A Programmer's Guide to the Internet

Version 8.1.3

Micro Focus
9920 Pacific Heights Blvd.
San Diego, CA 92121
858.795.1900

E-01-UG-100501-Internet-8.1.3

Contents

Chapter 1: Introduction

Chapter 2: Choosing a Deployment Method

Chapter 3: Using the Thin Client to Launch Web Applications

Chapter 4: Launching Web Applications Through CGI

Chapter 5: Using the ACUCOBOL-GT Web Runtime

Chapter 6: Other Internet Solutions

Appendix A: Building and Hosting a Web Site

Appendix B: Adding Internet Features to Your Program

Appendix C: Use the Runtime as a Helper Application or Viewer

Glossary of Terms

Index

1 Introduction

Key Topics

1.1 Introduction

Unlike anything before it or since, the Internet has taken the world of computing by storm. It is said that over 800 million people have Internet access today, that there are over 400 million Web sites. Even after the implosion of the dot com era, some industry watchers report that the number of Internet users is growing at a rate of 10% per month, and that the number of Web sites grows by 1.5 million *each day.*

So it's no wonder that the Internet, the Web, and the promise of electronic commerce remain a vital concern to corporate IS departments. How can they take advantage of the infrastructure that's in place, not to mention the user base that it reaches? Do companies have to start over, retrain their development staff, spend most of their computing budget?

On the contrary, Internet commerce is closer at hand than you might think. Rather than rebuilding your applications with a new "Internet" language, you can provide access to applications and data over the Internet without ever leaving COBOL!

Using ***extend*** technologies, you can deploy existing ACUCOBOL-GT® applications on the Internet today. This book explains how. You can make your applications accessible through popular Internet browsers and the World Wide Web, or you can harness the Internet in a more secure TCP/IP networking configuration. When new Internet initiatives surface, you can perform your development in ACUCOBOL-GT, or you can combine your COBOL programs with other Internet languages and technologies.

1.2 Purpose/Scope of Book

This book is designed to teach ACUCOBOL-GT developers how to deploy applications on the Internet. It includes a glossary of Internet technology and terminology, a description of alternative approaches to Internet deployment, specific instructions on implementing each approach, and samples where appropriate.

After reading this book, ACUCOBOL-GT developers will have a clear understanding of how to integrate COBOL programming with their current Internet strategies, how to shape future Internet strategies, and how to minimize development time and expense.

Unless otherwise indicated, the references to “Windows” in this manual denote the following versions of the Windows operating systems: Windows XP, Windows Vista, Windows 7, Windows 2003, Windows 2007, Windows 2008 R2. In those instances where it is necessary to make a distinction among the individual versions of those operating systems, we refer to them by their specific version numbers (“WindowsXP,” “Windows Vista,” etc.).

1.3 What You Need to Know

Although you may believe that the Internet is quite complex, all that you need to know to deploy your ACUCOBOL-GT applications over the Internet is COBOL. You can choose to use popular Internet languages like HTML, XML, and Java to deploy your applications, but you don’t have to. You can even write your CGI (Common Gateway Interface) programs in COBOL, if you use them at all. This book explains both how to stay with what you know, COBOL, and how to combine COBOL with other Internet technologies.

So what are your prerequisites for using this book?

You must be a COBOL developer familiar with ACUCOBOL-GT.

That’s it!

1.4 What You Don’t Need to Know

To use this book, you do *not* need to be expert in:

- HTML
- XML
- CGI

- Java

This book provides an overview to all of these languages and technologies. If you choose to use them in your application deployment, this book gives you some valuable guidelines and instructions as well as references to more detailed information. It also provides alternatives so that you can leverage your expertise in COBOL whenever possible.

1.5 Product Integration

This book describes several methods for deploying ACUCOBOL-GT applications over the Internet. All of the methods described involve one or more technologies from the ***extend*** family of solutions. Just as all of the technologies in the ***extend*** family work together to provide a complete enterprise computing solution, the ***extend*** technologies work together to provide a complete Internet solution as well. ***extend*** technologies can work alone or in combination with other technologies, depending on the functions that you require. For example:

- If you want to provide access to remote Vision, relative, or sequential data and object files over the Internet, you can use our AcuServer® file server technology.

- If you want to provide access to that data from a Web link, you can combine our Web thin client or Web runtime technology with AcuServer. Our AcuConnect® application server may be used to serve the thin client requests or to launch remote application components in a distributed environment.

- If you want to provide access to remote RDBMSs and applications, you can use our Acu4GL® COBOL-to-RDBMS interface in combination with AcuConnect and/or the thin client. Or you can embed SQL into your COBOL application and use our AcuSQL® precompiler in combination with these technologies.

- If you want to provide a user interface in HTML, you can write CGI programs in ACUCOBOL-GT to bridge the HTML and COBOL.

- And the list goes on . . .

For clarity, this book describes the simplest scenarios. Remember that a combination of approaches may in fact be more ideal for you.

Note: Although you can use earlier versions of ***extend*** technologies to deploy your applications and data on the Internet, the information included in this book is specific to the ***extend*** Version 8 and higher. For Internet-related information on previous versions of our technologies, please refer to an earlier version of this book.

1.6 Technical Services

For the latest information on contacting customer care support services go to:

http://www.microfocus.com/about/contact

For worldwide technical support information, please visit:

http://supportline.microfocus.com/xmlloader.asp?type=home

2 Choosing a Deployment Method

Key Topics

2.1 What Are My Options?

We offer a variety of different solutions for deploying COBOL applications on the Internet.

Some of them allow you to make your COBOL programs and data accessible on the Web from popular Internet browsers. Others allow you to harness the Internet in a more secure TCP/IP networking configuration.

When discussing your alternatives, we frequently make a distinction between the Internet and Web. We define the Internet as a global TCP/IP network, and the Web as a mechanism for finding and viewing information on the Internet.

Our **Web-based solutions** involve the use of browsers and often links on a Web page. Our **other Internet solutions** use more traditional methods of launching applications.

2.1.1 Web Solutions

Following are some of the ways you can deploy your applications on the Web:

- You can add the ACUCOBOL-GT® **Web Thin Client** to your Web page so that when users visit your site, the thin client downloads and installs on their machines and automatically launches your application on the server. In thin client architectures, the application logic runs on the server. Only the user interface displays on the client.

- You can add the ACUCOBOL-GT **Web Runtime** to your Web page so that when users visit that page, the runtime downloads and installs on their machines and automatically launches your application locally.

- You can create a Web interface to your COBOL application and allow users to interact with pages on your Web site via an HTTP browser or mobile device using our **COBOL CGI** technology.

These options are described below in more detail. If your users already have a licensed copy of the ACUCOBOL-GT runtime on their machine, they can also gain access to your applications on the Web by setting up the runtime as

an Internet helper application or viewer inside their browser. When they click a link on your Web site, the browser knows to associate the application with the ACUCOBOL-GT runtime. This is discussed in **Appendix C**.

In addition to all of these approaches to Web application deployment, ACUCOBOL-GT includes a WEB-BROWSER control that lets you add a variety of Internet features to your COBOL program. With this control, your programs can support Web browsing, display HTML pages, invoke e-mail, telnet, and FTP services, and more.

Web Thin Client

If you want Windows users to launch applications from your Web site and have the applications run exclusively on the remote server, you can use the ACUCOBOL-GT Web Thin Client. In this scenario, end users simply visit your Web site. The Web browser searches for the Web thin client on their machines. If successful, it launches the program on the server. If it cannot locate the Web thin client, it provides the software automatically with users' permission. It then invokes the server application transparently and "projects" the user interface back onto the client. The Web thin client is an ActiveX version of our thin client solution.

Alternatively, end users can install the standard ACUCOBOL-GT Thin Client on their local machine. They can install it from any ACUCOBOL-GT media or, subject to appropriate licensing agreements, you can distribute it on your Web site so that end users can download and install it from there. Using an Active Server Page (ASP), Java Server Page (JSP), Visual Basic, or perl script, you can automate the download and install process for users if you like. Once they have the thin client installed, they can visit your Web site and click a link to invoke your application.

Thin client users always have the option of executing the **acuthin** command with an Internet server or IP address as part of the command parameters. **acuthin** can launch programs on any server in a TCP/IP network, including the Internet. The only components required on the client in this case are the thin client software and an Internet connection. Users don't even need to have a Web browser.

With any of these thin client options, all application processing is performed on the server. Usually, data access is considered local because the data resides on the same server machine as the application. If you want to keep

data on a different server in a multi-tiered configuration, you can combine the thin client with our AcuServer® technology. Please note that although the thin client supports only Windows clients, it gives access to both Windows and UNIX servers running the AcuConnect® application server software.

Chapter 3 describes Internet application access via the ACUCOBOL-GT Thin Client and Web thin client.

COBOL CGI

Perhaps you want customers or users to run your applications by clicking a link on your Web site, but you don't want to require anything special of the user's machine (for instance, the presence of any ACUCOBOL-GT runtime, be it a standard, thin client, or Web runtime). In this case, you can create a new interface to your application using a markup language such as HTML, WML, or XML. With a Web interface, your application can be interpreted directly by the user's HTTP browser or mobile device, and the processing logic can remain in COBOL on the Web server.

In this scenario, you create your Web interface using one of many popular authoring tools. Then you write a Common Gateway Interface (CGI) program that can read CGI variables submitted by the client to the server. This program can launch your COBOL application or it can be a COBOL program itself. You can write it using ACUCOBOL-GT or any other language you choose. If you write the program in ACUCOBOL-GT, you do not have to UNSTRING the CGI variables in the program, because ACUCOBOL-GT takes care of this for you through special "IS EXTERNAL-FORM" syntax.

By default, your CGI program reads and writes HTML content for use in standard HTTP browsers and mobile devices. But using configuration variables, you can associate your program with the MIME content type for WML so that data can be displayed on WML-based devices as well.

Once you build a Web front end and write a CGI program, your customers or users can then visit your Web site and gain instant access to your COBOL application running on the server.

Note that CGI programs are inherently stateless—that is, they do not store information about previous browser actions. If you require a persistent connection to the browser, you can achieve this by adding pointers and cookies to your CGI program, or you may choose a different method.

This option runs on any platform where ACUCOBOL-GT runs, but it also requires the most coding. You can employ the CGI method wherever a user interface via DISPLAY/ACCEPT statements is *not* used. This includes batch processes, processes that use socket routines to communicate with an external UI, component adapter technology processes, BEA Tuxedo processes, and processes launched via AcuConnect in distributed processing mode, to name a few. The CGI method is described in **Chapter 4**.

Web Runtime

Another way to give end users access to your applications on the Web is to provide runtime services through the ACUCOBOL-GT Web Runtime.

Using this approach, you set up a Web site and embed a link to your ACUCOBOL-GT application. Because the Web runtime is freely distributable (in accordance with the terms and conditions of your Micro Focus ***extend*** license agreement), you embed the runtime in the link as well by designating the URL of our Web site in your HTML coding. Users can then visit your site and click a link to launch the program. If the Web browser detects that users do not yet have a runtime installed on their machine, it automates the install process, with the users' permission, and then launches the COBOL program locally.

The Web runtime is available only on supported Windows machines, but it gives users access to programs or data hosted on other platforms using AcuServer and AcuConnect.

The ACUCOBOL-GT Web Runtime is geared for Internet Explorer environments. It relies on ActiveX technology and does not run on any current versions of Netscape.

Chapter 5 describes this option in detail.

2.1.2 Other Internet Solutions

Following are some of our solutions for harnessing the Internet in a secure TCP/IP networking configuration:

- AcuServer can be used to provide access to Vision data over the Internet.
- AcuConnect can be used to provide access to server-resident COBOL programs over the Internet, even programs that are distributed across a number of different servers.
- The AcuXDBC™ interface can be used to give users of Windows applications access to Vision data over the Internet. AcuXDBC is combined with AcuXDBC Server for remote processing of SQL requests.
- The Acu4GL® interface and the AcuSQL® precompiler can be used to provide access to relational databases over the Internet.
- AcuXML can be used to provide access to XML documents over the Internet.

All of our technologies are designed to work in TCP/IP networks. Because the Internet is just a large TCP/IP network, you can use these same proven technologies in Internet deployments. **Chapter 6** provides details.

2.2 Helping You Decide

Many factors go into deciding your Internet implementation. In this section, we present some questions to think about as you proceed. In the chapters that follow, we use examples to demonstrate how these questions will affect your decision on how to deploy Internet- or Web-based applications.

How much do you want the user to do?

Simply visit your site?	→	Write a Web interface to your application then write a small CGI program to handle the communications between the interface and the program on the server; embed the Web runtime or Web thin client on your Web page.
Download and install a free runtime?	→	Include the standard thin client on your Web page.
Click a link?	→	Create a hyperlink to the Web runtime or Web thin client on your Web page; or set up the runtime as a helper applicaiton or viewer in the browser.
Purchase a fully-licensed runtime?	→	Set the runtime up as a helper application or viewer in the browser.

How much programming are you willing to undertake?

Perform no new programming?	→	Use the Web runtime or possibly Web thin client. (If you're already using the thin client, no new programming is required.)
Reengineer the application to distribute the processing among several machines?	→	Use AcuConnect.
Write a markup language front-end and a small CGI program in COBOL?	→	Use our COBOL CGI solution.
Write an entirely new Internet program?	→	Write it in ACUCOBOL-GT. You can feed COBOL data to browsers using our CGI syntax, and you can embed Internet browsing into your program using our WEB-BROWSER control.

Do you need to provide remote access to programs, data, or both?

Programs	→	Use any of our thin client or AcuConnect solutions.
Data	→	Use AcuServer, Acu4GL or AcuXDBC.
Programs and data	→	Use a combination of these technologies.

Do you want to provide Web access or more private access?

Web access	→	Use our Web thin client, COBOL CGI solution, or Web runtime.
More private access	→	Use AcuServer, AcuConnect, or thin client.

How is your application designed?

Does your application perform lots of file I/O?	→	Distribute the processing via AcuConnect or offload all the processing onto the server via the thin client or Web thin client.
Does the executing program require client computer access?	→	Use the Web runtime or a fully-licensed runtime.
Does the program need to execute if a network is unavailable?	→	Use the Web runtime or a fully-licensed runtime.

Do you require a persistent connection to the server (i.e., Does the server need to remember the last action from the browser)?

Yes	→	Use the thin client, Web thin client, Web runtime with AcuConnect, or add pointers and cookies to your CGI program to store browser information. (CGI programs are inherently stateless.)

How frequently do you update your programs and data?

Frequently	→	Use a server-processing solution like the standard thin client, Web thin client, and/or AcuConnect, or a remote file server like AcuServer.

What are the operating environments of your user population?

Windows clients and servers?	→	You may use any of our Internet solutions.
UNIX clients and servers?	→	Use our COBOL CGI solution. Use our other technologies such as AcuServer, AcuConnect, and Acu4GL in Internet deployments.
Windows clients and UNIX servers	→	You may use any of our Internet solutions.

3 Using the Thin Client to Launch Web Applications

Key Topics

3.1 What Is the Thin Client?

Our thin client technology is an innovation that lets you display the user interface portion of your server-based application on Windows graphical display hosts.

The thin client technology is designed for two main purposes:

- To allow ACUCOBOL-GT® programs running on a UNIX or Windows server to present a full Windows graphical user interface (GUI) on PCs networked with TCP/IP. To present a graphical user interface, you must convert the application screens to graphical, if you have not done so already. Character-based user interfaces are also supported, with some restrictions. In either case, your application stays in one piece on the UNIX or Windows server.

- To allow both UNIX and Windows users to enjoy the benefits of centralized application maintenance and to adopt the performance characteristics of a "thin" architecture, reducing the total cost of ownership (TCO). Many applications perform better when deployed in a thin fashion compared to other networking techniques such as remote file access ("fat clients") or distributed processing. This is because thin client configurations execute COBOL programs on the server where data access is local.

Although designed for local and wide-area TCP/IP networks, our thin client solution is well-suited to low-bandwidth, high-lag connections like the Internet because it eliminates the file I/O occurring over the network.

Sample Scenario

Your inventory program is UNIX-based. However, you know that many of your users typically work at Windows PCs or laptops, rather than UNIX workstations. On the road, sales people often connect to your application from their hotel rooms at low data speed connections.

The ACUCOBOL-GT Thin Client provides a way for you to keep your UNIX program, yet enable users to work in a graphical Windows environment. Users can simply visit your Web site to launch the program. They never know that they are running a UNIX program because all they see is the Windows UI.

Because processing is done on the server, your users don't have to be running especially powerful machines to be able to work efficiently with your program. The thin client's low bandwidth abilities make it especially desirable for users who are out of office and using cellular connections to laptops.

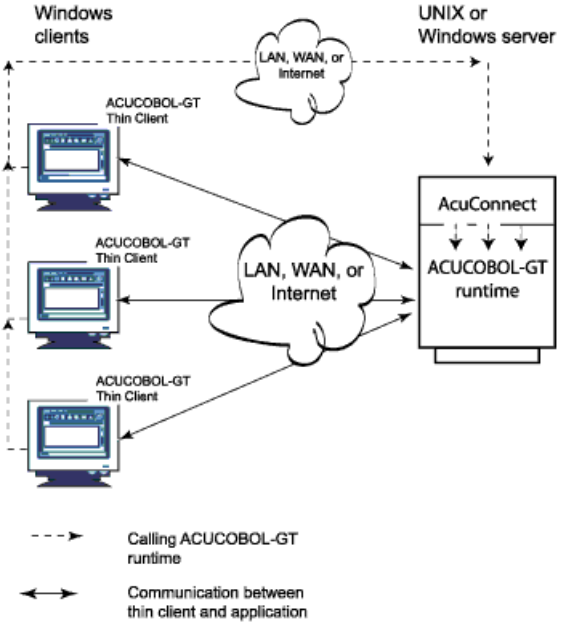
If the data file for the inventory program is remote to the application, you can use the AcuServer® file system interface to give your customers and staff access to that file.

3.2 How the Thin Client Works

In a thin client configuration, your application is composed of two logical layers: a user interface (UI) layer on the display host (client) and a COBOL layer on the application host (server). The UI layer handles screen, mouse, and keyboard activity, and the COBOL layer performs application processing. Because no application components are required on the client (unless you use ActiveX controls), it is considered to be "thin."

Rather than forcing you to split your application into client and server components, the ACUCOBOL-GT runtime has been split so that your existing application can be displayed on the client. The portion that is installed on the client is known as the "ACUCOBOL-GT Thin Client". (Currently, the thin client runs only on Windows machines.) The full ACUCOBOL-GT runtime is installed on the server.

To function, the split runtime makes use of the AcuConnect® application server software. The role of AcuConnect is to listen for requests from clients to launch the ACUCOBOL-GT runtime on the server. You license the server runtime for a specific number of concurrent users.



Thin client architecture

Together, the ACUCOBOL-GT Thin Client, the ACUCOBOL-GT runtime, and AcuConnect are the enabling technologies that make up our thin client solution.

For a comprehensive look at the thin client technology, please refer to the *AcuConnect User's Guide*.

3.3 Thin Clients and the Internet

Our thin client can be deployed over the Internet just as easily as a local- or wide-area network. There are three ways to accomplish this:

1. You can specify the name and port number or IP address of the server you are accessing when you enter thin client command line parameters. See **section 3.4** for details.

2. You can create a command line file that contains all the information needed by the ACUCOBOL-GT Thin Client to launch your application. You can then create a link on your Web page to this file, and if your license permits it, to a self-extracting archive for the thin client. Users can then download the thin client from your Web site and click a link to launch your program. See **section 3.5** for details on this method.

3. You can place an ActiveX version of the thin client, known as the ACUCOBOL-GT Web Thin Client, on your Web page. When set up properly, the Web thin client automates the process of download and install for end users, and it automatically invokes your application as well. See **section 3.6** for details.

Note: Please review the license agreements of any third-party product that may be invoked when you are accessing the server from a remote location through the ACUCOBOL-GT Thin Client. For instance, if you will be accessing a Windows server through a virtual private network, you should review Microsoft's end user license agreement.

3.4 Specifying an Internet Address on the Command Line

The ACUCOBOL-GT Thin Client can work in an Internet environment independent of the Web and Web browsers. After all, the Internet is just a big TCP/IP network. When end users invoke the ACUCOBOL-GT Thin Client in an Internet environment, they type command line parameters that specify

the server name, optionally the port number of the server they are accessing over the Internet, the name of the application, and any additional runtime command line parameters. For example:

```
acuthin bigserver.acucorp.com:5632 myprog1 10 20 30
```

Note that clients must have a live Internet connection when they execute this command, and the server name that they enter must be resolvable by the Internet name server used by their service provider. (The Internet name server then resolves the name with its IP address.)

If the server name is not exposed to Internet name servers, end users can enter the explicit IP address on the ACUCOBOL-GT Thin Client command line as in the following example:

```
acuthin 128.110.121.42:5632 myprog1 10 20 30
```

If the server is through a Virtual Private Network (VPN), the user must connect to the network before entering the **acuthin** command.

