included in your project, it must be checked in the Custom Controls dialog. If it is not included at all in the Available Controls list, check to see that it is on your system by clicking the Browse button.

With the Data control selected on your form, click Custom in the Properties Window. The Data Control Properties dialog will open on your screen.
The Custom Properties dialog of the Data control is used to set the control's data source and enter SQL statements.

You can use this dialog to enter the data source and any SQL statements to be run against the data source.

To configure a sample application, make sure that the General tab in the Custom Properties Window is selected, then use the DataSource drop-down list box to select an ODBC data source. Using the multi-line text input box at the bottom of the General tab page, enter any SQL statements that you would like.

Next, add a control to the form--such as a text box--that can be bound to a data source. Set the control's DataSource property to the name of the Data control (e.g., Data1) using the drop-down list.

Note that VBnet 5.0 only supports binding for the Visual Basic standard control set, that is, the Checkbox, ComboBox, Image, ListBox, Label, and Textbox controls.

Connect the data bound control to a particular field in the data source using the control's DataField property drop-down list box. After binding a control using its DataSource property, the control's DataField property can be used to connect it to a particular field found within the data source.

When you run the project, the user can advance the Data control through the matching fields in the data source. Fields are displayed in a bound control based on the SQL statement in a Data control that is connected to an ODBC data source. The next step is to convert the VB app to a web application.

Converting the VB App to a Web Application

To convert the VB data access application to a web client/server application, click the "Make Web Application" button on the VBnet Toolbar.

The resulting HTML application uses JavaScript and server-side scripts to access ODBC data sources across the web.

If you are running Windows NT server (with Internet Information Server), it must be configured with the same data source name that was used on the machine that created the VB application. In addition, after copying the directory structure generated by VBnet to your web server, you must use the Microsoft Internet Service Manager to allow Read and Execute access to the directory.

The following pointers should help you to get your Visual Basic data access applications up and running as web applications:

- You can add as many Data controls to a form as you'd like; each can be used to bind a different control to an ODBC data source. Note that ODBC access must be used.

- Actions you wish theData control to perform are set using the control's .Tag property. The allowed values for the .Tag property are INSERT, UPDATE, DELETE, and QUERY. Multiple values in any sequence or combination are allowed depending on the actions you wish to execute. Any character can be used as a separator. If you leave the .Tag property empty, the action will default to QUERY.

- If multiple editable controls are attached to a single column through the same Data control, only the first control will be used for INSERT purposes.

- If a Data control SQL property is left blank or it is not bound, a stub JavaScript function will be created. The HTML page will need to be edited manually using a text editor to enter the location of the CGI script.

- VBnet supports option button control arrays. No other control arrays are supported.

- Joint queries require the use of column aliases.

-INSERT, UPDATE, and DELETE based on queries referring to multiple tables use only the first table in the SELECT FROM statement.

- Subqueries are not supported.

- VBnet 5.0 only supports Web database access for the Visual Basic standard controls.

- Make sure that your check boxes are bound to a numeric field since the only valid VB property values for the checkbox control are 0, 1, and 2.

- Images need to be bound to a VARCHAR field containing the location of a .Jpg or .Gif file.

Obtaining Technical Support

The Crescent technical support staff is ready to help you with problems that you encounter when installing or using VBnet 5.0. The Crescent technical support staff will do its best to help you succeed with VBnet 5.0.

For fastest service, please provide the following information when contacting Crescent technical service:

- Name
- Company Name
- Address
- Phone and Fax Numbers
- Email Address
- Product Name and version
- Product Serial Number
- Platform (Win 95, NT, 3.1, etc)
- Development Environment (VB5, VB4/16, VB3, etc)
- The specific question or problem you are having
- The steps required to reproduce a problem if applicable

To request technical support, you may contact Crescent:

Via Web Site: *http://crescent.progress.com/techsupport.html*

By Email:	crescent-support@progress.com
By FAX:	01.781.280.4025
Via BBS:	01.781.280.4221
Via FTP site:	ftp.progress.com/pub/crescent
By Telephone:	Contact North American technical support staff at 01.781.280.3000 Monday through Friday from 9:00 a.m. to 5:00 p.m. EST.
By Mail:	Address your correspondence to: Technical Support Crescent Division, Progress Software Corporation 14 Oak Park Bedford, Massachusetts 01730 United States

End User Licensing Agreement

End-user Product License Agreement

Subject to the following terms and conditions, Progress Software Corporation (PSC) grants to you ("User") a non-exclusive license to use the Product and the related Documentation accompanying this license Agreement. The term "Product" means all the computer software products licensed to User as a single integrated product and the term "Documentation" refers to all the manuals licensed to User.

1. Scope of License

1.1 This license allows User to install and use the Product solely on a single computer (i.e. with a single central processing unit). Except as otherwise specified in the Documentation, User may not grant sublicenses, leases, or other rights in the Product, nor may User transfer, sell, assign, or otherwise convey the Product to another party without PSC's prior written consent. User may not split the Product into its component computer software products and transfer, sell, assign, distribute or re-license or otherwise convey those components as individual products to another party. Transfer of the Product to another computer may be made on a permanent basis provided no active copies are retained on the original computer. This Agreement automatically terminates if User transfers possession of any copy of the Product or Product Update to another party.