Note: If desired, users can set up a Windows shortcut so that they do not have to enter a command line in order to launch the thin client.

3.5 Providing Thin Client Links on the Web

If you want to give users access to your application on the Web, one way to do this is to distribute the thin client software on your Web page (license agreement permitting), and then provide a link to a command line file that launches your application. Users visit your page, download and install the thin client software, then click the command file link to run your program.

To provide this method of Web access, do the following:

1. Install your COBOL application on your application server host. This machine must also contain the ACUCOBOL-GT runtime, AcuConnect, and several additional components. These are listed in the *AcuConnect User's Guide*. Note that your application does not need to reside on the same host as your Web Server.

2. Create a thin client command line file that contains all the information needed by the ACUCOBOL-GT Thin Client to launch your application. The format of this file is described in **section 3.5.1, "Thin Client Command Line Files."**

3. Place the command line file and, license agreement permitting, the self-extracting archive for the thin client on your Web server. The archive name is "atcinst.exe". Create links to these files from an appropriate Web page. See **section 3.5.2, "Using Anchor Tags,"** for instructions on creating links.

4. To ensure that your thin client will be recognized across a wide variety of browsers, configure your Web server to generate the HTTP "Content-type" response header field containing the MIME type "application/vnd.acucorp.thincommandline" for the extensions ".atc" and ".acutc". This is particularly necessary if your end users have newer versions of Windows (such as Windows XP, SP2). Refer to your Web server documentation for instructions on adding a MIME type.

Once you have followed these steps, instruct your users to visit your Web page and click the thin client link to download and install the thin client. It is a self-extracting archive that will guide your users through the installation process. (If you prefer, your users can install the ACUCOBOL-GT Thin Client on their machines from any of our media, or they can download the thin client from the support section of our Web site, **www.microfocus.com**.)

Once they have the thin client on their machines, users can visit your Web site anytime and click the program link to launch your thin client application. The program runs on your application server host and the display occurs in an application window on the users' machine, outside of their browser.

Note: If you want to automate the process for end users, you can write an ASP, JSP, VB, or perl script that uses cookies to check for thin client installations. The user can activate the script by clicking a link on your Web site. The cookie can search for thin client installations on the user's machine, verify that the version is current if one is located, then launch the server application automatically. If a thin client installation is not found or is outdated, it can prompt the user to download new thin client software.

3.5.1 Thin Client Command Line Files

When users install the ACUCOBOL-GT Thin Client on a Windows display host, the ".atc" and ".acutc" file extensions are automatically associated with the thin client executable ("acuthin.exe"). As a result, when your browser downloads one of these files, it uses these associations to invoke "acuthin.exe". To establish the command line parameters that the executable should use, including the name and location of the server application it should launch, you must create a thin client command line file.

Format the thin client command line file as you would any ACUCOBOL-GT configuration file, with each entry on its own line. You can include any of the following variables in the file:

Variable name	Description
atc-server	the name or IP address of the server to connect to (a required variable with no default value)
atc-port	the TCP port number to use to grant access to thin client applications (default value is 5632)
atc-runtime-options	runtime options passed to the runtime via AcuConnect (no default value)
atc-alias	an alias that identifies a COBOL application on the server (a required variable with no default value)
atc-user	the username with which you want to log onto the AcuConnect server
atc-password	the password assigned to this username for granting access to AcuConnect.
atc-splash-screen	when set to "off" (0, false, no), an option that disables the thin client start-up splash screen
atc-cobol-args	COBOL arguments passed to your COBOL program via AcuConnect (no default value)
atc-trace	trace options passed to the runtime via AcuConnect (default "0", meaning no trace). Equivalent to the "-t" command-line option.

Note that comment lines (preceded by the "#" character) are allowed in this file, as they are in other configuration files.

When you are done, save the file with an ".atc" or ".acutc" extension.

The contents of the ".atc" or ".acutc" file are interpreted by the thin client as the following command line:

```
acuthin <atc-server>[:<atc-port>] [atc-runtime-options]
        <atc-alias> [<atc-cobol-args>]
```

3.5.2 Using Anchor Tags

Use the HTML Anchor tags to create two separate links: one to your COBOL application and another to the self-extracting archive of the thin client runtime. The HTML Anchor tags, <A> and </A>, are closed elements that, when combined with the HREF attribute, highlight text or images, making them clickable. When users click on a highlighted item on your Web page, they are transferred to the linked document.

In the case of the thin client runtime, when the user clicks the highlighted item, the runtime automatically downloads to the user's machine. When the download is complete, the self-extracting archive guides the user through an automated installation process.

In the case of the thin client command line file, when the user clicks the highlighted item, the thin client application is automatically invoked.

To turn text into a hypertext anchor, enclose the clickable text in the Anchor tags. For example, enter the following HTML command onto your Web page:

```
<A HREF="atcinst.exe">Click here to download the thin client</A>
<A HREF="myprog.atc">Click here to run the application</A>
```

where "atcinst.exe" is the name of the self-extracting archive for the thin client runtime, and "myprog.atc" is the name of the command line file that you created for your thin client application.

The HREF attribute is used within the starting anchor tag to specify the document to be linked (or retrieved). (See any commercially available HTML text for more information on anchors and hypertext links.)

3.5.3 Security and the ACUCOBOL-GT Thin Client

Security is a key consideration as you set up your thin client environment. By its very nature, in a thin client environment traffic is flowing into and out of your network.

You should consider setting up a firewall to limit access to your data and enforce your organization's access control policy. When you set up a firewall, you'll need to indicate the "port number" through which the thin client, AcuConnect, and the ACUCOBOL-GT runtime can communicate. The default port number for both the thin client and AcuConnect is 5632.

You can indicate an alternate port number for AcuConnect using the configuration variable, ACURCL_PORT. To indicate a port number for the thin client, set ATC_PORT in the thin client command line file.

If you want to encrypt data before transmitting it across the network, you can use the AGS_SOCKET_ENCRYPT runtime configuration variable on the server. You can also use the ENCRYPTION_SEED variable to initialize an industry-standard 128-bit AES encryption algorithm.

3.6 Using the ACUCOBOL-GT Web Thin Client

The ACUCOBOL-GT Web Thin Client is a special 32-bit version of the thin client that is based on Microsoft's ActiveX technology. It is itself an ActiveX control that you can embed on your Web page. It makes your existing thin client applications accessible through browsers that support ActiveX, particularly Microsoft Internet Explorer. While the Web thin client runs only on Windows machines, it can access files or run programs on UNIX and other platforms using AcuConnect.

With the Web thin client, users who visit your site are automatically provided the software they need to run your application on the remote host (providing that they grant permission first). Software installation and program invocation are transparent.

Here is the general process:

1. The end user visits a page on your Web site.

2. The Web browser software looks for the Web thin client on the user's machine. If it locates the Web thin client, it runs it, and the Web thin client launches the program on the server.

   If the Web browser cannot find the Web thin client locally, it downloads the file from the specified location and asks the user "Do you want to install and run ACUCOBOL-GT Web Thin Client?" This message may appear in a dialog box, or in newer versions of Windows (such as Windows XP, SP2), it may appear in an "Information Bar" at the top of the browser window. To assure the user that the installation is safe, the Web thin client is supplied as a cabinet (CAB) file with a digital signature from Acucorp.

   If the user grants permission, the Web thin client is sent to the client machine, where it automatically installs itself and launches your program on the server. The server then "projects" the user interface back onto the client.

Note: For automatic installation to be performed, you program your Web page with the CODEBASE attribute of the OBJECT tag as instructed in **section 3.6.8**.

3.6.1 Windowing Options

With the thin client, even though your program executes on the server, the user interface is "projected" onto the client. You can choose whether the UI displays inside the user's browser window or in a separate application window. How and where your program starts depends on the HTML commands that you use to include it on your Web site. (Please refer to **section 3.6.7** for details.) Listed below are some things to consider when choosing a windowing method.

Inside the User's Web Browser

In this case, your COBOL program starts inside the user's browser window. Nearly all of the ordinary functions of your COBOL program are available to your user and your program has access to library routines that can be used to communicate with the browser.

In this mode, users also have access to browser functions like Forward, Back, and Search, but when they use these functions, the COBOL program terminates. To avoid losing data, the COBOL program should be designed to handle the closing action gracefully within ten seconds. If users return to the page from which they launched the ACUCOBOL-GT application, it reloads like any Web page.

Note that because of a Microsoft child window restriction, applications running inside a browser window cannot display a main window menu bar as you or your users might expect. To work around this restriction, you can program your application's menu functions to be accessed from a toolbar or a pop-up menu that is activated with the right mouse button.

In a Separate Application Window

In this mode, your COBOL program starts in a window separate from the browser, making it look the same as it would if it were launched locally. All the functions of your original program are available to users, including a main window menu bar.

Although your program cannot access the library routines used to communicate with the browser in this mode, users can still access functions like Forward, Back, and Search simply by activating the browser window. However, when the user selects a browser function, the Web thin client object terminates. This is because the thin client object executes as an object related to the HTML page, even when running in a separate window.

3.6.2 How Your Program Executes

With the Web thin client, your program executes on the server, and only the user interface displays on the client.

Note: The Web thin client only supports a single instance.

The recommended method of executing the ACUCOBOL-GT Web Thin Client is with the <OBJECT> tag (see **section 3.6.8**).

The browser loads the Web thin client (which you recall is an ActiveX control) and feeds it the data coming from the Web server. In this context there is no concept of a command line.

To pass “command line” style parameters to the ACUCOBOL-GT Web Thin Client, you use the AcuCommandLine property of the OBJECT tag when you invoke the application in your Web page.

The Web thin client only supports a single instance. Internet Explorer 6 introduced tabs that allow more that one web page to be displayed in a single browser. IE 6 & 7 load all tabs into the same process. IE 8 loads the first couple of tabs in separate processes but eventually (depending on the client computer's resources) will start loading subsequent tabs in a previous tab's process. So it may be possible to run more than one Thin Client application in a single Internet Explorer window using version 8.

Whenever a user attempts to run a second instance of the Web Thin Client in a process, the second instance of the control will detect this condition and enter a "blocked" state. This means the acuthin DLLs haven't been loaded and the Thin Client application is not executing on this page/tab. The Web Thin Client control has properties to detect this state and give the developer choices in how to handle it (see **Section 3.6.8.3, “Object interface for the Web thin client”**).

3.6.3 Browser Versions Supported by the Web Thin Client

The Web thin client is designed for browsers that support ActiveX controls. We have confirmed that the Web thin client runs on Internet Explorer Versions 5.5 Service Pack 2 and later. Browsers that do not support ActiveX controls cannot use the Web thin client. Currently, this includes Netscape browsers, and some earlier versions of Internet Explorer.

3.6.4 Deploying Applications via the Web Thin Client

To deploy your application on the Web via the Web thin client, you (the developer) have three tasks:

1. Set up a Web site. (See **section 3.6.5** for details.)
2. Consider whether you need to make any coding adjustments to your program. (See **section 3.6.6** for coding considerations.)
3. Update your Web page to invoke your thin client application. To do this, you typically embed the URL of the Web thin client along with the URL of your application on your Web page using an <OBJECT> element and the CODEBASE attribute. (See **section 3.6.7**.)

Once your work is done, the user has only one task:

1. Visit your Web site. The Web thin client will install itself and launch your program automatically.

   Note that two dialog boxes may be displayed in the process: one containing a security message if required by the security setting of the user's browser (see **section 3.6.12** for more information), and the other containing a click-wrap license agreement from Micro Focus.

3.6.5 Setting Up a Web Site

Setting up a Web site is probably the most time-intensive portion of this method, but it is not very difficult to do. Appendix A gives general information about setting up a Web site, including information on Web servers, posting a site, and promoting a site. Many different tools are available to help you create a Web page quickly and easily. Refer to **Appendix A** for guidelines.

3.6.6 Coding Considerations

If you plan to have your application run inside the user's browser window, it cannot display the main window menu bar to which you may be accustomed. (This is a Microsoft child window restriction.) In this case, you will need to

program your application's menu functions to be accessed from a toolbar or a pop-up menu that is activated with the right mouse button. You can avoid this restriction by having your application run in its own separate window. You can specify this as a parameter of the <OBJECT> tag or by setting the AcuEmbedded property supplied with the object interface. See **section 3.6.8** for more information.

3.6.7 Updating Your Web Page to Invoke Your COBOL Application

To invoke your COBOL application and the Web thin client from your Web page, you use the HTML **<OBJECT> tag**. The <OBJECT> tag allows you to invoke both of these items with a single HTML element. If you add the CODEBASE attribute, users can automatically download the control the first time they access the application, and they can get updates automatically. The browser checks the *CLASSID* property and downloads a new version of the control, if one is available.

In addition, we have developed an object interface containing several properties and methods for communicating with browsers. If desired, you can implement all of the properties in the object interface as attributes of the <OBJECT> tag.

Advanced users may choose to instantiate their application with the <OBJECT> tag, and then write scripts with the object interface to invoke the application. Please note that scripting the Web thin client may require changes to your end users' security settings. (See **section 3.6.12.2** for more information.)

3.6.8 Using the <OBJECT> Tag

When authoring Web pages to launch your application, you can use the <OBJECT> tag to invoke the Web thin client and start your program at the same time. For example, you could include the following in your HTML code:

```
<OBJECT ID="AcuThinAX" WIDTH=512 HEIGHT=384
   CLASSID="CLSID:087C768D-64C1-4AC1-845D-4589B4B2C24E"
```

```
   CODEBASE="http://www.acucorp.com/support/downloads/acuthinax/
   acuthinax800.cab#version=8,0,0,900">
   <PARAM NAME="SRC" VALUE="http://yourserver/yourdirectory/
   yourcommandlinefile.acutc">
</OBJECT>
```

where the following values are described as:

ID	Optional. The name of the instance of the object. This name is only used for scripting the object interface. It is a user-defined value to which you refer in your script.
CLASSID	The GUID (globally unique identifier) assigned to the ACUCOBOL-GT Web Thin Client control, specifically this value: `CLSID:087C768D-64C1-4AC1-845D-4589B4B2C24E`
CODEBASE	The CODEBASE URL, from which the Web thin client can be downloaded and installed automatically by end users, specifically: `http://www.acucorp.com/support/downloads/acuthinax/acuthinax###.cab` where *###* is a 3-digit segment that identifies the cab file version. You can also append the version information that applies to the control, including a build number, by adding it as described in subsequent sections. Although the CODEBASE attribute is optional, using it is now a common practice among software vendors to provide access to controls in this way. This allows you to distribute the control easily. If you do not use the CODEBASE attribute, you must direct users to the download page on the Acucorp Web site. Or, with a proper written license agreement, you may provide the control on your own distribution media or Intranet site.
HEIGHT	Optional. The height (in pixels) of the object's window. Use to define the area within the browser window that the application object will occupy.

WIDTH	Optional. The width (in pixels) of the object's window. Use to define the area within the browser window that the application object will occupy.

By default, the application appears in the browser window, using the HEIGHT and WIDTH attributes, if provided, to define the area that the application occupies.

If desired, you can add any of the properties and methods of the Web thin client object interface as parameters of the <OBJECT> tag. To do so, add "PARAM NAME=" followed by the property name from the object interface enclosed in quotes. You then supply the VALUE attribute.

For example, if you want to have your application appear in its own window rather than in the browser window, set the AcuEmbedded property of the object interface to "FALSE" as follows:

```
<PARAM NAME="AcuEmbedded" VALUE="FALSE">
```

The following example illustrates how you might display your application in a separate window.

```
<OBJECT ID="AcuThinAX" WIDTH="512" HEIGHT="384"
   CLASSID="clsid:087C768D-64C1-4AC1-845D-4589B4B2C24E"
   CODEBASE="http://www.acucorp.com/support/downloads/acuthinax/
acuthinax800.cab#version=8,0,0,900">
   <PARAM NAME="AcuEmbedded" VALUE="FALSE">
   <PARAM NAME="SRC" VALUE="http://yourserver/yourdirectory/
yourcommandlinefile.acutc">
  </OBJECT>
```

Note that there are two ways to specify command line parameters for your program: you can use the SRC property or the AcuCommandLine property of the object interface, but you never use both at the same time. The original example in this section showed the SRC form:

```
<PARAM NAME="SRC" VALUE="http://yourserver/yourdirectory/yourcommandlinefile.acutc">
```

This points to a separate file containing the command line parameters that you wish to invoke on start-up. (See **section 3.5.1** for more details on creating command line files.) If you already have such a file that you're using for the thin client, then this will likely be your preferred method.

Alternatively, you can use the AcuCommandLine form:

```
<PARAM NAME="AcuCommandLine" VALUE="myserver.mysite.com:5632 myalias">
```

The advantage of the AcuCommand Line property is that you can specify the command line directly in the <OBJECT> tag without maintaining a separate file.

For more information on the object interface and its properties, including SRC and AcuCommandLine, refer to the subsequent sections.

3.6.8.1 How the <OBJECT> tag works

The CODEBASE parameter indicates to the browser where to look for the Web thin client if it is not installed on the target system. Internet Explorer automatically offers to download and install the Web thin client, prompting most users to accept the digital signature provided by Acucorp.

Note: If the user's security settings are high or customized to prohibit ActiveX controls, the user cannot install the Web thin client. If the user's security settings are low, the control installs without confirmation from the user.

In the HTML document, you can introduce the Web thin client with the <OBJECT> tag and then supply a script to invoke the object through either a window event or a push-button event. For information on using the object interface, see the next section. However, you need not use scripting in order to invoke the control with the <OBJECT> tag.

3.6.8.2 Version number of Web thin client

The name of the CAB file identified in the CODEBASE URL is version-specific. Therefore, "acuthinax800.cab" always contains the Version 8.0.0 Web thin client. However, you can implement your CODEBASE URL with a version string that refers to the version number as well as the build number.

If, for example, you deploy your Web site with the following CODEBASE value:

```
http://www.acucorp.com/support/downloads/acuthinax/acuthinax800.cab
```

a control associated with the specified CLASSID will be used (if found), regardless of version. If no associated control is available on the client machine, the CAB file will be downloaded, installed, and executed.

On the other hand, if your Web site contains the following value:

```
http://www.acucorp.com/support/downloads/acuthinax/
acuthinax800.cab#version=8,0,0,900
```

users with an earlier version of the control will automatically be prompted to download the new version the next time they visit your Web site. The convention for the Web thin client control is therefore:

```
acuthinaxMmr.cab#version=M,m,r,b
```

where *M,m,r,b* represent the Major, minor, release, and build numbers of the particular version of the control. This allows you to determine which version of the control is available to your end users.

To determine the version and build number of a Web thin client control, you can use the AboutBox method available in the object interface. See the next section for more information. You can also obtain the version number by following these steps:

1. Locate the file, “acuthinax.ocx”, for the version you want to deploy.
2. In Windows Explorer, right-click on the control and select **Properties**.
3. Select the Version tab and view the value in the File Version field.

3.6.8.3 Object interface for the Web thin client

The Web thin client’s object interface consists of the following methods and properties. Some are bi-directional (B) and others are read only (R). The component also has a default property (D).

Note: Using the object interface is optional. The <OBJECT> tag exposes the same properties (but not methods). The object interface is available for the advanced user who prefers scripting and wishes to use the available methods. However, please be aware that support for specific scripting languages and their implementations should be obtained from the appropriate vendor.

Methods

AcuIsActive	R
AcuExecute	R
AcuShutDownAx	R
AcuGetLastError	R
AboutBox	R

Properties

AcuEmbedded	B
AcuShowLogo	B
AcuCommandLine	B
SRC	BD
IsBlockedInstance	R
BlockedInstanceAction	B
BlockedInstanceMsgText	B

Syntax

The following sections describe the methods and properties for the Web thin client control. Each method or property name is listed, followed by the type of the input parameter. (The type is shown in parentheses.) Finally, the input value and the output value (returned value) are shown, with acceptable values indicated in square brackets "[ ]". Empty brackets indicate that the value is not limited, or takes no input.

Examples of limited values include Booleans, which can be only "True" or "False". The output of the AcuIsActive method uses this limited value.

You will notice that the syntax examples in the following sections refer to "AcuThinAX". This refers to the HTML object ID assignment of the Web thin client control given in the <OBJECT> tag. For example:

```
<object classid="clsid:087C768D-64C1-4AC1-845D-4589B4B2C24E"
ID="AcuThinAX" width="251" height="144">
```

AcuIsActive

Returns the status of the Web thin client. AcuIsActive takes no input parameters.

Parameter Type	Input	Output
( )	[ ]	[TRUE, FALSE]

The return value is of the data type BOOL:

- TRUE = the thin client is executing
- FALSE = the thin client is not executing

For example:

```
If AcuThinAX.AcuIsActive() = TRUE
```

AcuIsActive may be executed any time after the Web thin client (in this example, “AcuThinAX”) has been invoked.

This method is useful for determining when a loaded thin client is running or not. This method is not userful for determining if an invoked web thin client will be blocked due to an already running instance of the web thin client. For this situation, use the IsBlocked instance property.

AcuExecute

Starts the Web thin client and returns the result of the action.

Note: If you invoke your application directly by specifying the SRC or AcuCommandLine property in the <OBJECT> tag, you do not need to call AcuExecute. Your application will start automatically.

The AcuExecute method takes no parameters.

Parameter Type	Input	Output
()	[ ]	[0, -4, -5, >0]

The return value is of data type LONG.

Value	Explanation
0	Success, Web thin client started.
-4	The Web thin client is already running.
-5	The acuthin DLL could not be found, or the acuthin DLL is the wrong version.
>0	Unexpected error.

For example:

```
Return_value = AcuThinAX.AcuExecute()
```

Do not invoke this method until you have set either the SRC or AcuCommandLine property to specify the command line file to use; otherwise, AcuExecute will not have the information necessary to start your application.

AcuShutdownAx

This is an optional method that forces a shutdown of a specific Web thin client instance. For example:

```
AcuThinAX.AcuShutdownAx()
```

shuts down the thin client instance known as “AcuThinAX”, if it is running.

AcuShutdownAx terminates the COBOL application invoked by the Web thin client. In general, you need not use it because the Web thin client either terminates as a result of the COBOL program execution, or because the browser either displays another URL or closes. If, for some reason, you want to force shutdown of the Web thin client instance, this method is available.

Parameter Type	Input	Output
()	[ ]	[ ]

The shutdown action assumes that the COBOL application is currently in idle mode. If the Web thin client is not idle, the browser hangs, waiting for the application to idle. This method takes no parameters, and there is no return value for this function.

AcuShutdownAx implicitly terminates the Web thin client when you terminate the COBOL application you are running. So, when you execute this method, the Web thin client is terminated, closing files properly, even though data that was not stored permanently is lost.

AcuGetLastError

Returns the last known error code.

Parameter Type	Input	Output
()	[ ]	[ ]

This method takes no parameters, and the return value, which is the error code, is of data type LONG. See “AcuExecute” for more information.

For example:

```
AcuThinAX.AcuGetLastError()
```

AcuGetLastError may be executed any time after the Web thin client has been invoked.

AboutBox

Displays a dialog box that presents version information about the Web thin client.

Parameter Type	Input	Output
()	[ ]	[ ]

There is no return value.

For example:

```
AcuThinAX.AboutBox()
```

AboutBox may be executed any time after the Web thin client ("AcuThinAX" in this example) has been invoked.

AcuEmbedded

Determines whether the Web thin client should run in an independent window or as a frame within the document.

Parameter Type	Input	Output
(BOOL)	[TRUE, FALSE]	[TRUE, FALSE]

The AcuEmbedded property accepts a BOOL as a value. When this property is set to "TRUE", the default, the window appears embedded in the browser document. Set it to "FALSE" when you want the window to appear independently, outside the browser.

For example:

```
AcuThinAX.AcuEmbedded = TRUE
if AcuThinAX.AcuEmbedded
```

The AcuEmbedded property is available any time after the Web thin client has been invoked. Its contents are read and applied to the Web thin client when AcuExecute is invoked.

Note that if you use the HEIGHT and WIDTH attributes of the <OBJECT> tag and you set AcuEmbedded to "FALSE", the Web thin client's logo screen will occupy that area on the HTML page while the application runs in its own window.

AcuShowLogo

Determines whether to display the ACUCOBOL-GT Web Thin Client logo when you invoke your application.

Parameter Type	Input	Output
(BOOL)	[TRUE, FALSE]	[TRUE, FALSE]

For example:

```
AcuThinAX.AcuShowLogo = TRUE
if AcuThinAX.AcuShowLogo
```

To use this property, specify it as part of the <OBJECT> element where you invoke the application. The default is TRUE.

AcuCommandLine

The command line that you would normally specify for the **acuthin** executable. See the *AcuConnect User's Guide* for a list of valid **acuthin** command-line options. The minimum value is a server name and alias name. For example:

```
myserver.mysite.com:5632 myalias
```

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

Do not specify this property if you are using SRC.

SRC

Contains a URL (in HTTP notation) for the thin client command line file to load. (See **section 3.5.1** for a discussion of command line files and how to create them). SRC is the default property of the Web thin client ActiveX control. Do not specify this property if you are using AcuCommandLine.

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

The following example demonstrates using the <OBJECT> element to specify the program for execution:

```
<OBJECT
  CLASSID="CLSID:087C768D-64C1-4AC1-845D-4589B4B2C24E"
  ID="AcuThinAX" width="512" height="384">
  <PARAM NAME="AcuEmbedded" VALUE=”TRUE”>
  <PARAM NAME="SRC" VALUE="http://server.acucorp.com/location/commandlinefile.acutc">
</OBJECT>
```

The SRC property is intended for HTTP URLs only. However, you may also address local files if the path is prefixed with “file://” rather than “http://”, as shown below. Note that this is also considered URL notation.

For example:

```
<OBJECT
  CLASSID="CLSID:087C768D-64C1-4AC1-845D-4589B4B2C24E"
  ID="AcuThinAX" WIDTH="512" HEIGHT="384">
  <PARAM NAME="AcuEmbedded" VALUE=”TRUE”>
  <PARAM NAME="SRC" VALUE="file://c:/webdemo/commandlinefile.acutc">
</OBJECT>
```

When you use the SRC property to call a file, the downloaded file is given a temporary name and is stored in the current master temporary directory for the user, as specified in Windows. Note that the Web thin client automatically captures the temporary name and uses it internally; however, the internal name is not exposed.

IsBlockedInstance

Indicates whether or not the Web thin client has detected that it is a second instance and is blocked.

IsBlockedInstance is Read Only.

Parameter Type	Input	Output
(BOOL)	N/A	[TRUE, FALSE]

The property is of the data type BOOL.

TRUE = the Web Thin Client control is "blocked" and will not run the Thin Client application.

FALSE = the control is the first instance and will run the Thin Client application.

For exampe:

```
If AcuThinAX.IsBlockedInstance = TRUE
```

The IsBlockedInstance property may be checked any time after the Web thin client (in this example, "AcuThinAX") has been invoked.

BlockedInstanceAction

Indicates whether or not the Web thin client has detected that it is a second instance and is blocked.

IsBlockedInstance is Read Only.

Parameter Type	Input	Output
SHORT)	[0, 1, 2]	[0, 1, 2]

The property is of the data type SHORT

Value	Description
0	Display nothing. For use in scripting.

Value	Description
1	Display the default message, "ACUCOBOL-GT Web Thin Client may only be loaded once." in the upper left corner of the client area. This is the default setting.
2	Display a custom message in the upper left corner of the client area.

For exampe:

```
AcuThinAX.BlockedInstanceAction = 1
Return_value = AcuThinAX.BlockedInstanceAction
```

BlockedInstanceMsgText

Contains the text that will display when the Web Thin Client is blocked due to being a second instance and the BlockedInstanceAction property is set to 2 (custom message).

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

For exampe:

The following example demonstrates using the <OBJECT> element to specify the custom message :

```
<OBJECT
  CLASSID="CLSID:087C768D-64C1-4AC1-845D-4589B4B2C24E"
  ID="AcuThinAX" width="512" height="384">
  <PARAM NAME="AcuEmbedded" VALUE="TRUE">
  <PARAM NAME="BlockedInstanceAction" VALUE="2">
  <PARAM NAME="BlockedInstanceMsgText" VALUE="The Web Thin
Client is already loaded.">
</OBJECT>
```

3.6.8.4 Scripting with the object interface

Scripting is an optional method for invoking the Web thin client. While it is possible to invoke the Web thin client without scripting, some users may want to take advantage of all the methods and properties in the object interface.

The following VB Script example illustrates how to invoke the Web thin client object by a push button event using a script:

```
<INPUT id=button1 name=button1 type=button value=Button>
<SCRIPT LANGUAGE="VBScript">
sub button1_onclick()
   call AcuThinAX.SRC("http://www/Acucorp.com/demo/demo.acutc")
   call AcuThinAX.AcuEmbedded(1)
   call AcuThinAX.AcuExecute()
end sub
</SCRIPT>
```

Note: Only one instance of the Web thin client is allowed. You cannot have multiple instances of the Web thin client in one browser instance.

3.6.9 Licensing Considerations

The ACUCOBOL-GT Web Thin Client is licensed with a runtime license on a server running AcuConnect, licensed for the number of concurrent users anticipated.

There is no licensing requirement on the end user client machine.

3.6.10 The User's Job

To use your application via our Web thin client, end users typically have just one task: they must visit your Web site. If the user's browser supports the Web thin client and you have implemented the CODEBASE in your HTML code, users install the Web thin client automatically the first time they visit your Web site, and the Web thin client automatically launches your application.

If your customers do not already have the ACUCOBOL-GT Web Thin Client installed on their machine when they try to launch your application, they are informed that a control for the file type was not found, and they are asked if they want to install and run your program. If they respond "Yes", the Web thin client installs with no further user interaction and automatically launches your program.

Note: Users must configure the browser to enable "ActiveX controls ". To do this in Internet Explorer, they should check the appropriate box in the Security tab of the Internet Explorer Options dialog.

For most users installing the Web thin client, they will be prompted to accept the digital signature of the control. The system also displays the terms and conditions for using the ACUCOBOL-GT Web Thin Client. Users must agree to the terms of the agreement in order to use the Web thin client.

3.6.11 Troubleshooting

Users may encounter the following message when installing or using the Web thin client:

A plug-in for this file type was not found.

Users receive this message if they do not have the Web thin client installed on their machine when they try to launch your application and, for some reason, cannot install one automatically using the CODEBASE implementation. They should obtain a copy of the ACUCOBOL-GT Web Thin Client through another means (***extend*** media or Web site, for example) and install it on their machine.

You can try reinstalling the control, running **regsvr32** to register it, or accessing an HTML document where the CODEBASE attribute is correctly implemented.

3.6.12 Security and the Web Thin Client

Security for the Web thin client begins with the following provisions:

- Warning messages appear the first time users launch the Web thin client.
- We supply the Web thin client with a digital signature indicating that Acucorp, Inc., verifies the content. This enables users to run the control using a medium security setting on their Internet Explorer browser.

Since we distributes the Web thin client via the Internet, it is packaged as a signed Cabinet (CAB) file. This ensures your users that the code is safe. The CAB file contains a compressed version of the control, with information that tells Internet Explorer how to install it.

Caution: Although we have built in a number of security features, the Web thin client allows COBOL programs to call library routines, DLLs, and ActiveX controls on target machines. Because we have no control over the programs that are executed by the Web thin client, we cannot fully guarantee that they are safe.

3.6.12.1 Digital signature of Web thin client

A digital signature provides users with a way of identifying who published the software they are downloading from the Internet.

By distributing the Web thin client with a CAB file and digital signature, we ensure that most standard configurations of Microsoft Internet Explorer will accept the Web thin client application after the user responds “yes” to the Security Warning dialog when it is installed. Users who have set their browser security level to low will not see this dialog, since their browsers automatically accept digitally signed components.

Note: Users with security levels set to high cannot run any ActiveX-based controls, regardless of the digital signature.

Because the CAB installation does depend on some Microsoft files, it contains an embedded link to a file on the Acucorp Web site to ensure that it can obtain the proper files. If these files are needed, users are prompted to

install the Microsoft files; doing so may impact the download and installation time of the control, due to the file size of some of these required files.

3.6.12.2 How Internet Explorer security affects the Web thin client

Certain security settings of Internet Explorer can affect how it handles the Web thin client. The following table outlines the security settings that could have an effect on your end users, depending on their configuration of Internet Explorer.

Security Setting Option	**Low**	**Medium Low**	**Medium**	**High**
Download signed ActiveX controls	Enable	Prompt	Prompt	Disable *
Download unsigned ActiveX controls	Prompt	Disable	Disable	Disable
Initialize and script ActiveX controls not marked as safe	Prompt	Disable	Disable	Disable
Run ActiveX controls and plug-ins	Enable	Enable	Enable	Disable*
Script ActiveX controls marked safe for scripting	Enable	Enable	Enable	Disable**
Don't prompt for client certificate	Enable	Enable	Disable	Disable
Active Scripting	Enable	Enable	Enable	Disable**

* If your users have selected "Disable" for either of these settings—either via the default High security setting or a custom setting—they cannot run the Web thin client.

**If your users have selected "Disable" for either of these settings, and they are using the scripting facilities, they cannot run the Web thin client.

3.6.12.3 Security warning messages

In general, programs run with Web controls can potentially damage an end user's computer system or corrupt memory. Therefore, to help reduce the chances of this happening, the Web thin client displays warnings to end users, asking them to accept responsibility. Based on your end users' expectations, you may need to provide instructions for handling these messages.

4 Launching Web Applications Through CGI

Key Topics

4.1 What Is CGI?

If you want to deploy your COBOL applications on the Web without any configuring of the user machine, you can develop a Web interface to your application and write a CGI program on the server. CGI stands for Common Gateway Interface. It is an Internet standard that defines how a Web server communicates with an external program.

Using CGI, your application becomes immediately available to any user with a Web browser or mobile device. The end user does not require any special runtime or plug-in to interface with your application, because the CGI script handles the communication for them.

The CGI program (also known as the CGI script) can be written in any language, including COBOL. By writing it in the ACUCOBOL-GT® Development System, you can take advantage of your COBOL programming experience, as well as many other benefits.

This option gives the most flexibility and platform independence, but it requires the development of a new user interface and a modest amount of COBOL programming.

Why Write CGI Programs in ACUCOBOL-GT?

ACUCOBOL-GT has features designed to simplify CGI programming. For instance, using familiar ACCEPT and DISPLAY syntax, you can accept CGI input data and write HTML, WML, or XML output forms. It also has configuration variables that address special formatting and storage issues in this environment.

By writing CGI programs in a language you already know, you have to learn only what the Web server expects, and how it formats messages that it passes back and forth across the network.

Another advantage is that your CGI program is compiled when it is written in ACUCOBOL-GT. This not only speeds up processing, but it reduces the size of the finished product, and secures your program from anyone who might try to acquire and modify it. Your CGI program is not sent across the Internet, only the data (if any) and response (if any). Users won't know if your CGI program is written in COBOL or another language. They see only the results.

Sample Scenario

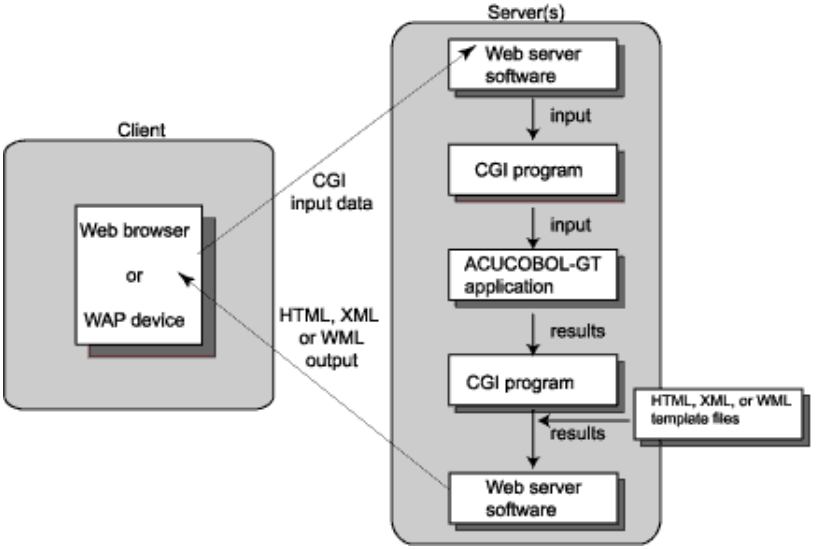
Your inventory program allows customers to place orders online. They fill out an HTML form on your Web site, which provides input to a CGI program (written in COBOL). The CGI program updates inventory information in your database to reflect the customer order, and then returns a confirmation notice in HTML format to the customer's browser. Alternatively, the notice may be sent in WML format to the customer's mobile phone or other wireless device.

4.2 How CGI Works

The flow of typical CGI implementation is as follows:

1. From a client machine—which could be anything from a Web browser to a mobile phone—a form or query is sent to the Web server. The user might launch a request by pressing a button or clicking a link.
2. The Web server forwards the request to the CGI program, in this case, written in ACUCOBOL-GT. This program ACCEPTs the CGI data sent by the client into an external form which you define in working storage.
3. The request is either processed by the CGI program itself, or passed to another ACUCOBOL-GT program for execution.
4. The CGI program uses the DISPLAY verb to merge processing results with an HTML, XML, or WML template, and sends the result to the Web server in the standard output stream. The ACUCOBOL-GT runtime creates and outputs the appropriate content response headers automatically.

5. The Web server sends the content response header followed by the appropriate content message to the user's machine or mobile device where it is displayed.



Note that this diagram shows the CGI program and ACUCOBOL-GT application as separate components. In many instances, you may choose to add CGI functions directly to your COBOL application, making them one in the same thing.

In addition, any and all connectivity options available to ***extend*** customers can be applied in the third step if desired. For instance, the CGI program can CALL a remote program using the AcuConnect® application server for connectivity. Likewise, the CGI program or CALLed program can access any file system or database that we support to perform a lookup. If the file system resides on another machine, the program can use the AcuServer® remote file server to achieve access. If a relational database access is desired, the Acu4GL® interface or AcuSQL® precompiler can be used to translate the request into SQL.

The CGI method is best suited for programs that do not use a user interface (UI) based on COBOL DISPLAY/ACCEPT statements—for instance, programs with a Web interface, batch processes, processes that use socket routines to communicate with an external UI, component adapter technology processes, and BEA Tuxedo processes.

4.3 Deploying Your Applications on the Web Using CGI

To deploy your applications on the Web using CGI, you perform the following steps:

1. **Create a Web interface** to your application, targeted for your intended environment. Include pointers to the CGI program that you will develop.
2. **Write a CGI program in COBOL** that will read CGI variables, perform the processing task or launch a separate program to do so, and generate output from the results. Note that ACUCOBOL-GT has been designed to make CGI programming easy.
3. **Create a runtime configuration file** for your CGI program. In it, define input and output requirements, such as output content type, headers, caching, and stripping carriage returns.
4. Place your CGI program and ACUCOBOL-GT application on the Web server, along with the necessary configuration, license, and data files.
5. **Configure your Web server** software. Have it run the program with the “-f” runtime option or set the A_CGI environment variable to “1”. This tells the runtime to run in CGI mode and not perform user interface functions. If your program will be running on UNIX, also specify the “-b” option.
6. Follow normal procedures for creating the license files for Windows and UNIX environments. If you ordered a CGI license file, see section 2.4 in the *Getting Started* guide.

4.4 Creating a Web Interface

To make your COBOL program directly accessible to Web browsers—whether on computers, personal digital assistants (PDAs), or mobile phones—you should develop a Web interface to replace your application's current graphical or character-based front end.

For HTTP browsers like Internet Explorer and Netscape, you typically develop an interface in HyperText Markup Language (HTML) or eXtensible Markup Language (XML). HTML is the most common language for Web pages viewed with HTTP browsers, although XML is rapidly gaining ground for display of dynamic content on the Web.

For mobile device browsers based on Wireless Application Protocol (WAP), you typically develop an interface in Wireless Markup Language (WML), but you can also develop an XML interface if desired.

XML documents are typically transformed to the output format of their targeted environment through the use of style sheets and a style sheet transformation language (XSLT). For instance, if a Web browser client makes a request, XML documents are typically transformed to HTML or PDF format for output. If a WAP device issues the request, they are typically transformed to WML. It is a published form of the document that is returned to the client, not the XML document itself. XML Web publishing vendors can provide more information on this subject, as well as the tools to develop such a solution.

HTML, XML, and WML are all markup languages that tell devices how to present information. All three of these languages use headers and tags to pass structure, formatting, hyperlink, and form description information to the receiving device. If you choose to, you can use other languages like VB script or Java script to develop your user interface. The only restriction is that the targeted browser or WAP device must be able to support the version of the language that you use.

To develop a markup language interface to your COBOL program, you can use any of several authoring tools. Many are available over the Internet free of charge. For instance, Cocoon Project is an open-source XML Web publishing framework from Apache. A search on HTML authoring tools turns up literally thousands of suggestions.

Whichever language you choose, you typically create fill-out forms to collect information from the user and send it to your CGI program for processing. *The form must include information about how and where to send the information so that it can be processed.* In HTML, this is accomplished with the METHOD and ACTION attributes of the <FORM> tag.

Section 4.4.1 describes how to construct a form in HTML. For information on constructing XML and WML forms, refer to XML and WML reference books.

Note: If desired, you can display an HTML interface using the WEB-BROWSER control described in **Appendix B** of this manual. Then the HTML interface can interact with CGI programs on the Web server.

4.4.1 Creating HTML Forms

To create a form in HTML, you use the <FORM> tag. As with many HTML tags, you need to place a beginning tag at the start of the form and a closing tag at the end of the form, in this case <FORM> and </FORM> respectively. More than one form can be in a single document, but forms cannot be nested.