1.2 A Product Update replaces part or all of a Product or Product Update previously licensed. Use of a Product Update terminates the license to use the Product or that part of the Product which the Product Update replaces and User shall destroy or return to PSC all copies of any prior Product or Product Update. User may obtain rights to acquire Product Updates and other technical services under PSC's then current fees and terms.

2. Proprietary Rights. The Product and Documentation are proprietary products of PSC's licensor(s) or PSC and are protected by copyright law. By virtue of this Agreement, User acquires only the non-exclusive right to use the Product and does not acquire any rights of ownership in the Product or the media upon which it is embodied. PSC's licensor(s) or PSC, shall at all times retain all rights, title, and interest in the Product and the media.

3. Non-Disclosure; Copies; Alterations. User agrees not to cause or permit the reverse engineering, disassembly, copying, or decompilation of the Product, except to reproduce machine-readable object code portions for backup purposes and installation of new releases, under penalty of license termination but not exclusive of any other remedies. User may copy the Product for installation, backup or other purposes as described in the Documentation. User may not copy nor allow others to copy the Product or Product Update for any other purpose. User agrees not to remove any product identification, copyright notices, or other notices or proprietary restrictions from the Product. User may not copy nor allow others to copy any part of the manuals or other printed material provided with the Product or Product Update by any means, including data transmission or translation.

4. Limited Warranty. PSC warrants that it has the right to license the Product(s). PSC will defend User against any claim based on an allegation that a Product infringes a U.S. patent or copyright, but only if PSC is notified promptly in writing of such claim and is given sole control of the defense thereof and all related settlement negotiations relating thereto. Notwithstanding the foregoing, PSC shall not be liable to User for any claim arising from or based upon the alteration or modification of any of the Product(s).

The Product has been tested and the Documentation has been reviewed. However, except as specifically stated above, PSC MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESSED OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTIBILITY OR FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THIS PRODUCT AND DOCUMENTATION. For example, PSC does not warrant that there are no discrepancies between the Product and the Documentation, nor that errors cannot arise during the use of the Product.

THIS WARRANTY GIVES THE USER SPECIFIC LEGAL RIGHTS, AND MAY ALSO IMPLY OTHER RIGHTS WHICH VARY FROM STATE TO STATE. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES, AND DO NOT ALLOW A LIMITATION ON HOW LONG ANY IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS MAY NOT APPLY. No PSC employee, supplier, or agent is authorized to make any modifications or addition to this warranty.

5. Limitation of Liability. TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE LIABILITY OF PSC, OR ANY OF PSC'S SUPPLIERS OR ON-LINE SERVICE PROVIDERS, IF ANY, FOR DAMAGES RELATING TO ANY PRODUCTS SHALL BE LIMITED TO THE ACTUAL AMOUNTS PAID BY USER FOR SUCH PRODUCT AND SHALL IN NO EVENT INCLUDE INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND. SOME STATES DO NOT ALLOW THE EXCLUSION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MAY NOT APPLY

6. Export Administration. User agrees to comply fully with all relevant regulations of the United States Department of Commerce and with the United States Export Administration Act to assure that the Product and Documentation are not exported or re-exported in violation of United States law. Further, User shall not directly or indirectly export or re-export any Products, Documentation, or the direct Product thereof without first obtaining PSC's written approval.

7. U.S. Government Restricted Rights. The Product is provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Program Product clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/manufacturer is the Crescent Division of Progress Software Corporation, 14 Oak Park, Bedford, MA 01730. Unpublished-all rights reserved under the copyright laws of the United States.

8. Taxes. User shall be responsible for paying any sales or use tax imposed at any time whatsoever on this transaction.

9. Miscellaneous. This Agreement is governed by the laws of the Commonwealth of Massachusetts. If any provision of this Agreement is declared invalid, illegal or unenforceable, the remaining provisions of this Agreement shall remain in effect.

Hypertext Transfer Protocol

Hypertext Markup Language, a tagged markup language, based on ISO Standard Generalized Markup Language (SGML). Formatting information and other specialized instructions (for example, executable instructions in a script language) are indicated by tags, keywords enclosed in brackets.

A user interface that allows quick connection to related topics by clicking on the link.

Forms are used to create windows in Visual Basic.

An add-on program that extends the functionality of Netscape.

Open Database Connectivity

A private network that uses web browsers and HTTP.

Visual Basic Integrated Development Environment

A private profile string file containing information about your Visual Basic preferences and settings.

When creating a client/server web application, images such as .Jpg or .Gif files need to be bound to a VarChar field containing their location.

For more information on Visual Basic language, syntax and features, refer to the VB4 Language Reference manual.

You might want to take a look at Ian S. Graham's HTML Sourcebook (2d Edition, John Wiley & Sons, 1996) and Paul J. Perry's World Wide Web Secrets (IDG Books, 1995).

For more information on Visual Basic language, syntax and features, refer to the VB4 Language Reference manual.

For more information on ODBC setup, please refer to Microsoft documentation.

For example, the SQL statement SELECT * FROM products was entered to select all data in the Products table of the Northwind database   that ships with Access 7.

Please see the Microsoft Internet Service Manager documentation for more information.