Forms can contain single- and multiple-line entry fields, check boxes, radio buttons, list boxes, push buttons, and hidden data. Every form has a Submit button that users can press to have their data sent to the Web server for processing. This data is a list of NAME/VALUE pairs, one for each component of the form. The NAME and, optionally, the initial VALUE of each form component is specified in the HTML code. Then the user can fill out the form and modify the VALUEs.

Here is an example of an HTML form:

```
<FORM METHOD="POST" ACTION="/cgi-bin/name.acu">
Please enter your name or leave the entry field
blank for "anonymous":
  <INPUT TYPE="text" NAME="username" SIZE=60>
  <INPUT TYPE="submit" VALUE="Submit">
</FORM>
```

The FORM tag has two attributes that must be defined:

- The **METHOD attribute** tells the browser how to send the information gathered in the form to the Web server.
- The **ACTION attribute** tells the browser where to send the encoded form information, typically the URL (uniform resource locator) of the CGI program that will process the form.

METHOD attribute

The METHOD part of the <FORM> tag tells the browser how to send the information gathered in the form to the Web server. METHOD has only two possible values: GET and POST. The difference between these two methods is how the browser encodes the data to be sent to the server.

The GET method encodes the NAME/VALUE pairs into the URL itself. Then the Web server moves the NAME/VALUE list into an environment variable, QUERY_STRING, before calling the CGI program.

The drawback of GET is that the NAME/VALUE list is limited to 255 characters including the special URL encoding symbols. The advantage is that all of the input data is visible in the URL (although this could be considered a disadvantage as well). After submitting the form, the user can "bookmark" the URL or save a copy of the URL text for future quick access to the results of the submittal.

On the other hand, the POST method sends the data to the Web server as a data stream. Using this method, the NAME/VALUE list is limited to 2,147,483,647 characters. With the POST method, the browser does not modify the URL. Instead, it connects to the Web server and sends the NAME/VALUE list as a stream of data. The Web server pipes this data stream through the standard input stream (STDIN) of the CGI program.

All Web servers support the GET method, and most modern Web servers support the POST method. ACUCOBOL-GT supports both methods. This means that when you write your CGI program using the ACUCOBOL-GT CGI enhancements, the method you choose makes absolutely no difference in the way you code your CGI program. You may change the method in the HTML FORM tag without recoding or even recompiling your CGI program.

ACTION attribute

ACTION tells the browser where to send the encoded form information. This is the URL of the CGI program that will process the form. In the example above, the browser is going to send the form information to a COBOL program called “name.acu” on the same server as the HTML document. The program “name.acu” is located in a directory called “/cgi-bin/”. Note that “/cgi-bin/” is not necessarily the name of an actual file system directory on the Web server. The Web server is configured to map URL paths to disk directories. You must examine your Web server settings to determine the actual disk directory represented by a URL path.

You can also include the complete URL within the ACTION attribute as follows:

```
<FORM METHOD=“POST” ACTION=“http://www.mycompany.com/cgi-bin/name.acu”>
```

4.4.2 FORM Components

In the following syntax diagrams, square brackets, [ ], are used to indicate optional attributes, and curly braces, { }, to enclose a set of choices.

The following components are described in the sections that follow:

- **INPUT tag**
- **TYPE attribute**
- **Single-line entry fields**
- **Multiple-line entry fields**
- **Check boxes and radio buttons**
- **List boxes**
- **Submit and Reset buttons**
- **Hidden fields**

INPUT tag

Use the INPUT tag to specify how you want users to enter data into the form. It tells the browser to get ready for some sort of data input.

TYPE attribute

Use the TYPE attribute to tell the browser what kind of data will be entered and how it should look on the form. The default value for TYPE is "TEXT" (i.e. TYPE="TEXT").

Single-line entry fields

To create a single-line entry field, use the following syntax:

```
<INPUT TYPE="{TEXT|PASSWORD}" NAME="name"
  [VALUE="default_text"][SIZE="width,height"]
  [MAXLENGTH="width"]>
```

where:

- TYPE selects the type of input field (text box or password box).
- NAME assigns a name to the field.
- VALUE assigns default text that will be entered in the box when the form is displayed.
- SIZE specifies a width and height (in characters) for the box (default is width 20 and height 1).
- MAXLENGTH specifies the maximum number of characters that may be entered.

The main difference between the TEXT and PASSWORD types is that when you use PASSWORD, anything the user types into the field is displayed as asterisks.

Multiple-line entry fields

To create a multiple-line entry field, use the following syntax:

```
<TEXTAREA NAME="name" [ROWS=rows] [COLS=columns]>
[Default_text]
</TEXTAREA>
```

where:

- NAME assigns a name to the field.
- ROWS specifies the number of rows in the field.
- COLS specifies the number of columns in the field.

Check boxes and radio buttons

Check boxes allow the user to select one or more options. Each check box must have a unique name. Radio buttons allow only one choice within a group of buttons. Each radio button within a group should have the same name. You can create more than one group of radio buttons by using different names.

```
<INPUT TYPE="{CHECKBOX|RADIO}" NAME="name" VALUE="value" [CHECKED]>
```

where:

- TYPE selects the type of field (checkbox or radio button).
- NAME assigns a name to the field.
- VALUE assigns the value that will be sent to the CGI program if the user selects this field.
- CHECKED specifies that this field should be selected by default when the form is displayed.

List boxes

List boxes allow the user to select from a number of options, using either a pull-down menu or a scrolling list box. They are similar to groups of radio buttons or check boxes, but they take up less space for long lists of items.

```
<SELECT NAME="name" [SIZE="size"] [MULTIPLE]>
<OPTION [SELECTED]>Option 1
<OPTION [SELECTED]>Option 2
...
</SELECT>
```

where:

- NAME assigns a name to the field.
- SIZE specifies how many lines should be visible at once (default is 1).
- MULTIPLE allows the user to select more than one option (i.e. check box behavior).
- OPTION designates a list item; the text beside the tag will be the value sent to the CGI program.
- SELECTED specifies that this list item should be selected by default when the form is displayed.

Submit and Reset buttons

When users finish filling out the form, the submit button allows them to send the contents of the form to your CGI program. The reset button allows them to clear the form, resetting the fields to their default values.

```
<INPUT TYPE="{SUBMIT|RESET}" [VALUE="value"]>
```

where:

- TYPE specifies the type of button (submit or reset).
- VALUE assigns text that will be displayed in the button (default text is "Submit" or "Reset").

If you want to enable the <Enter> key so that users can press <Enter> to submit a completed form, add the USE-RETURN style to the WEB-BROWSER control. See Appendix B for more information on the WEB-BROWSER control.

Hidden fields

Hidden fields allow you to send data to the CGI program without showing that data to the user.

```
<INPUT TYPE="HIDDEN" NAME="name" VALUE="value">
```

where:

- TYPE specifies that this should be a hidden field.
- NAME assigns a name to the field.
- VALUE assigns the value that will be sent to the CGI program.

Note that “Hidden” fields are not completely hidden from users, because any user can select the “View Source” option in their browser to see the value of these fields.

4.5 Writing a CGI Program

Each element of your HTML, XML, or WML interface has a corresponding CGI variable. Your application must be able to interpret the CGI input data and return an appropriate response to the user. This is where your CGI program comes in.

Your CGI program must perform three basic functions:

Function	Description
Read CGI input data from the client	When a user enters information onto the form, that information is sent to the CGI program in the form of CGI data. Your program must be able to read CGI input data. In ACUCOBOL-GT, this is accomplished with the ACCEPT verb.
Process the input data and arrive at results	Typically, this involves either a calculation, database lookup, or file read, but it could involve a CALL to an existing COBOL program on a local or remote machine.
Generate output that can be read by the client browser	Minimally, this includes an HTTP response header with a URL pointer to the response data. Otherwise, the header may be followed by response data formatted in HTML, WML, or XML. In ACUCOBOL-GT, HTTP output is accomplished using the DISPLAY verb.

In the simplest case, your ACUCOBOL-GT CGI program can contain one ACCEPT statement and one DISPLAY statement. Even if your program is more complicated, it will always start with an ACCEPT and end with a DISPLAY. Few languages make CGI programming so simple.

When writing your CGI program, consider the following:

- CGI programs must be written to be non-interactive. They take a set of input data, process it, and produce output.

- While the CGI program is running, the user is waiting for a response in the Web browser. CGI programs should be kept small and do their job quickly to reduce the user wait time.

- If users get tired of waiting and press the Stop button on their browsers, the Web server generally kills the CGI program. The output from the CGI program is discarded.

- One of the limitations of CGI is that it does not automatically maintain any state information. It is the CGI programmer's responsibility to record state information in a file or database and then encode a "key" to that state information in the HTML, XML, or WML output.

  For example, if a client invokes a CGI program to "log in" to your application or to add an item to his/her "shopping cart," the CGI program must record that fact along with any user identification information in a file or database. When the CGI program generates the output, it should encode a user ID or key in a CGI variable that will get passed to the next CGI program that the client invokes. The next CGI program can then look up the user state information (for example, shopping cart contents) from the database. The user state information should also include a date/time "stamp" so that a maintenance program can delete records for users who haven't logged on in a specified amount of time or who left the application without logging out.

- If you choose to use ANSI style ACCEPT and DISPLAY statements instead of—or in addition to—the ACCEPT and DISPLAY *external-form-item* syntax described in this section, you must include the UPON SYSOUT phrase or else compile with the "-Ca" option. "-Ca" implies UPON SYSOUT for all ANSI DISPLAY statements.

With these considerations in mind, you are ready to write your CGI program. The following sections describe how to accomplish the necessary I/O and processing tasks. **Section 4.5.4, "Sample CGI Programs"** provides some sample code for your reference.

4.5.1 Reading CGI Input Data

With ACUCOBOL-GT, there are two ways to read CGI input data:

- With the **ACCEPT** verb. Use this method in most cases.
- With the **C$GETCGI** library routine. Use this method when converting existing COBOL CGI programs to ACUCOBOL-GT.

Using the ACCEPT verb

To read CGI variables from the client machine, you can use the ACCEPT verb in your CGI program. ACUCOBOL-GT includes special syntax for accepting HTML, XML, or WML form records. The syntax is:

```
ACCEPT external-form-item
```

where *external-form-item* is an input record for an HTML, XML, or WML form. It is a group data item (declared with the IS EXTERNAL-FORM clause) that has one or more elementary items associated with CGI variables. For "input forms," the association is made using the IDENTIFIED BY clause in the description of the elementary item. The value of external-name is the name of the CGI variable. If the IDENTIFIED BY phrase is omitted, the data item's own name (data-name) is used as the name of the CGI variable.

External-form-item may also be an output record for an HTML, XML, or WML form. In this case, the group item is declared with both the IS EXTERNAL-FORM and IDENTIFIED BY clauses.

The "external form" is called an "output form" if the IDENTIFIED BY clause is used in the description of the group item to associate it with a template file.

For example, the following is an input form:

```
01  CGI-FORM   IS EXTERNAL-FORM.
    03  CGI-VAR1   PIC X(10).
    03  CGI-VAR2   PIC X(10).
```

and here is an output form:

```
01  HTML-FORM   IS EXTERNAL-FORM IDENTIFIED BY "tmplate1".
    03  HTML-VAR1   PIC X(10).
    03  HTML-VAR2   PIC X(10).
```

The ACCEPT verb treats input and output forms the same. ACCEPT sets the value of each elementary item in the external form, in order, to the value of its associated CGI variable, padding with trailing spaces. ACCEPT automatically decodes and translates the CGI input data before moving it to the elementary items of *external-form-item*. The value of each CGI variable is converted to the appropriate COBOL data type when it is moved to the external form.

Please note that when some browsers encounter multiple-line entry fields (also known as HTML TEXTAREAs, they send a carriage return line feed sequence to the CGI program. If carriage returns are not desired, as in operating systems that automatically terminate text lines with line feed characters, you can have them removed by using the CGI_STRIP_CR runtime configuration variable.

Also note that CGI variable names are case-sensitive. However, for convenience, if ACCEPT cannot identify a CGI variable, it will repeat the search for the variable ignoring the case.

If the CGI variable is empty or does not exist, ACCEPT sets the value of numeric data items to zero and nonnumeric data items to spaces.

If the CGI variable is repeated in the CGI input data, as in the case where multiple items have been selected from a “choose-many” list, the external form item that is identified with the CGI variable must be in a table using the OCCURS clause. Otherwise, only the first CGI value is moved to the external form item.

For example:

```
01  CGI-FORM   IS EXTERNAL-FORM.
    03  CGI-TABLE   OCCURS 10 TIMES.
        05  CGI-VAR1   PIC X(10).
        05  CGI-VAR2   PIC X(10).
```

or

```
01  CGI-FORM   IS EXTERNAL-FORM.
    03  CGI-VAR1   PIC X(10) OCCURS 10 TIMES.
    03  CGI-VAR2   PIC X(10) OCCURS 10 TIMES.
```

ACCEPT moves the values of the CGI variable to the items in the table. After all of the CGI values have been moved to items in the COBOL table, the remaining items in the table are set to “0” if they are numeric items and spaces otherwise.

Using the C$GETCGI routine

The C$GETCGI library routine retrieves CGI variables from the environment or the standard input stream, “stdin”, like other types of COBOL CGI programs. The C$GETCGI routine should be used by those with existing COBOL CGI programs to retrieve CGI variables as normal while incrementally converting to ACUCOBOL-GT’s external form method of CGI data retrieval. Although “ACCEPT from stdin” and “ACCEPT *external-form-item*” cannot be used together, you may use C$GETCGI instead of or in combination with external forms. The C$GETCGI routine retrieves the exact size of a CGI variable.

To use C$GETCGI, include a CALL in your CGI program using the following syntax:

```
CALL “C$GETCGI”
   USING VARIABLE-NAME, DEST-ITEM, VALUE-INDEX
   GIVING VALUE-SIZE
```

where the following are defined in the Working-Storage or Data Division sections of your program.

Parameter	**Type**	**Description**
VARIABLE-NAME	PIC X(n)	Contains the name of the CGI variable.
DEST-ITEM	PIC X(n)	Receives the value of the given CGI variable.

Parameter	Type	Description
VALUE-INDEX	Numeric value	Contains the CGI value index. This optional parameter contains an index that is used when a CGI variable has multiple values in the CGI input data. This typically happens when multiple items have been selected from a "choose-many" list box. For example, to receive the third selected value, pass 3 for VALUE-INDEX. If VALUE-INDEX is greater than the total number of values in the input stream for the given CGI variable, spaces are moved to DEST-ITEM.
VALUE-SIZE	Signed numeric value	Receives the size of the resulting value. This may be "0" to indicate that the variable exists but has no value or "-1" to indicate that the variable does not exist.

C$GETCGI automatically determines whether to read the CGI variable from the environment or "stdin" depending on the value of the "REQUEST_METHOD" environment variable, which is set by the Web Server. The first time C$GETCGI is called, it reads all of the CGI variables and values into a variable-length buffer. If REQUEST_METHOD is "GET", the data is read from the QUERY_STRING environment variable. If the REQUEST_METHOD is "POST", it is read from "stdin".

Each time C$GETCGI is called, it searches for the variable name passed in the first parameter, translates the value from CGI format into standard format, and moves the result to the destination item passed in the second parameter.

Note: When some browsers encounter multiple-line entry fields (also known as HTML TEXTAREAs), they send a carriage return line feed sequence to the CGI program. If carriage returns are not desired, as in operating systems that automatically terminate text lines with line feed characters, you can have them removed by using the CGI_STRIP_CR runtime configuration variable. Refer to **section 4.6** for details.

An optional third parameter specifies a CGI value index. This index is used when a CGI variable has multiple values in the CGI input data, as in the case where multiple items have been selected from a "choose-many" list.

4.5.2 Processing the User's Request

The second function of most CGI programs is to process the CGI input data or user's request. Usually, this is a calculation, database lookup, or file read, although CGI programs can invoke modules of an existing application to perform a function and return data. This is the part of CGI programming that is most familiar to COBOL developers: the application processing component.

It is important to note that since CGI programs are executed on the Web server machine by the Web server itself, they are not allowed to perform any user interface operations directly. If a CGI program attempts any operation that waits for a user response, it may cause the Web server to "hang."

Therefore, when writing CGI programs, you should be especially careful not to include code that waits for user input (using the ACCEPT verb).

There are some cases when the runtime displays a message box even before loading the COBOL program. There are other cases when the runtime shuts down due to an error that is not handled by the COBOL program. In these cases the runtime displays a message box containing the text of the message and waits for the user to press the OK button. This can cause a problem when the runtime is executed by a Web server.

To solve this problem, when you are configuring the Web server to execute the runtime, be sure to use the "-f" command line option or to configure your environment with the A_CGI environment variable. (Refer to **section 4.7, "Configuring the Web Server,"** for more information.) Both of these methods cause the runtime to suppress warning messages that are normally displayed in a message box. When the runtime shuts down due to an error that is not handled by the COBOL program, it constructs an HTML, XML, or WML page containing the shutdown message and sends it to the standard output stream before terminating.

4.5.3 Generating Output

Your CGI program can generate many types of output, including—but not limited to—HTML, WML, and XML. By default, it returns HTML output for users accessing your program on a Web site. If you want the output to be in XML format, or if you intend for it to be displayed on a WAP device, you must configure your program to generate XML or WML output instead. To do so, you use the runtime configuration variable, CGI_CONTENT_TYPE. (See **section 4.6** for more information on this variable.)

Regardless of the format, to generate output, your CGI program must use the DISPLAY statement. With ACUCOBOL-GT, the DISPLAY verb constructs HTTP response headers for you automatically and routes them to "stdout". The response header that the DISPLAY verb generates can include a pointer to a URL where response data can be found, or it can be followed by an HTML, WML, or XML document. Headers that point to URLs have the content type "location". Headers that include form data have the content type "text/html", "text/wml", or "text/xml".

Note that the runtime can DISPLAY virtually any type of content, as long as the content type ID corresponds to the form specified in the DISPLAY syntax.

Using the DISPLAY Statement

You can use the DISPLAY statement to display records from many types of forms, including HTML, WML, and XML forms. To do so, use the following syntax:

```
DISPLAY external-form-item
```

where *external-form-item* is an output record for the form when used in a Common Gateway Interface (CGI) program. It is a group data item (declared with the IS EXTERNAL-FORM and IDENTIFIED BY clauses) that may have one or more elementary items associated with fields in an HTML, WML, or XML template. The association is made using the IS IDENTIFIED BY clause.

External-form-item may also be an input record for a form. In this case, the group item is declared with only the IS EXTERNAL-FORM clause. This is used primarily when you are debugging your CGI program.

The DISPLAY verb treats input and output forms differently. For output forms, DISPLAY merges the data contained in the elementary items into the associated template file and sends the result to the standard output stream in conformance with the CGI specification. To do this, DISPLAY scans the template file for data names delineated by two percentage signs on either side (i.e., %%data-name%%). It then replaces those data names with the contents of the associated elementary items from the output form, stripping trailing spaces.

The maximum length of a single line in the template file is 256 bytes. The maximum length of a single output line is 512 bytes. No conversion is performed on the output form items before they are merged with the template file.

You may specify a series of directories for locating template files. To do this, use the HTML_TEMPLATE_PREFIX configuration variable, even if you are specifying a directory for locating XML or WML templates. (See **section 4.6** for details.) For related information about file content, see also the configuration variable “CGI_CONTENT_TYPE”.

When associating the template file with the IS IDENTIFIED BY clause, you may omit the template file suffix if it is either “.html” or “.htm”; otherwise, you must include the suffix. If the suffix is omitted, DISPLAY first appends “.html” to the specified file name and tries to open it. If that fails, DISPLAY appends “.htm” to the file name and tries to open it. If that fails, DISPLAY tries to open the file exactly as specified. If all these attempts fail, the following error message is sent to the standard output stream in HTML format:

```
Can't open HTML template “template-file-name”
```

When the Web server executes your CGI program, the current working directory depends on the configuration of the specific Web server that is running. In many cases the current working directory is the same as the Web server’s “root” directory. As part of the CGI specification, when the Web server executes your CGI program, it sets an environment variable called PATH_TRANSLATED to the directory containing your CGI program. You may want to use this information to locate your template files.

For example, if your template files are in the same directory as your CGI programs, set the HTML_TEMPLATE_PREFIX configuration variable to the value of PATH_TRANSLATED as follows:

```
01  CGI-DIRECTORY   PIC X(100) VALUE SPACES.
...
ACCEPT CGI-DIRECTORY FROM ENVIRONMENT "PATH_TRANSLATED".
SET CONFIGURATION "HTML_TEMPLATE_PREFIX" TO CGI-DIRECTORY.
```

The output from a CGI program must begin with a "response header". DISPLAY automatically generates a "Content-Type" response header if the specified template file is a local file (i.e., not a URL).

You may specify the EXTERNAL-FORM clause for an item that has no subordinate items. This is useful for displaying static Web pages. To do this, specify the name of the static Web page in the IDENTIFIED BY clause. For example, if you have a Web page called "webpage1.html", add the following lines to your COBOL program:

```
01  WEB-PAGE-1   IS EXTERNAL-FORM
IDENTIFIED BY "webpage1"
...
DISPLAY WEB-PAGE-1.
```

You may also specify a complete URL instead of a template file name in the IDENTIFIED BY clause. In this case, DISPLAY generates a "Location" response header that contains the URL. This header specifies that the data you're returning is a pointer to another location. To determine whether the template file name is a URL, DISPLAY scans it for the "://" string. DISPLAY does not apply the HTML_TEMPLATE_PREFIX when the template file name is a URL.

For example, if your program determines that the information the user has requested is on another Web server, and its URL is "http://www.theinfo.com", add the following lines to your COBOL program:

```
01  THE-INFO-URL   IS EXTERNAL-FORM
IDENTIFIED BY "http://www.theinfo.com"
...
DISPLAY THE-INFO-URL.
```

The length of the URL must not exceed 256 bytes.

Only one response header is sent to the standard output stream. Your CGI program should exit immediately after sending a location header (i.e., after displaying an external form identified by a URL).

You may use as many template files as you like in a single program. A common way to use multiple template files is to have three output forms: a header, body, and footer. Each of these has a corresponding template file. You first display the header form, then move each row of data to the body form and display it, and finally display the footer form.

When an input form is specified in a DISPLAY statement, the names and values of each elementary item are sent to the standard output stream in the format specified by the CGI_CONTENT_TYPE variable (HTML, WML, XML, etc.). One line is generated for each elementary item. The line consists of the name of the item followed by " = ", followed by the first 100 bytes of the item's value. This can be useful when you are testing and debugging your CGI program.

4.5.4 Sample CGI Programs

The following CGI programs were written in ACUCOBOL-GT.

Oscars sample

The first program, **oscars**, is designed to return the names of actors and actresses who won Oscar awards in a specified year. Notice the definition of external form items in the Working-Storage section of this program. Notice also that the ACCEPT statement in the main logic section ACCEPTs these external form items, and that the DISPLAY statement defines the HTML templates to be used: "HTML-header-form," "HTML-footer-form," and "body-para." The contents of these templates follows.

```
identification division.
program-id. oscars.
remarks.
working-storage section.
01  cgi-form       is external-form.
    05 y2004       pic x(5) is identified by "y2004".
    05 y2003       pic x(5) is identified by "y2003".
    05 y2002       pic x(5) is identified by "y2002".
01  cgi-form-table redefines cgi-form.
```

```
    05  cgi-year   pic x(5) occurs 5 times.
01  html-header-form is external-form identified by “header”.
    05  opening-messagepic x(40).
01  html-body-form is external-form identified by “body”.
    05  ryear      pic x(5).
    05  html-oscar-info.
        10  rmovie       pic x(25).
        10  ractor       pic x(42).
        10  ractress     pic x(42).
01  html-footer-form is external-form identified by “footer”.
    05  closing-message  pic x(40).
01  movie-values.
     05  2004-oscar.
        10 movie     pic x(25) value “THE LORD OF THE RINGS”.
        10 actor     pic x(42) value “Sean Penn MYSTIC RIVER”.
        10 actress   pic x(42) value “Charlize Theron MONSTER”.
     05  2003-oscar.
        10 movie     pic x(25) value “CHICAGO”.
        10 actor     pic x(42) value “Adrien Brody THE PIANIST”.
        10 actress   pic x(42) value “Nicole Kidman THE HOURS”.
     05  2002-oscar.
        10 movie     pic x(25) value “A BEAUTIFUL MIND”.
        10 actor     pic x(42) value “Halle Berry MONSTER’S BALL”.
        10 actress  pic x(42) value “Denzel Washington TRAINING DAY”.
01  movie-table redefines movie-values occurs 5 times.
     05  oscar-winners.
         10  best-movie   pic x(25).
         10  best-actor   pic x(42).
         10  best-actress pic x(42).
01  various-counters.
     05 idx-1              pic 99 value 1.
procedure division.
main-logic.
accept cgi-form.
     if cgi-form = space
          move “You did not select any years!” to opening-message
          display html-header-form
          move “Back up and try again.” to closing-message
     else
          move “Acucorp CGI in action.” to opening-message
          display html-header-form
          perform display-body-para
          move “THE END.” to closing-message
     end-if.
     display html-footer-form.
     stop run.
display-body-para.
     perform varying idx-1 from 1 by 1 until idx-1 > 12
          if cgi-year(idx-1) = space
```

```
                continue
            else
                move cgi-year(idx-1) to ryear
                move movie-table(idx-1) to html-oscar-info
                display html-body-form
            end-if
        end-perform.
```

Header.htm

```
<HTML><HEAD><TITLE>ACUCOBOL-GT CGI Header</TITLE></HEAD>
<BODY>
<H2>%%opening-message%%</H2>
<CENTER><H1>Oscar Winners</H1>
<HR>
<TABLE border cellspacing=0 cellpadding=5>
<TR>
<TH colspan=4 align=center>Your Selections</TH>
</TR>
<TR align=center>
<TH>Year</TH>
<TH>Best Movie</TH>
<TH>Best Actor</TH>
<TH>Best Actress</TH>
</TR>
```

Footer.htm

```
</TABLE>
</CENTER>
<H2>%%closing-message%%</H2>
<HR>
<P>The information you requested was processed by the ACUCOBOL-GT CGI
program.
<BR>Following the CGI standard, ACUCOBOL-GT was able to send
<BR>the requested data items to the appropriate templates and
<BR>return the completed HTML document back to you.</P>
</BODY></HTML>
```

Body.htm

```
<TR align=center>
<TD>%%ryear%%</TD>
<TD>%%rmovie%%</TD>
<TD>%%ractor%%</TD>
<TD>%%ractress%%</TD>
</TR>
```

Oscar.htm

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1252">
<META NAME="Generator" CONTENT="Microsoft Word 97">
<TITLE>ACUCOBOL-GT CGI Example</TITLE>
<META NAME="Template" CONTENT="C:\PROGRAM FILES\MICROSOFT
OFFICE\OFFICE\html.dot">
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080">
<H2>ACUCOBOL-GT CGI Example using ACUCOBOL-GT.</H2>
<H3>This example shows how easily you can use ACUCOBOL-GT to act as a
CGI program.
<BR>User input is transferred from the following HTML page to a
ACUCOBOL-GT program running <BR>
on the web server. The appropriate output is returned.</H3>
<P><HR></P>
<H1 ALIGN="CENTER">Oscar Trivia</H1>
<P>Select a year(s)and press the Submit Query button. <BR>
The Best Picture, Best Actor and Best Actress for each year selected
will be returned. </P>
<FORM ACTION="http://your_server_name/Scripts/oscars.acu"
METHOD="post">
<P>Year: </P>
<P>
<INPUT TYPE="checkbox" NAME="y2004" VALUE="2004">
2004
<INPUT TYPE="checkbox" NAME="y2003" VALUE="2003">
2003
<INPUT TYPE="checkbox" NAME="y2002" VALUE="2002">
2002
<P>
<INPUT TYPE="submit"  VALUE="Submit Query" >
</P></FORM></BODY>
</HTML>
```

Hello User sample

The following CGI program assumes that the user has been asked to enter his or her name, and that the entered name is stored in the CGI variable "username". This can be accomplished using the following HTML form:

```
<body>
<form action="/cgi-bin/simple.acu" method="get">
Please enter your name or leave blank for "anonymous":
<input type="text" name="username" size=60>
<input type="submit" value="Submit">
```

```
</form>
</body>
```

This program also uses an HTML template, “greeting.html”. Here is an example:

```
<body>
Hello %%username%%
This Web page is still under construction.
Please try again in a few days.
</body>
```

The CGI program then, is as follows:

```
identification division.
program-id. simple.
remarks.
data division.
working-storage section.
01  input-output-form is external-form is identified by “greeting”.
    03  user-name  pic x(60) identified by “username”.
procedure division.
main-logic.
    accept input-output-form.
    display input-output-form.
```

Substituting a URL sample

This program demonstrates using an external form to substitute a URL for the output.

```
identification division.
program-id. acusrch.
remarks.
data division.
working-storage section.
01  acucorp-search-url   pic x(40) is external-form
          identified by “http://www.acucorp.com/cgi-bin/acusearch”.
procedure division.
main-logic.
     display acucorp-search-url.
```

4.6 Creating a Runtime Configuration File for Your CGI Program

You can configure many aspects of input and output handling by including a runtime configuration file with your CGI program on the Web server. For instance, you can configure the type of output to be generated by using the CGI_CONTENT_TYPE variable. Name and format the configuration file as you would any ACUCOBOL-GT runtime configuration file. (See section 2.7 of the A*CUCOBOL-GT User's Guide* for details.) The configuration file for your CGI runtime can contain any or all of the following variables:

- **CGI_STRIP_CR**
- **CGI_CONTENT_TYPE**
- **CGI_NO_CACHE**
- **CGI_AUTO_HEADER**
- **HTML_TEMPLATE_PREFIX**
- **CGI_CLEAR_MISSING_VALUES**

CGI_STRIP_CR

When some browsers encounter multiple-line entry fields (also known as HTML TEXTAREAs), they send a carriage return line feed sequence to the CGI program. When this sequence is subsequently output to files on operating systems that terminate text lines with line feed characters, the data may appear to be double spaced.

If desired, you can configure the runtime to automatically remove carriage return characters from HTML data entered in multiple-line entry fields. To do so, set the runtime configuration variable CGI_STRIP_CR to "1" (on, true, yes). Stripping the carriage returns from this kind of input prevents double-spacing problems, as well as conflicts that may arise if the data is used in a context that does not expect a carriage return character to precede each line feed character. The default value for this variable is "0" (off, false, no).

For example, if an end user enters the following three lines in a TEXTAREA for a field called “thetext”:

```
Sometext line 1
Sometext line 2
Sometext line 3
```

The browser sends the following to the CGI program:

```
thetext=Sometext+line+1%0D%0ASometext+line+2%0D%0ASometext+line+3%0D%0A
```

If the CGI_STRIP_CR is set to “1” (on, true, yes), the runtime strips the carriage return characters so that the input line is the following:

```
thetext=Sometext+line+1%0ASometext+line+2%0ASometext+line+3%0A
```

CGI_CONTENT_TYPE

By default, the output generated by your CGI program is mapped as HTML content. To associate your CGI output with a MIME content type other than “text/html”, use the CGI_CONTENT_TYPE configuration variable. This variable lets you control the content type information in the header of output files created by ACUCOBOL-GT. Such information informs recipients of the type of content that they are about to receive.

Using this variable, you can configure your CGI program for many types of output, including eXtensible Markup Language (XML) or Wireless Markup Language (WML) for Wireless Application Protocol (WAP) devices like mobile phones.

Whichever format you choose, the US-ASCII character set is applied to the output by default. If you want the CGI output to be mapped to an alternate character set such as ISO-8859-I (Western European), then you can specify the character encoding set to use with the variable as well.

Include this variable in your runtime configuration file as follows:

```
CGI_CONTENT_TYPE contenttype; charset=encoding_set
```

Where *contenttype* is the MIME content type of the generated output, and *encoding_set* is the preferred character encoding set to use.

For example, the WML content type for WAP mobile phones is “text/vnd.wap.wml”. To associate your CGI output with WML, include the following in your configuration file:

```
CGI_CONTENT_TYPE text/vnd.wap.wml
```

If you want your WML output to be mapped to the Western European character set, include the following:

```
CGI_CONTENT_TYPE text/vnd.wap.wml; charset=iso-8859-I
```

The content type for eXtensible Markup Language (XML) documents is “text/xml”. If your program generates XML data, include the following:

```
CGI_CONTENT_TYPE text/xml
```

Caution: To avoid overriding other Content-Type associations, we suggest that you create a different configuration file for each of the MIME Content-Type associations that you make in your Web server setup.

Please note that if you use this variable, the external forms indicated in your program’s DISPLAY syntax *must* contain the appropriate content. In other words, if you associate your program with the “text/xml” content type, the forms must be “.xml” documents with XML syntax. If you associate it with “text/vnd.wap.wml”, the forms must be “.wml” documents with WML syntax. Your program can DISPLAY virtually any type of data, as long as the Content-Type ID corresponds to the external form file that you provide.

Be aware that if you do not use the proper file extension for your external form documents, the Web server will interpret the data as HTML and display the wrong data. WML and XML are also more sensitive to syntax errors than HTML.

In addition, note that the capabilities of the configuration entry CGI_NO_CACHE may be affected by the content type that you choose.

CGI_NO_CACHE

Using the runtime configuration variable CGI_NO_CACHE, you can choose whether or not the HTML output of your CGI program will be cached by the requesting client. By default, the runtime generates “Pragma: no-cache” in

the HTTP response header that gets sent to the standard output stream. If you set CGI_NO_CACHE to "0" (off, false, no) in the runtime configuration file, the runtime suppresses this line of the response header. The default value is "1" (on, true, yes).

CGI_AUTO_HEADER

Set the runtime configuration variable CGI_AUTO_HEADER to "0" (off, false, no) if you want to suppress the output of the HTML header. This can be useful when you want to execute a CGI program and include its output into an existing flow of HTML text. For example, with server-side includes (SSI), you can instruct the Web server to execute a subprogram in the manner of CGI and incorporate its output right into the HTML document before sending it to the requesting client. SSIs are commands in an HTML document that are interpreted by the Web server. They are used when you want to include the contents of another file in the current HTML document or to execute a script whose output will be included in the current HTML document before being sent to the browser client. Refer to any HTML documentation for information on using SSIs.

HTML_TEMPLATE_PREFIX

Use this configuration variable to specify a series of directories for locating HTML, XML, or WML template files. This variable is similar to FILE_PREFIX and CODE_PREFIX. Specify the directories as a sequence of space-delimited prefixes to be applied to the file name. All directories in the sequence must be valid names. The current directory can be indicated by a period (regardless of the host operating system). For example:

```
HTML_TEMPLATE_PREFIX . /html/templates
```

tells the runtime to look for templates in the current directory and the "/html/templates" directory.

If the template name specified in your CGI program's IS IDENTIFIED BY clause is a URL with "\\" characters, the runtime ignores the HTML_TEMPLATE_PREFIX setting.

CGI_CLEAR_MISSING_VALUES

This variable lets you to control the behavior of the ACCEPT statement when CGI variables are empty or do not exist in the CGI inut data.

By default, ACCEPT sets the value of numeric data items to zero and non-numeric data items to spaces if a CGI variable is empty or does not exist. Set the CGI_CLEAR_MISSING_VALUES configuration variable to “0” (off, false, no) if you do not want ACCEPT to clear the value of the data item in this case.

4.7 Configuring the Web Server

Most Web servers must be specially configured to invoke the ACUCOBOL-GT runtime to execute your CGI program. For instance, using **IIS** on Windows machines, you map the “.acu” extension with the ACUCOBOL-GT runtime (“wrun32.exe”) in the Web server configuration settings. On UNIX machines using **Apache**, you must provide a small shell script to invoke the runtime along with the COBOL CGI program.

Although we have included helpful suggestions in this section for configuring Web server products for this purpose, please be aware that we are unable to provide technical assistance for Web server configuration efforts. Configuration procedures vary with each Web server product, so be sure to refer to your Web server documentation for specific details.

Although configuration procedures vary with Web server platforms, each Web server shares one characteristic: if a CGI program attempts any operation that waits for a user response, it will cause the CGI process to “hang.” At best this wastes system resources. At worst, it may cause the entire Web server to “hang.” Even if your code does not wait for user input, there may be some cases when the runtime displays a message box even before loading the COBOL program, or when shutting down due to an error that is not handled by the COBOL program.

To help you avoid problems with the Web server when running your CGI program, configure the Web server with the **“-f” runtime command line option** as shown in the procedures below. If you prefer, you can accomplish

the same thing by configuring the environment with the A_CGI environment variable. In general, we recommend that you use the "-f" option, because environment variables can affect other COBOL programs as well.

In addition, if you will be running your CGI program on a UNIX system, you should include the **"-b" runtime option**. It inhibits terminal initialization, which can lead to problems with CGI.

To configure IIS on a Windows 2000 server

Following is a procedure for associating your ".acu" applications with the ACUCOBOL-GT runtime using Microsoft's Internet Information Server (IIS) Version 5.0 on a Windows 2000 Server. Your own Web server software may vary.

1. Start the Internet Services Manager, normally available under Start/ Programs/Administrative Tools/Internet Services Manager.
2. Select "Default Web Site" (or the Web site you wish to configure) in the left pane.
3. Expand this node and select the directory (or virtual directory) containing your ACUCOBOL-GT CGI program files.
4. From the Action menu, select **Properties**.
5. In the Properties sheet, select the **Directory** (or Virtual Directory) tab.
6. Uncheck the "Read" check-box. This will prevent the Web server from trying to deliver your object files to the client.
7. Create an application by clicking the **Create** button under Application Settings. (If the application has already been created, you will see a **Remove** button instead of a **Create** button. If so, skip this step.)
8. Set the "Application name" field to any name of your choosing. The default value is the name of the containing directory or virtual directory.
9. Make sure the "Execute Permissions" field is set to "Scripts only". This is the default.
10. Click the **Configuration…** button. A configuration screen appears with the "App Mappings" tab selected.

11. Click the **Add** button.

12. In the "Executable" field, enter the path to the ACUCOBOL-GT runtime ("wrun32.exe"), or use the **Browse** button to locate it.

13. To the end of this path, add your desired runtime options, followed by a space and "%s". Be sure to include the "-f" runtime option.

    For instance:

    ```
    C:\Program
    Files\Acucorp\Acucbl800\AcuGT\bin\wrun32.exe -f "%s"
    ```

14. In the "Extension" field, enter the extension you use for your ACUCOBOL-GT object files (by default, ".acu").

15. Click **OK** three times to close the Add dialog, the Configuration screen, and the Properties sheet.

To configure Apache on a UNIX server

If you are using the Apache HTTP server on a UNIX machine, a small shell script is required to invoke the runtime and your COBOL CGI program. Depending on your Web server settings, you may need to place this script in a particular directory (called a *ScriptAlias* directory in Apache), or you may need to assign it a particular extension, such as ".cgi". In Apache, you normally set extensions using the AddHandler directive.

A typical script to invoke an ACUCOBOL-GT program called "myprog.acu" might look like this:

```
#!/bin/sh
RUNTIME=/usr/acucobol/bin/runcbl
exec $RUNTIME –f –b myprog.acu
```

Note that both the "-f" and "-b" runtime options are specified. These options are described in the following sections.

4.7.1 "-b" Runtime Option

If you will be running your CGI program on a UNIX system, you should include the "-b" runtime option to prevent the runtime from attempting terminal initialization. If you do not, the runtime will attempt to open the file "/etc/a_termcap" (or the file named in the A_TERMCAP environment variable). It will then try to match the value of either the A_TERM or TERM environment variables, if set, to an entry in the "a_termcap" file. If any of these files or variables is not set up properly, the runtime will terminate with an error. By using the "-b" option, this entire process is bypassed. In addition, using the "-b" option prevents extraneous characters from being displayed in the HTML output.

This option has no effect on Windows platforms.

4.7.2 "-f" Runtime Option

The "-f" command line option ensures that the runtime does not perform user interface functions when running as a CGI program. It is recommended for both Windows and UNIX CGI implementations.

When executed with "-f", the server runtime suppresses warning messages that are normally displayed in a message box. When the runtime shuts down due to an error that is not handled by the COBOL program, it constructs an HTML page containing the shutdown message and sends it to the standard output stream before terminating.

4.7.3 A_CGI Environment Variable

The A_CGI environment variable also ensures that the runtime does not perform user interface functions when running as a CGI program, but it does so by changing the entire server environment. As with any environment variable, A_CGI may affect other COBOL programs in addition to your CGI program.

When A_CGI is set to “1” in the environment, the runtime suppresses warning messages that are normally displayed in a message box. When the runtime shuts down due to an error that is not handled by the COBOL program, it constructs an HTML page containing the shutdown message and sends it to the standard output stream before terminating.

5 Using the ACUCOBOL-GT Web Runtime

Key Topics

5.1 What Is the Web Runtime?

The ACUCOBOL-GT® Web Runtime is a special 32-bit version of the ACUCOBOL-GT runtime that enables you to run existing ACUCOBOL-GT applications over the Internet. The Web runtime is based on Microsoft's ActiveX technology. It is itself an ActiveX control that you can embed on your Web page. It takes your existing COBOL applications and quickly makes them accessible through browsers that support ActiveX, particularly Microsoft Internet Explorer.

The Web runtime extends the built-in capabilities of Web browsers, allowing you to execute a COBOL program over the Internet without opening and running it as a separate application.

Because it works with existing applications, the Web runtime provides an easy way for you to deploy your applications on the Internet or in an intranet or extranet. For example, if your corporation has employees or distributors dispersed throughout a region, you can give these users remote access to sales and inventory applications via the Web.

The Web runtime is designed to run applications that make use of our AcuServer® or AcuConnect® technologies on the data or application host. While the Web runtime runs only on Windows machines, it can access files or run programs on UNIX and other platforms using these remote file and application server technologies, or through other file system interfaces like the Acu4GL® interface.

Although it would be unusual to do so, you can run the Web runtime without AcuServer and AcuConnect if your application does not need access to remote files. If the Web runtime cannot find a license to AcuServer or AcuConnect, it runs in restricted mode. Please refer to **section 5.7.1** for more details.

Sample Scenario

Your inventory application is already written and compiled in ACUCOBOL-GT. You want to make it accessible to field employees on your Web site right away. You add the Web runtime to your Web page, along with the URL of your inventory application. You put the application and data, along with AcuServer, on the designated server. Now, field representatives

can visit your Web site to run your inventory application and check or update the availability of every item in stock. All they need is a laptop and Internet connection.

If the application is complex, or processing intensive, you could use AcuConnect to distribute the application processing among several servers.

5.2 How the Web Runtime Works

The ACUCOBOL-GT Web Runtime works by communicating between the end user at a client machine and the remote server. The process is simple:

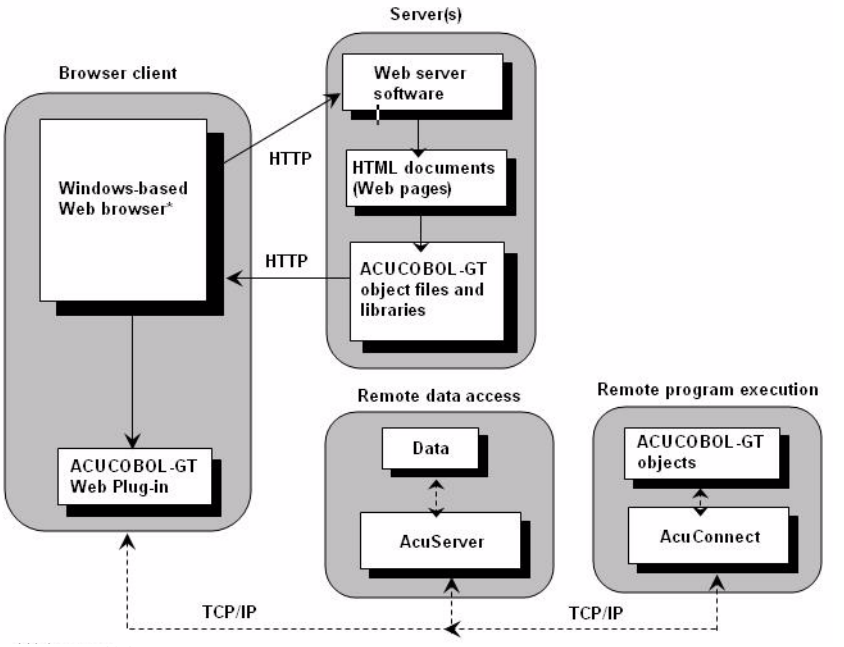
1. The end user visits your Web site.
2. The Web browser software looks for the Web runtime on the user's machine. If it locates the Web runtime, it runs it. The Web runtime, in turn, requests your application via HTTP, and launches the application locally.

   If the Web browser cannot find the Web runtime locally, it downloads the file from the specified location and asks the user "Do you want to install and run ACUCOBOL-GT Web Runtime?" This message may appear in a dialog box, or in newer versions of Windows (such as Windows XP, SP2), it may appear in an "Information Bar" at the top of the browser window. To assure the user that the installation is safe, the Web runtime is supplied as a cabinet (CAB) file with a digital signature from Acucorp.

   If the user grants permission, the runtime and program are sent to the client machine, where the runtime automatically installs itself and launches your program locally.

Note: For automatic installation to be performed, you must program your Web page with the CODEBASE attribute of the OBJECT tag as instructed in **section 5.6.1**.

As the following diagram illustrates, data may reside on a remote server, or the ACUCOBOL-GT application may be distributed over several servers. In this instance, sites will require AcuServer for remote file access or AcuConnect for remote application access. See **Chapter 6** for additional information on these technologies.



*Netscape or
or Microsoft Internet Explorer, Version 5.5 Service Pack 1 and earlier

5.2.1 Windowing Options

You can program the ACUCOBOL-GT Web Runtime to execute COBOL objects in their own application window or inside the user's Web browser window. How and where your program starts depends on the HTML commands that you use to include it on your Web site. (Please refer to **section 5.6** for details.) Listed below are some things to consider when choosing a windowing method.

Inside the User's Web Browser

In this case, your COBOL program starts inside the user's browser window. Nearly all of the ordinary functions of your COBOL program are available to your user and your program has access to library routines that can be used to communicate with the browser.

In this mode, users also have access to browser functions like Forward, Back, and Search, but when they use these functions, the COBOL program terminates. To avoid losing data, the COBOL program should be designed to handle the closing action gracefully within ten seconds. If users return to the page from which they launched the ACUCOBOL-GT application, it reloads like any Web page.

Note that because of a Microsoft child window restriction, applications running inside a browser window cannot display a main window menu bar as you or your users might expect. To work around this restriction, you can program your application's menu functions to be accessed from a toolbar or a pop-up menu that is activated with the right mouse button.

In a Separate Application Window

In this mode, your COBOL program starts in a window separate from the browser, making it look the same as it would if it were launched locally. All the functions of your original program are available to users, including a main window menu bar.

Although your program cannot access the library routines used to communicate with the browser in this mode, users can still access functions like Forward, Back, and Search simply by activating the browser window. However, when the user selects a browser function, the runtime object terminates. This is because the runtime object executes as an object related to the HTML page, even when running in a separate window.

5.2.2 How Your Program Executes

To execute your program, the Web runtime actually makes a copy of it, delivers the copy to the browser client, and then executes the copy. Because the Web runtime utilizes the browser cache, the cache retains a copy of the object file as long as it is not reloaded. When the application is closed, the copy is removed as well.

Note: The Web runtime only supports a single instance.

Although the recommended method of executing the ACUCOBOL-GT Web Runtime is with the <OBJECT> tag (see **section 5.6**), it can be executed by the browser in direct response to receiving data with the MIME content type, "application/vnd.acucobol." Internet Explorer looks at the Windows Registry to execute the program.

The browser loads the Web runtime and feeds it the data coming from the Web server. In this context there is no concept of a command line.

To pass "command line" style parameters to the ACUCOBOL-GT Web Runtime, you must specify them in HTML when you invoke the application in your Web page. (See **section 5.6**.) Note that any file specified as a runtime option that requires write access (such as "errors.txt") is written to the directory listed in the authorization file, "acuauth.txt". If the authorization file is missing, an error results.

5.2.3 Browser Versions Supported by the Web Runtime

The Web runtime is designed for browsers that support ActiveX controls. We have confirmed that the Web runtime runs on Internet Explorer Versions 5.5 Service Pack 2 and later. Browsers that do not support ActiveX controls cannot use the Web runtime. Currently, this includes Netscape browsers, and some earlier versions of Internet Explorer.

5.3 Deploying Applications via the Web Runtime

To deploy your application on the Web via the Web runtime, you (the developer) have three tasks:

1. Set up a Web site. (See **section 5.4** for details.)
2. Prepare and configure your ACUCOBOL-GT application for use with the Web runtime. Minimally, this could mean creating a library file with your application resources and configuration files, but optionally, you can also add Web-related library routines. (See **section 5.5**.)
3. Update your Web page to invoke your COBOL program. To do this, you typically embed the URL of the Web runtime along with the URL of your application on your Web page using an <OBJECT> element and the CODEBASE attribute. (See **section 5.6**.)

Once your work is done, the user has only one or two tasks:

1. Visit your Web site. The Web runtime will install itself and launch your program automatically.

   Note that two dialog boxes may be displayed in the process: one containing a security message if required by the security setting of the user's browser (see **section 5.9.3** for more information), and the other containing a click-wrap license agreement from Micro Focus.
2. If your application requires access to local resources or local network resources, the user has one additional task: to edit an authorization file. (See **section 5.9.4**.) If you write your application in such a way that the resources are accessed remotely (using AcuConnect and AcuServer), this step is not required.

5.4 Setting Up a Web Site

Setting up a Web site is probably the most time-intensive portion of this method, but it is not very difficult to do. Appendix A gives general information about setting up a Web site, including information on Web

servers, posting a site, and promoting a site. Many different tools are available to help you create a Web page quickly and easily. Refer to **Appendix A** for guidelines.

5.5 Preparing Your ACUCOBOL-GT Application for the Web Runtime

The Web runtime component is significantly different from standard ACUCOBOL-GT runtimes in that you invoke it to process a *single* file. The single file may be either:

- a library file that "packages" all of the objects and resources of your application into a single file. The library file can contain optional configuration files, together with bitmap or JPEG images. It can also contain extended file descriptors (".xfd" files) for use with Acu4GL.
- the initial object of your COBOL application, with the remaining objects residing on a server. (This approach requires the use of AcuServer or AcuConnect technologies.)

The procedure for preparing your application for use with the Web runtime depends on whether you will be deploying your application in a **distributed** or **non-distributed** environment.

To prepare your application for use in a non-distributed environment:

1. Code your application as usual, or use an existing application. As needed, include the appropriate library routines or fields described in **section 5.5.1**. Be sure to refer to the special coding considerations listed at the end of that section.
2. Compile your programs with ACUCOBOL-GT. The resulting object file can be executed from within a browser on any Windows machine that has a Web runtime component installed.
3. Configure the Web runtime if desired. This procedure is described in **section 5.5.2**. Note that you cannot access a configuration file on the client machine.

4. Package your COBOL objects and resources into a single library file. You can also bundle configuration files, bitmaps, and ".xfd" files into the package if desired. This procedure is described in **section 5.5.3**. (Please note that if you have a single-object application and you are not using a separate configuration file, bitmaps, or Acu4GL, there is no need to package your application into a library.)

5. Include the library file or single object in your Web site as described in **section 5.6**.

To prepare your application for use in a distributed environment:

1. Identify the files or processes that you want to distribute onto the server. Distributing files involves the use of AcuServer, our file server. Distributing processes involves the use of AcuConnect, our application server.

2. Code your application as usual, or review your existing code for usefulness in the distributed Web runtime environment. As needed, include the appropriate library routines or fields described in **section 5.5.1**. Be sure to refer to the special coding considerations listed at the end of that section. If you plan to distribute processes, you may need to make some programming changes.

3. Compile your initial program using ACUCOBOL-GT. The resulting object file can be executed from within a browser on any Windows machine that has the Web runtime installed.

4. Create client/server configuration files as described in the *AcuServer* and *AcuConnect User's Guides*. Minimally, the client configuration file should contain the path of the server files or processes. You can achieve this using the FILE_PREFIX or CODE_PREFIX configuration variables.

5. Configure the Web runtime as needed. This is described in **section 5.5.2**.

6. Move all programs except the initial program onto the server.

7. Set up and run AcuServer and/or AcuConnect on the server machine.

8. Include the initial object file in your Web site using one of the methods described in **section 5.6**.

5.5.1 Coding for the Web Runtime

Following are several optional library routines and fields included with ACUCOBOL-GT. These library routines and fields are designed specifically for use with the Web runtime.

Field/Routine	Description
W$BROWSERINFO routine	Returns the version and name of the requesting browser.
W$STATUS routine	Displays status information in the host browser's status bar.
IS-PLUGIN field in ACUCOBOL.DEF	Indicates whether or not the application is running in a Web browser via the ACUCOBOL-GT Web Runtime. Defined in ACUCOBOL.DEF.
W$GETURL routine	Passes a given URL to the browser. Lets you give end users access to other Web pages and the ability to send e-mail messages, conduct Web searches, and execute JavaScript.

Each of these routines and fields is described briefly below. They are described in detail in the *ACUCOBOL-GT User's Guide* or *Reference Manual*. For precise syntax and usage information, refer to those books.

The Web runtime component (acugtax.ocx) supports the same ACUCOBOL-GT syntax as the standard runtime, with one exception. The following command is not supported:

```
ACCEPT mystring FROM COMMAND-LINE.
```

To pass parameters for the Web runtime, the instantiation of the object must call the methods "AcuParam1" through "AcuParam14" to set parameters for the application. In the COBOL code, you must then use the C$GETVARIANT function to retrieve parameters and C$SETVARIANT to return parameters.

W$BROWSERINFO routine

Sometimes, you may want your application to respond in different ways to different browsers. In this case, you can use the W$BROWSERINFO library routine to determine the version and name of the requesting browser, that is, the one that will be hosting the COBOL application.

CALL this routine using the following syntax:

```
CALL "W$BROWSERINFO" USING BROWSERINFO-DATA
```

Parameters:

BROWSERINFO-DATA Group item as follows:

```
01  BROWSERINFO-DATA.
    03  USER-AGENT-STRING          PIC X(50).
    03  BROWSER-MAJOR-VERSION      PIC X COMP-X.
    03  BROWSER-MINOR-VERSION      PIC X COMP-X.
```

BROWSERINFO-DATA is found in the COPY library "ACUCOBOL.DEF."

Upon return from W$BROWSERINFO, all of the data elements contained in BROWSERINFO-DATA are filled in. If you call W$BROWSERINFO and the COBOL application is not running in a Web browser via the Web runtime, the first field is set to spaces and the last two fields are set to zero. The BROWSERINFO-DATA fields have the following meaning:

Field	**Description**
USER-AGENT-STRING	The browser's user_agent field. This contains the name of the browser software as it is sent to the HTTP server. It may also contain version numbers, product name, and operating system name. Netscape browsers set the first seven characters of this field to "Mozilla". Microsoft Internet Explorer sets this field to "Microsoft Internet Explorer".

Field	Description
BROWSER-MAJOR-VERSION	The major version number reported by the browser. This is not the same as the major version number displayed in the browser's "About" screen. For example, both Netscape and Internet Explorer put "0" in this field.
BROWSER-MINOR-VERSION	The minor version number reported by the browser. This is not the same as the minor version number displayed in the browser's "About" screen. For example, both Netscape and Internet Explorer put "9" in this field.

W$STATUS routine

If desired, you can display status information in the host browser's status bar by CALLing the W$STATUS routine.

CALL this routine using the following syntax:

```
CALL "W$STATUS" USING STATUS-MESSAGE
```

Where STATUS-MESSAGE, PIC X(n), contains the message to be displayed in the browser's status bar.

Note: This routine is available only when the calling COBOL program is running in a Web browser window via the ACUCOBOL-GT Web Runtime. This routine is unavailable to programs run in a separate window when it is executed by a Web browser. The RETURN-CODE register is set to "1" after a successful call and "0" if this routine is unavailable.

IS-PLUGIN field in ACUCOBOL.DEF

Using a field in ACUCOBOL.DEF, your COBOL program can determine whether or not it is running in a Web browser via the ACUCOBOL-GT Web Runtime. The field, known as IS-PLUGIN, is defined in the SYSTEM-INFORMATION group of ACUCOBOL.DEF.

For example, you might include the following code:

```
ACCEPT SYSTEM-INFORMATION FROM SYSTEM-INFO.
if IS-PLUGIN then
   display message box “Running via the Web runtime”
else
  display message box “Running via stand-alone runtime”
end-if.
```

W$GETURL routine

If desired, you can enhance your COBOL program to give your end users access to other Web pages and the ability to send e-mail messages, conduct Web searches, and execute JavaScript. You determine how much functionality you want to give to your end users, and you code those functions into your application by passing URLs to the Web browser with the W$GETURL library function.

The W$GETURL library routine tells the runtime to pass a given URL to the host browser. The browser will handle the URL as if it were typed in the URL entry field.

Each URL that you pass with the W$GETURL routine contains a protocol and a path, separated by a colon.

Accessing a Web page uses the “http” protocol. For example,

```
http://www.acucorp.com/
```

tells the browser to contact the Web server “www.acucorp.com” and ask for the root page (/).

Sending e-mail uses the “mailto” protocol. For example,

```
mailto:support@acucorp.com
```

opens an e-mail message to the user “support” at the machine “acucorp.com”.

JavaScript is also supported as a protocol, so you can execute JavaScript sequences that display dialog boxes, create Web pages, build text files, and much more.

Use this library routine as follows:

```
CALL "W$GETURL" USING URL, TARGET
```

where:

URL, PIC X(n), contains the complete URL. This can be of any type, such as http, FTP, news, mailto, gopher, or javascript.

TARGET, PIC X(n), represents the destination for displaying the URL. This can be a window or a frame. You can specify "_blank", "_parent", "_self" or "_top". You can also write the response data to a frame by specifying the frame name as the target parameter.

- "_blank" loads the link into a new unnamed window.
- "_parent" loads the link into the immediate parent of the document in which the link is contained.
- "_self" loads the link into the same window where the link was clicked, causing the page to be repainted.
- "_top" loads the link into the full body of the current window.
- "<window_name>" loads the link into a named HTML frame. If no frame or window exists that matches the specified target name, a new window is opened for the link.

After a CALL is made to W$GETURL, subsequent URL requests are ignored until the CALL completes.

Note: This routine is available only when the calling COBOL program is running in a Web browser window via the ACUCOBOL-GT Web Runtime. This routine is unavailable to programs run in a separate window when it is executed by a Web browser. The RETURN-CODE register is set to "1" after a successful call and "0" if this routine is unavailable.

Other coding considerations

When coding for the Web runtime, also consider the following points:

- If you plan to have your application run inside the user's browser window, it cannot display the main window menu bar to which you may be accustomed. (This is a Microsoft child window restriction.) In this case, you will need to program your application's menu functions to be accessed from a toolbar or a pop-up menu that is activated with the right mouse button. You can avoid this restriction by having your application run in its own separate window. You can specify this as a parameter of the <OBJECT> tag or using the AcuEmbedded method supplied with the object interface. See **section 5.6.1** and **section 5.6.1.3** for information on these options.

- When running in QUIT-MODE, your program may not accept or display anything after an ACCEPT terminates with the QUIT-MODE exception. If your program attempts to accept or display anything, the runtime terminates immediately as if the program has executed a stop run.

- If there is a file in the cache with the same name as your object file, it will be reused rather than downloaded from your Web site. You may need to implement a file naming scheme to prevent this.

- When required, consider how your application will access local resources without affecting end user security. Refer to **section 5.9, "Security,"** for security issues that affect coding.

- Applications designed for the Web runtime should accommodate situations where a runtime license file cannot be found or the maximum number of allowable connections has been exceeded. In some cases, the application can provide messages that explain why program functions may be restricted. In other cases, the application can prompt users to try to access the server again later. See **section 5.7.1, "Licensing Considerations,"** for more information.

- To prevent file I/O restrictions, you may want to design your program in such a way that one of the first activities it performs is to open a file with AcuServer. This enables the program to "check out" or secure a runtime license, thereby ensuring that end users have full access to runtime

functions. If a license file is found, the serial number is available in the "SERIAL-NUMBER" field of the SYSTEM-INFORMATION group in "acucobol.def".

- To minimize the amount of setup work required by users, we recommend that you write your application in such a way that resources are always accessed remotely (using AcuConnect and AcuServer). If your application requires access to local resources or local network resources, your users will have to edit the authorization file accordingly. Refer to **section 5.9.4.2** for information on editing this file.

- For security purposes, COBOL programs run in the Web runtime are not permitted to call certain library routines. For a complete list of these routines, refer to **section 5.9.4.3**.

5.5.2 Configuring the Web Runtime

Once you've coded your application for the Web runtime, you can configure the runtime system in one of two ways: **programmatically** using the SET CONFIGURATION or SET ENVIRONMENT variables, or with a runtime **configuration file**.

5.5.2.1 Programmatic configuration

To configure the runtime programmatically, use the SET CONFIGURATION or SET ENVIRONMENT phrase in your source code. For example, if your application uses a configuration file with the entries:

```
FILE_PREFIX @hal:/u2/serverfiles
COMPRESS_FILES 1
KEYSTROKE EDIT=PREVIOUS EXCEPTION=52 kl
```

add the following lines to your COBOL initialization code:

```
SET CONFIGURATION "FILE_PREFIX" TO "@hal:/u2/serverfiles".
SET CONFIGURATION "COMPRESS_FILES" to "1".
SET CONFIGURATION "KEYSTROKE" TO "EDIT=PREVIOUS EXCEPTION=52 kl".
```

MAX_FILES, MAX_LOCKS, and LOCKS_PER_FILE cannot be modified with the SET verb. These variables are used during the runtime initialization that occurs before the COBOL program is executed. In order to allow a wider range of applications to be used with the ACUCOBOL-GT Web Runtime, these variables have been given new default values.

When you are using the ACUCOBOL-GT Web Runtime, these variables are initialized to the following values:

MAX_FILES 255
MAX_LOCKS 512
LOCKS_PER_FILE 256

Note that the COBOL program can read environment variables using ACCEPT FROM CONFIGURATION (or ACCEPT FROM ENVIRONMENT).

5.5.2.2 Runtime configuration files

Create a configuration file for the Web runtime the same way that you would for the standard ACUCOBOL-GT runtime. Instructions can be found in the *ACUCOBOL-GT User's Guide* section 2.7.

Once you have a configuration file, you can access it via AcuServer, using the standard "@ServerName:/pathname" syntax, or you can bundle it into a library file along with the COBOL object files and resources. For information on using AcuServer to access your configuration files, refer to the *AcuServer User's Guide*, section 3.4. For information on creating library files, refer to **section 5.5.3** of this book.

Note that if you choose the library file method, you must also specify the name of the configuration file as an option when you invoke the Web runtime. (This is described in **section 5.6**.)

The Web runtime loads the configuration file by performing the equivalent of a SET CONFIGURATION statement for each line of the configuration file. Note that there are a few configuration variables that cannot be modified this way. They are MAX_FILES, MAX_LOCKS, and LOCKS_PER_FILE. When you are using the ACUCOBOL-GT Web Runtime, these variables are initialized to the following values:

```
MAX_FILES   255
MAX_LOCKS   512
LOCKS_PER_FILE   256
```

Note that FILE_PREFIX is initially set to the list of directories specified in the "acuauth.txt" security file. This authorization file specifies the directories to which your COBOL programs have access on the user's machine. The path given in "acuauth.txt" is limited to 4096 characters, like the FILE_PREFIX variable. The path given in this file must be enclosed in quotation marks. If you define an alternate FILE_PREFIX in the configuration file that you embed with your object library, that value overrides the "acuauth.txt" setting.

5.5.3 Packaging Your Application and Resources

When planning to package your application and resources, remember that the Web runtime is designed to process a single file. Therefore, if your application contains more than one object file and you do not plan to distribute your application across a client/server network, you must package all your COBOL object files and bitmap resources into a single library file. You can then access this file using the Web runtime.

If you will be using a configuration file, you can package the configuration file into the library file as well, or the configuration file must be on the remote computer. If your application will be accessing a relational database through our Acu4GL technology, you can package the ".xfd" files into the library file.

To create the library file package, you use the **cblutil** utility, the **COPY RESOURCE** statement, or a combination of the two. These are described in sections 5.5.3.1 and 5.5.3.2. Refer to the *ACUCOBOL-GT Reference Manual* for additional information.

If you are creating a library containing multiple COBOL objects, we recommend using "cblutil -lib" instead of using COPY RESOURCE. Using **cblutil**, you do not need to worry about the order in which COBOL objects are compiled (if you use COPY RESOURCE, you must ensure that the copied object is compiled first), and **cblutil** also checks for duplicated program names while COPY RESOURCE does not.

5.5.3.1 Using cblutil

The **cblutil** utility lets you embed *resources,* which are defined as pieces of static data, directly into an object file. For the purposes of the Web runtime, these resources can be applications, bitmap or JPEG image files, wave files, configuration files, and extended file descriptors (".xfd" files). The program treats the resource as if it were a disk file, but the resource is not actually a separate file in the target environment.

Using "cblutil -lib", you can specify any type of file as an input file. If an input file is a COBOL object, **cblutil** includes it in the resulting library as a COBOL object. If an input file is another library, each component of the library is individually added to the resulting library. Any other file is included as a resource.

To use the **cblutil** utility program, type "cblutil -lib" followed by the desired options, main program name, and all the modules you want to include, separated by a space. Be sure to add the main or initial program to the library first, because the Web runtime executes the first program it finds in the library.

Syntax:

```
cblutil -lib [options] main_program modules
```

Example:

```
cblutil -lib -v -o mylib.acu prog1.obj prog2.obj
  logo.bmp cblconfi data1.xfd data2.xfd
```

When using "cblutil -lib", you must use the "-o" option to specify the name of the output file. Because the Web runtime is registered to run files with the ".acu" extension, the output library file must have the ".acu" extension. In the example above, "mylib.acu" is the specified output file.

5.5.3.2 Using COPY RESOURCE

If you are creating a simple, single-object library, you can use the COPY RESOURCE statement to package your applications, bitmaps, and configuration file instead of the **cblutil** utility.

Use the COPY RESOURCE statement as follows:

```
COPY RESOURCE resource-name [ {IN} path-name ] .
                             {OF}
```

where r*esource-name* and *path-name* identify a resource file to be included in the resulting object file.

The effect of a COPY RESOURCE statement is to add *resource-name* to a list of resources that the compiler embeds into the resulting COBOL object file. The resources are added to the end of the COBOL object in the same order as the corresponding COPY statements. Because the resources are added to the end of the object, the location of the corresponding COPY RESOURCE statement in the COBOL program is irrelevant. Conventionally, COPY RESOURCE statements are placed either in Working-Storage or at the end of the program, but any location is acceptable.

If *resource-name* resolves to a COBOL object or library file, the compiler includes this object or library in the resulting object in a manner similar to "cblutil -lib". These are not considered resources, but are embedded COBOL objects.

5.6 Invoking Your COBOL Application with the Web Runtime

There are a number of approaches you can take to invoke your COBOL application from your Web page. You can:

- ***Use the <OBJECT> tag*** to invoke the application and Web control.

  Using an <OBJECT> tag, you can invoke your application and the Web runtime with a single HTML element. If you add the CODEBASE attribute, users can automatically download the control the first time they access the application, and get updates automatically. The browser checks the *clsid* attribute and downloads a new version of the control, if one is available.

In addition, we have developed a Web runtime object interface containing several properties and methods for communicating with browsers. If desired, you can implement all of the properties in the object interface as attributes of the <OBJECT> tag.

Advanced users may choose to instantiate their application with the <OBJECT> tag, and then write scripts with the Web runtime object interface to invoke the application. Please note that scripting the Web runtime may require changes to your end users' security settings. (See **section 5.9.2** for more information.)

The <OBJECT> tag is by far the most flexible method of invoking your application with the Web runtime; therefore, it is the method that we recommend for most situations.

- ***Embed* your COBOL** application in the HTML document using the HTML <EMBED> tag.

  This is convenient for sites that have already used the <EMBED> element with our Web Plug-in; however, the <EMBED> tag does not offer the capability of automatically downloading the Web runtime component, and it is not strictly part of the HTML specification.

  If you EMBED an application on your Web page, end users who visit your site and have already installed the ACUCOBOL-GT Web Runtime immediately start executing the first embedded object code on the page.

- ***Create a hyperlink*** to your COBOL application by placing the HTML anchor tags <A> and </A> in the HTML document.

  If you set up your object as a hyperlink (using the <ANCHOR> tag), end users who visit your Web page can run the program by clicking on the program's link. However, this method gives you the least amount of control over how your application is displayed, and it does not offer the capability of automatically downloading the Web runtime component.

5.6.1 Using the <OBJECT> Tag

When authoring Web pages to launch your application, you can use the <OBJECT> tag to invoke the Web runtime and start your program at the same time.

For example, include the following in your HTML code:

```
<OBJECT ID="AcuGTAX1" WIDTH=512 HEIGHT=384
   CLASSID="CLSID:077C768D-64C1-4AC1-845D-4589B4B2C24E"
   CODEBASE="http://www.acucorp.com/support/downloads/acugtax/
acugtax800.cab#Version=8,0,0,900">
   <PARAM NAME="SRC" VALUE="http://192.168.128.100/webinfo.acu">
</OBJECT>
```

where the following values must equal:

ID	The name of the instance of the object. This name is used for scripting the object interface. It is a user-defined value to which you refer in your script.
CLASSID	The GUID (globally unique identifier) assigned to the ACUCOBOL-GT Web Runtime control, specifically this value: `077C768D-64C1-4AC1-845D-4589B4B2C24E`
CODEBASE	The CODEBASE URL, from which the Web runtime can be downloaded and installed automatically by end users, specifically: `http://www.acucorp.com/support/downloads/acugtax/acugtax###.cab` where *###* is a 3-digit segment that identifies the cab file version. You can also append the version information that applies to the control, including a build number, by adding it as described in **section 5.6.1.1**. Although the CODEBASE attribute is optional, using it is now a common practice among software vendors to provide access to controls in this way. This allows you to distribute the control easily. If you do not use the CODEBASE attribute, you must direct users to the download page on the Acucorp Web site. Or, with a proper written license agreement, you may provide the control on your own distribution media or Intranet site.
HEIGHT	Optional. The height (in pixels) of the object's window. Use to define the area within the browser window that the application object will occupy.

WIDTH	Optional. The width (in pixels) of the object's window. Use to define the area within the browser window that the application object will occupy.
SRC	The ACUCOBOL-GT application you want to run. Typically the name of the library file package with a ".acu" extension.

By default, the application appears in the browser window, using the HEIGHT and WIDTH attributes, if provided, to define the area that the application occupies.

If desired, you can add any of the properties and methods of the Web runtime object interface as parameters of the <OBJECT> tag. To do so, add "PARAM NAME=" followed by the property name from the object interface enclosed in quotes. You then supply the VALUE attribute.

For example, if you want to have your application appear in its own window rather than in the browser window, set the AcuEmbedded property of the object interface to "FALSE" as follows:

```
<PARAM NAME="AcuEmbedded" VALUE="FALSE">
```

To supply runtime options or specify the name of a configuration file that was included in the library file, use the "AcuOptions" property, as shown below:

```
<PARAM NAME="AcuOptions"  VALUE="-d -c cblconfi">
```

The following options are valid for the Web runtime: "-c", "-d", "-e", "-l", "-v", or "-x".

The following example illustrates how you might debug your application and display it in a separate window.

```
<OBJECT ID="AcuGTAX1" WIDTH="512" HEIGHT="384"
   CLASSID="clsid:077C768D-64C1-4AC1-845D-4589B4B2C24E"
   CODEBASE="http://www.acucorp.com/support/downloads/acugtax/
acugtax800.cab#Version=8,0,0,900">
   <PARAM NAME="AcuEmbedded" VALUE="FALSE">
   <PARAM NAME="SRC" VALUE="http://yourserver/yourdirectory/yourprogram.acu">
   <PARAM NAME="AcuOptions"  VALUE="-d -c cblconfi">
</OBJECT>
```

Note: Since this example invokes the runtime in debug mode instantly, no scripting is necessary. In this case, “yourprogram.acu” is the name of a library.

For more information on the object interface, refer to **section 5.6.1.3**.

5.6.1.1 How the <OBJECT> tag works

The CODEBASE parameter indicates to the browser where to look for the Web runtime if it is not installed on the target system. Internet Explorer automatically offers to download and install the Web runtime, prompting most users to accept the digital signature provided by Acucorp.

Note: If the user’s security settings are high or customized to prohibit ActiveX controls, the user cannot install the Web runtime. If the user’s security settings are low, the control installs without confirmation from the user.

In the HTML document, you can introduce the Web runtime with the <OBJECT> tag and then supply a script to invoke the object through either a window event or a push-button event. For information on using the object interface, refer to **section 5.6.1.3**. However, you need not use scripting in order to invoke the control with the <OBJECT> tag.

5.6.1.2 Version number of Web runtime

The name of the CAB file identified in the CODEBASE URL is version-specific. Therefore, “acugtax800.cab” always contains the Version 8.0 Web runtime. However, you can implement your CODEBASE URL with a version string that refers to the version number as well as the build number.

If, for example, you deploy your Web site with the following CODEBASE value:

```
http://www.acucorp.com/support/downloads/acugtax/acugtax800.cab
```

a control associated with the specified CLASSID will be used (if found), regardless of version. If no associated control is available on the client machine, the CAB file will be downloaded, installed, and executed.

On the other hand, if your Web site contains the following value:

```
http://www.acucorp.com/support/downloads/acugtax/acugtax800.cab#Version=8,0,0,900
```

users with an earlier version of the control will automatically be prompted to download the new version the next time they visit your Web site. The convention for the Web runtime control is therefore:

```
acugtaxMmr.cab#Version=M,m,r,b
```

where *M,m,r,b* represent the Major, minor, release, and build numbers of the particular version of the control. This allows you to determine which version of the control is available to your end users.

To determine the version and build number of a Web runtime control, you can use the AboutBox method available in the object interface. See **section 5.6.1.3** for more information. You can also obtain the version number by following these steps:

1. Locate the file, acugtax.ocx, for the version you want to deploy.
2. In Windows Explorer, right-click on the control and select **Properties**.
3. Select the Version tab and view the value in the File Version field.

5.6.1.3 Web runtime object interface

The Web runtime's object interface consists of the following methods and properties. Some are bi-directional (B), some are write only (W), and others are read only (R). The component also has a default property (D).

Note: Using the object interface is optional. The <OBJECT> tag exposes the same properties (but not methods). The object interface is available for the advanced user who prefers scripting and wishes to use the available methods. However, please be aware that support for specific scripting languages and their implementations should be obtained from the appropriate vendor.

Methods

AcuIsActive	R
AcuExecute	R
AcuShutDownAx	R
AcuGetLastError	R
AboutBox	R

Properties

AcuParam1	W
AcuParam2	W
AcuParam3	W
AcuParam4	W
AcuParam5	W
AcuParam6	W
AcuParam7	W
AcuParam8	W
AcuParam9	W
AcuParam10	W
AcuParam11	W
AcuParam12	W
AcuParam13	W
AcuParam14	W
AcuOptions	B
AcuEmbedded	B
AcuShowLogo	B
AcuProgram	B
SRC	BD

Syntax

The following sections describe the methods and properties for the Web runtime control. Each method or property name is listed, followed by the type of the input parameter. (The type is shown in parentheses.) Finally, the input value and the output value (returned value) are shown, with acceptable values indicated in square brackets "[ ]". Empty brackets indicate that the value is not limited, or takes no input.

Examples of limited values include Booleans, which can be only "True" or "False". The output of the AcuIsActive method uses this limited value.

An unlimited value, for example, would be any valid ACUCOBOL-GT object. The AcuProgram property takes this unlimited input value.

You will notice that the syntax examples in the following sections refer to "AcuGTAX" or sometimes "AcuGTAX1". This refers to the HTML object ID assignment of the Web runtime control given in the <OBJECT> tag. For example:

```
<object classid="clsid:077C768D-64C1-4AC1-845D-4589B4B2C24E"
ID="AcuGTAX1" width="251" height="144">
```

AcuIsActive

Returns the status of the runtime. AcuIsActive takes no input parameters.

Parameter Type	Input	Output
( )	[ ]	[TRUE, FALSE]

The return value is of the data type BOOL:

- TRUE = the runtime is executing
- FALSE = the runtime is not executing

For example:

```
If AcuGTAX1.AcuIsActive() = TRUE
```

AcuIsActive may be executed any time after the Web runtime (in this example, “AcuGTAX1”) has been invoked.

AcuExecute

Starts the Web runtime, using the program specified in AcuProgram, and returns the result of the action.

Note: If you invoke your application directly using SRC as opposed to scripting, you do not need to call AcuExecute. Your application will start automatically.

The AcuExecute method takes no parameters.

Parameter Type	Input	Output
()	[ ]	[0, -4, -5, >0]

The return value is of data type LONG.

Value	Explanation
0	Success, runtime started.
-4	The runtime is already running.
-5	The runtime DLL could not be found, or the runtime DLL is the wrong version.
>0	COBOL error.

For example:

```
Return_value = AcuGTAX1.AcuExecute()
```

Do not execute this method until you have executed AcuProgram to specify the program to run; otherwise, AcuExecute does not know which program to run.

AcuShutdownAx

This is an optional method that forces a shutdown of a specific Web runtime instance. For example:

```
AcuGTAX1.AcuShutdownAx()
```

shuts down the runtime instance known as "AcuGTAX1", if it is running.

AcuShutdownAx terminates the COBOL application invoked by the Web runtime. In general, you need not use it because the Web runtime either terminates as a result of the COBOL program execution, or because the browser either displays another URL or closes. If, for some reason, you want to force shutdown of the Web runtime instance, this method is available.

Parameter Type	Input	Output
()	[ ]	[ ]

The shutdown action assumes that the COBOL application is currently in idle mode. If the runtime is not idle, the browser hangs, waiting for the application to idle. This method takes no parameters, and there is no return value for this function.

AcuShutdownAx implicitly terminates the Web runtime when you terminate the COBOL application you are running. So, when you execute this method, the Web runtime is terminated, closing files properly, even though data that was not stored permanently is lost.

AcuGetLastError

Returns the last known COBOL error code.

Parameter Type	Input	Output
()	[ ]	[ ]

This method takes no parameters, and the return value, which is the COBOL error code, is of data type LONG. Note that if the COBOL code could not be executed, the output is an error code indicating why the code could not execute. See "AcuExecute" for more information.

For example:

```
AcuGTAX.AcuGetLastError()
```

AcuGetLastError may be executed any time after the Web runtime has been invoked.

AboutBox

Displays a dialog box that presents version information about the Web runtime.

Parameter Type	Input	Output
()	[ ]	[ ]

There is no return value.

For example:

```
AcuGTAX.AboutBox()
```

AboutBox may be executed any time after the Web runtime (“AcuGTAX” in this example) has been invoked.

AcuParam1 … AcuParam14

AcuParam1...AcuParam14 is a variant data item. It can hold values to be passed to the COBOL application as parameters, or you can set the data item before execution and have data returned when the program terminates.

Note that you will not get any data until the application has halted with STOP RUN, GOBACK or EXIT PROGRAM. You should call the method, AcuIsActive, before trying to access the data. When AcuIsActive returns false (“0”), the data is available, provided there is any data.

Parameter Type	Input	Output
variant	[ ]	[TRUE, FALSE]

Because this is variant, the properties accept any variant-compliant data type: numbers, float, strings, etc.

For example:

```
AcuGTAX.AcuParam1 = "C:\TMP"
CurrentParam = AcuGTAX.AcuParam1
```

These properties are available any time after the Web runtime has been invoked. The contents of each property specified are read and applied to the runtime when AcuExecute is invoked.

You can use these additional properties to specify more parameters for the COBOL application. Note that to access properties, you must use C$GETVARIANT in your COBOL program.

Here is a Visual Basic example:

```
' Cobol test variables
Public testNum As Variant
Public testStr As Variant
Public testLongNum As Variant
Public testfloatNum As Variant

Dim CobolApp As Object
Set CobolApp = New AcuGT

' Set up config and error files
strCommandLine = _
   "-c ..\sample\autosrv\project1\acuvb.cfg -dle
..\sample\autosrv\project1\acuerr.log"

CobolApp.Initialize strCommandLine
' Set initial values
testNum = 123
testStr = "qwertyuiopasdfghjklzxcvbnm"
testLongNum = 1234567890
testfloatNum = 23.4
Label1.Caption = testNum
Label2.Caption = testStr
Label3.Caption = testLongNum
Label4.Caption = testfloatNum
Label9.Caption = "Before Cobol Call"
```

```
returnValue = CobolApp.Call("astest", testNum, testStr,
testLongNum, testfloatNum)

'Values after cobol program. This did not work before.
Label1.Caption = testNum
Label2.Caption = testStr
Label3.Caption = testLongNum
Label4.Caption = testfloatNum
Label9.Caption = "After Cobol Call"
```

You can find the whole project in sample\autosrv\project and sample\autosrv\cobol.

AcuOptions

Sets runtime options, such as "-d" to run the debugger or "-c *configfile*" to set the configuration file. The following options are valid for the Web runtime: "-c", "-d", "-l", "-e", "-x", or "-v". Note that you must have a local license file to run the debug option.

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

The AcuOptions property accepts a BSTR as a value.

For example:

```
AcuGTAX.AcuOptions = "-d -c cblconfi"
CurrentParam = AcuGTAX.AcuOptions
```

The AcuOptions property is available any time after the Web runtime has been invoked. Its contents are read and applied to the runtime when AcuExecute is invoked.

AcuEmbedded

Determines whether the runtime should run in an independent window or as a frame within the document.

Parameter Type	Input	Output
(BOOL)	[TRUE, FALSE]	[TRUE, FALSE]

The AcuEmbedded property accepts a BOOL as a value. When this property is set to "TRUE", the default, the window appears embedded in the browser document. Set it to "FALSE" when you want the window to appear independently, outside the browser.

For example:

```
AcuGTAX.AcuEmbedded = TRUE
if AcuGTAX.AcuEmbedded
```

The AcuEmbedded property is available any time after the Web runtime has been invoked. Its contents are read and applied to the runtime when AcuExecute is invoked.

Note that if you use the HEIGHT and WIDTH attributes of the <OBJECT> tag and you set AcuEmbedded to "FALSE", the Web runtime's logo screen will occupy that area on the HTML page while the application runs in its own window.

AcuShowLogo

Determines whether to display the ACUCOBOL-GT Web Runtime logo when you invoke your application.

Parameter Type	Input	Output
(BOOL)	[TRUE, FALSE]	[TRUE, FALSE]

For example:

```
AcuGTAX.AcuShowLogo = TRUE
if AcuGTAX.AcuShowLogo
```

To use this property, specify it as part of the <OBJECT> element where you invoke the application. The default is TRUE.

AcuProgram

Sets the COBOL application to run. You must use the absolute path even if the COBOL application resides inside the current directory and you have not specified the directory with the AcuOptions property.

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

Note: The security on your network may have an impact on which disks and directories an instance of the runtime is allowed to access.

The AcuProgram property accepts a BSTR as a value.

The AcuProgram property is available any time after the Web runtime has been invoked. Its contents are read and applied to the runtime when AcuExecute is invoked.

For example:

```
AcuGTAX.AcuProgram = "c:\tmp\myprog.acu"
CurrentProgram = AcuGTAX.AcuProgram
```

If you are using scripting, rather than implicit activation (with the <OBJECT> element), a call to AcuExecute() is necessary to activate the runtime.

Note: Use AcuProgram for MS-DOS and UNC paths only, such as “c:\mydir\myprog.acu” or “\\myserver\c\mydir\myprog.acu”. Use the SRC property instead for URL paths. You can use SRC or AcuProgram, but not both.

SRC

Required if you are using a URL path as the default property of the Web runtime. The value of the SRC property uses HTTP notation to specify the name of the remote ACUCOBOL-GT object to load. If you are using a local file path to specify the name of the ACUCOBOL-GT object, you must use AcuProgram instead.

Parameter Type	Input	Output
(BSTR)	[ ]	[ ]

The Web runtime provides this default property, since a browser may invoke a control by associating its file extension with the MIME type. This means the browser passes the .acu file to the object as the default property, so the Web runtime knows which program to run.

Note: You can use SRC or AcuProgram, but not both.

The following example demonstrates using the <OBJECT> element to specify the program for execution:

```
<OBJECT
  CLASSID="CLSID:077C768D-64C1-4AC1-845D-4589B4B2C24E"
  ID="AcuGTAX1" width="512" height="384">
  <PARAM NAME="AcuEmbedded" VALUE=”TRUE”>
  <PARAM NAME="SRC" VALUE="http://server.acucorp.com/location/webinfo.acu">
</OBJECT>
```

The SRC property is intended for HTTP URLs only. However, you may also address local files if the path is prefixed with “file://” rather than “http://”, as shown below. Note that this is also considered URL notation. For example:

```
<OBJECT
  CLASSID="CLSID:077C768D-64C1-4AC1-845D-4589B4B2C24E"
  ID="AcuGTAX1" WIDTH="512" HEIGHT="384">
  <PARAM NAME="AcuEmbedded" VALUE=”TRUE”>
  <PARAM NAME="SRC" VALUE="file://c:\webdemo\webinfo.acu">
</OBJECT>
```

When you use the SRC property to call a file, the downloaded file is given a temporary name and is stored in the current master temporary directory for the user, as specified in Windows. Note that the runtime automatically captures the temporary name and uses it internally; however, the internal name is not exposed.

5.6.1.4 Scripting with the object interface

Scripting is an optional method for invoking the Web runtime. While it is possible to invoke the Web runtime without scripting, some users may want to take advantage of the methods and properties in the object interface.

The following VB Script example illustrates how to invoke the object by a push button event using a script:

```
<INPUT id=button1 name=button1 type=button value=Button>
<SCRIPT LANGUAGE="VBScript">
sub button1_onclick()
   call AcuGTAX.src("http://www/Acucorp.com/demo/demo.acu")
   call AcuGTAX.AcuEmbedded(1)
   call AcuGTAX.AcuExecute()
end sub
</SCRIPT>
```

Note: Only one instance of the Web runtime is allowed. You cannot have multiple instances of the runtime in one browser instance.

To script runtime options, use the AcuOptions property. For example, to implement the configuration file and error file options with VB Script, you would use:

```
<SCRIPT LANGUAGE=”VBScript>
Sub window_onLoad()
  call AcuGTAX1.SRC(“http://www.acucorp.com/demo.acu”)
  call AcuGTAX1.AcuEmbedded(1)
  call AcuGTAX1.AcuOptions(“ -c cblconfi –e error.txt”)
  call AcuGTAX1.AcuExecute()
  end sub
</SCRIPT>
```

5.6.2 Using the <EMBED> Tag

You can use the HTML <EMBED> tag to embed your COBOL application in the HTML document. Using this tag, you define a rectangular area within your Web page that your COBOL application should use as its initial "window."

When you include a configuration file in the library, you must also specify the name of the file using the OPTIONS attribute of the <EMBED> element. This ensures that the application uses the correct configuration file rather than defaulting to another that may reside on the user's system. You can also specify runtime options using the OPTIONS keyword. The following options are valid for the Web runtime: "-c", "-d", "-e", "-l", "-v", or "-x".

For example:

```
<EMBED SRC="myprog.acu" WIDTH=400 HEIGHT=200
   OPTIONS="-c cblconfi -dlex myerrs">
</EMBED>
```

embeds "myprog.acu" on the Web page and makes it accessible using the Web runtime, with configuration file "cblconfi" and various options. In this example, "myprog.acu" could be an entire COBOL application, or it could be the initial COBOL object of a distributed application using AcuServer or AcuConnect, or it could be the name of a library file.

When a user visits this Web page, the embedded object ("myprog.acu") is automatically sent from the Web server to the client machine where it immediately starts running inside the user's browser window. Configuration variables are read from the configuration file "myconfig", and errors are written to the error file "myerrs".

While the COBOL object or library is downloading and after the COBOL program exits, the control displays a splash screen in the browser window. You can disable this feature by setting the AcuShowLogo property to FALSE. For complete information on the methods and properties of the Web runtime object interface, refer to **section 5.6.1.3**.

Please note that your application is not able to display its own main window menu bar when running inside a browser window. If you choose to use this method, you must program another way for your users to access your menu functions.

When invoking the Web runtime with the <EMBED> element, you can also specify runtime options and other properties using the AcuOptions attribute of the object interface.

If, for example, you want to specify both a configuration file and error file for the Web runtime, you could use:

```
<EMBED SRC="myprog.acu" WIDTH=400 HEIGHT=200
   AcuOptions="-c cblconfi -e errors.txt">
</EMBED>
```

The following code demonstrates using the property for instant debugging with the Web runtime:

```
<EMBED SRC="http://server.acucorp.com/activex/
   webinfo.acu"
   HEIGHT=384 WIDTH=512 AcuOptions="-d">
</EMBED>
```

These examples assume that the Web runtime has been installed and registered. When you use the <EMBED> tag, the Web runtime is not automatically downloaded, as it may be when you specify the CODEBASE attribute and a valid URL with the <OBJECT> element.

Note: Do not use the AcuProgram property when the ACUCOBOL-GT object is accessed via a URL. See **section 5.6.1.3** for more information.

5.6.3 Using a Hyperlink to Launch Your Application

You can use the HTML anchor tags, <A> and </A>, to create a link to your COBOL application in the HTML document.

Anchor tags are closed elements that highlight text or images. When users click a highlighted item on your Web page, they are transferred to the linked document. Because the link in this case is a COBOL program, when users click the highlighted item, the COBOL program is automatically invoked.

Note: To run your COBOL application in a separate window using the anchor tags, you must set up the standard ACUCOBOL-GT runtime as a helper application as described in Appendix C. To run your COBOL application in a separate window with the Web runtime, use the <EMBED> tag, the <OBJECT> tag, or the scripting method.

To turn text into a hypertext anchor, enclose the text in the anchor tags. The browser usually displays it underlined and in a different color. For example, your HTML document may include this:

```
<A HREF="myprog.acu">Click here to run the application</A>
```

The "HREF" attribute is used within the starting anchor tag to specify the document to be linked (or retrieved). Then when the user clicks on the highlighted text, "Click here to run the application," on your Web page, the COBOL object "myprog.acu" is retrieved from the specified location on the Web server and run inside the user's browser window.

To use images as hypertext anchors, you place the <IMG> element within the Anchor tags. For example:

```
<A HREF="myprog.acu">
<IMG SRC="myprog.gif" ALT="Click here to run application">
</A>
```

Then when the user clicks on the "myprog.gif" image on your Web page, the COBOL object "myprog.acu" is retrieved from the specified location on the Web server and run inside the user's browser window. While the graphic file is loading (or if the user's browser does not support images), the browser displays the alternate text, "Click here to run the application". Clicking this text invokes the application as before.

If you use a hyperlink to invoke the Web runtime, you must also configure your Web server software to generate the appropriate response header when it sends the object files to the client. The extension ".acu" has been registered as a MIME content type that is embedded in the Web runtime. The end user's browser will execute the Web runtime only if the MIME content type field of

your response header contains “application/vnd.acucobol”. For this reason, be sure to add a new File Extension/MIME Content Type mapping for “.acu” and “application/vnd.acucobol”. Your Web server software provides instructions for making MIME Content Type associations.

5.7 Obtaining and Distributing the Web Runtime

End users must place the Web runtime on the client machine, along with all the necessary license and data files. You can obtain the Web runtime from any ACUCOBOL-GT media or download it at no cost from the Acucorp Web site, **http://www.acucorp.com/support/downloads/**. (It cannot be loaded from the ACUCOBOL-GT runtime module.)

To make it easy for your end users to obtain the Web runtime, we have instructed you to embed the URL of the Acucorp download site to your Web page so that you can automate the download and installation process for your users. Your runtime license agreement must permit this form of distribution.

If you want, you can also distribute the Web runtime internally. If the target machine has Internet Explorer Version 5.5 Service Pack 2 or later installed, all that you need to do is copy the file “acugtax.ocx” to the preferred directory, and register it with the computer as described in **section 5.7.3**.

The capabilities of the Web runtime depend on its ability to locate a runtime license file either locally or via a server running AcuServer. If a runtime license file is found and available, the Web runtime has the capabilities of the standard ACUCOBOL-GT runtime. If a license file is not found, or if the maximum limit allowed by the runtime license provided by an AcuServer connection has been reached, the Web runtime runs in a restricted mode.

5.7.1 Licensing Considerations

The ACUCOBOL-GT Web Runtime is licensed in one of two ways.

1. With a runtime license on a **server** running AcuServer, licensed for the number of concurrent users anticipated.

2. With a runtime license on a **local machine** or network server.

5.7.1.1 Licensing the server

Most often, you install an ACUCOBOL-GT runtime license on the server running AcuServer. The license should accommodate the number of concurrent Web runtime users that you anticipate. (If you anticipate 100 concurrent users, you need a 100-user runtime license file on the server in addition to the AcuServer license file.) AcuServer detects the Web control connections and checks out a standard runtime license for each connection if a local license file is not found. When a Web runtime user disconnects from AcuServer by exiting the COBOL program, the runtime license authorization is released and made available to subsequent users. If the number of users exceeds the number of authorizations available on the server at any given time, the COBOL program receives a file status code 9D,105 and this message is output to AcuServer's error log:

```
You have exceeded the licensed number of clients for
AcuServer. If you would like to add clients, please
contact your customer service representative
```

If your COBOL program receives a status code of 9D,105, we recommend that you display a message box stating that the Web file server is busy, and to please retry later.

It is important to note that *the Web runtime continues to run even when no license file is available, but in a restricted mode*. In restricted mode, programs cannot use the following features:

- Indexed files
- Acu4GL
- The AcuSQL™ precompiler
- Debugging
- Printing
- Calling DLLs

If an end user tries to perform an activity that is not permitted, your program should present a message that explains why the activity cannot be completed. For example, if an end user tries to print something in restricted mode, your program could display a message that says "A software license could not be

found. XYZ Software is unable to print." Otherwise, the user might think that the printer is turned off, unplugged, or jammed. Your program should accommodate all of the limitations of a restricted runtime in some manner.

Programs running in restricted mode *can* use AcuConnect to call programs on a server machine that can use Acu4GL, AcuSQL, and the server's printing facilities. AcuConnect counts the number of concurrent runtimes it launches on the server. Programs running in restricted mode can also use COBOL applets, local sequential and relative files in permissible directories, C$GETURL, ActiveX, and OLE.

Note: If you want to test from COBOL whether a Web control application is running in restricted mode, you can test the SERIAL-NUMBER field of the SYSTEM-INFORMATION item after an ACCEPT from SYS-INFO. If the SERIAL-NUMBER field is SPACES, the COBOL program is being run in restricted mode.

5.7.1.2 Licensing by machine

Alternatively, users can purchase a runtime license and install the Web runtime on their local machine or network server. Upon execution, the control searches for a license file in the same directory as the executable.

The license file for the Web runtime is named "wrun32.alc." If a license file is found, the Web runtime has all the power of a standard ACUCOBOL-GT runtime. If no license file is available, the capabilities are restricted until the Web runtime tries to open a file with AcuServer. At that point, AcuServer checks for an AcuServer license file. If one is found, the OPEN succeeds, indicating that AcuServer has checked out an authorization for it. Then the control runs in unrestricted mode.

If you move the control's executable (.dll or .ocx) to a new directory, be sure to move the corresponding license file.

If you are licensed for additional ***extend*** technologies, such as Acu4GL or AcuSQL, a license file for each product the Web runtime will work with must reside in the same directory as the executable. Contact your Micro Focus ***extend*** representative for additional details.

Nothing in this section is intended to amend the terms and conditions of the applicable license agreement between you and Micro Focus. Rather, this section is meant to summarize the various aspects of our licensing technology that are required to operate the Web runtime. The terms and conditions of your licensing of ***extend*** software shall continue to be governed by the applicable license agreement between you and Micro Focus.

5.7.2 File System Dependencies

If you use file systems such as Pervasive.SQL or an Acu4GL product, the Web runtime depends on DLLs for these file systems.

The dependencies for these file systems must be stored in the same directory as the Web runtime (acugtax.ocx) because the system looks in the path of the control to load the additional files. However, since the installation directory of the Web runtime is determined by Windows and includes a registry entry, its dependencies can be stored in any directory. During installation, the control is registered so the operating system can locate the control and load it from that location.

5.7.3 Manual Registration of the Web Runtime

Registration of the ActiveX control will occur automatically during installation of your ACUCOBOL-GT development system, and for your users when they download the CAB file. As a result, there is no requirement to ever perform manual registration. However, should you need to do this as part of your development or testing, you can use the following commands.

Using the basic syntax with no options, a dialog box appears, indicating success or failure of the registration:

```
regsvr32 acugtax.ocx
```

You can omit the dialog box in silent mode by adding the “/s” parameter:

```
regsvr32 acugtax.ocx /s
```

To unregister the Web runtime, use the “/u” parameter:

```
regsvr32 acugtax.ocx /u
```

The “regsvr32.exe” utility is a Microsoft system tool and most Windows installations already contain the utility.

5.8 The User’s Job

To use your application via the Web runtime, end users typically have just one task: they must visit your Web site. If the user’s browser supports the Web runtime and you have implemented the CODEBASE in your HTML code, users install the Web runtime automatically the first time they visit your Web site, and the Web runtime automatically launches your application.

If your customers do not already have the ACUCOBOL-GT Web Runtime installed on their machine when they try to launch your application, they are informed that a control for the file type was not found, and they are asked if they want to install and run your program. If they respond “Yes”, the runtime installs with no further user interaction and automatically launches your program. The installation script automatically makes a file type association for ACUCOBOL-GT object files (“.acu” extension) with the Web runtime.

Note: Users must configure the browser to enable “ActiveX controls ”. To do this in Internet Explorer, they should check the appropriate box in the Security tab of the Internet Explorer Options dialog.

For most users installing the Web runtime, they will be prompted to accept the digital signature of the control. The system also displays the terms and conditions for using the ACUCOBOL-GT Web Runtime. Users must agree to the terms of the agreement in order to use the Web runtime.

If your application requires access to local resources or local network resources, the user has one additional task: to edit an authorization file. (See **section 5.9.4.**) If you write your application in such a way that the resources are accessed remotely (using AcuConnect and AcuServer), this step is not required.

5.9 Security

Security for the Web runtime begins with the following provisions:

- The “acuauth.txt” file contains a list of directories where COBOL applications are authorized to create or delete files and subdirectories.
- Calls to certain library routines are disallowed to prevent potentially damaging operations.
- Warning messages appear the first time users launch the Web runtime.
- We supply the Web runtime with a digital signature indicating that Acucorp verifies the content. This enables users to run the control using a medium security setting on their Internet Explorer browser.

Since we distribute the Web runtime via the Internet, it is packaged as a signed Cabinet (CAB) file. This ensures your users that the code is safe. The CAB file contains a compressed version of the control, with information that tells Internet Explorer how to install it.

Caution: Although we have built in a number of security features, the Web runtime allows COBOL programs to be executed on target machines. Because we have no control over the programs that are executed by the Web runtime, we cannot fully guarantee that they are safe.

5.9.1 Digital Signature of Web Runtime

A digital signature provides users with a way of identifying who published the software they are downloading from the Internet.

By distributing the Web runtime with a CAB file and digital signature, we ensure that most standard configurations of Microsoft Internet Explorer will accept the Web runtime application after the user responds “yes” to the Security Warning dialog when it is installed. Users who have set their browser security level to low will not see this dialog, since their browsers automatically accept digitally signed components.

Note: Users with security levels set to high cannot run any ActiveX-based controls, regardless of the digital signature.

Because the CAB installation does depend on some Microsoft files, it contains an embedded link to the Microsoft Web site to ensure that it can obtain the proper files. If these files are needed, users are prompted to install the Microsoft files; doing so may impact the download and installation time of the control, due to the file size of some of these required files.

5.9.2 How Internet Explorer Security Affects the Web Runtime

Certain security settings of Internet Explorer can affect how it handles the Web runtime. The following table outlines the security settings that could have an effect on your end users, depending on their configuration of Internet Explorer.

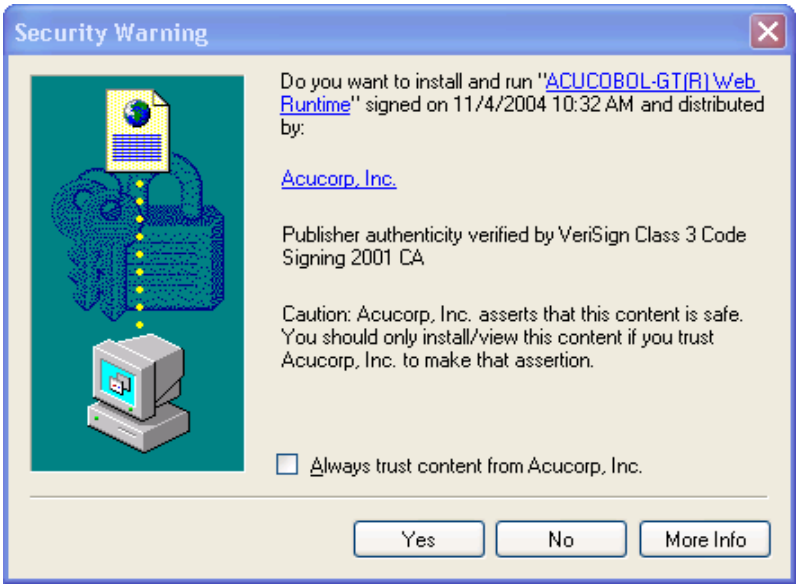
Security Setting Option	Low	Medium Low	Medium	High
Download signed ActiveX controls	Enable	Prompt	Prompt	Disable *
Download unsigned ActiveX controls	Prompt	Disable	Disable	Disable
Initialize and script ActiveX controls not marked as safe	Prompt	Disable	Disable	Disable
Run ActiveX controls and plug-ins	Enable	Enable	Enable	Disable*
Script ActiveX controls marked safe for scripting	Enable	Enable	Enable	Disable**
Don’t prompt for client certificate	Enable	Enable	Disable	Disable
Active Scripting	Enable	Enable	Enable	Disable**

* If your users have selected “Disable” for either of these settings—either via the default High security setting or a custom setting—they cannot run the Web runtime.

**If your users have selected “Disable” for either of these settings, and they are using the scripting facilities, they cannot run the Web runtime.

5.9.3 Security Warning Messages

In general, programs run with Web controls can potentially damage an end user’s computer system or corrupt memory through pointers or Working-Storage tables. Therefore, to help reduce the chances of this happening, the Web runtime displays warnings to end users, asking them to accept responsibility. Based on your end users’ expectations, you may need to provide instructions for handling these messages.



5.9.4 How the Authorization File Works

When a user first installs the Web runtime, the installation creates a default authorization file, “acuauth.txt”, and places it in the same directory that contains “acugtax.ocx”. If downloaded from our Web site, these files are installed into a hidden directory such as “Temporary Internet Files” or “Downloaded Program Files,” depending on the browser configuration.

If installed from our distribution media, “acuauth.txt” and “acugtax.ocx” are placed in the “C:\Program Files\Acucorp\Acucbl800\AcuGT\bin\” directory.

By default, the “acuauth.txt” file contains one line that specifies a list of directories where COBOL programs may create or delete files and subdirectories. The directories are specified as a sequence of pathnames delimited by quotation marks. The purpose of this file is to allow applications that create local temporary files to work immediately.

If you install the Web runtime as part of the ACUCOBOL-GT Development System, “acuauth.txt” contains only the following line:

```
"C:\Program Files\Acucorp\controldir\plgndata"
```

where *controldir* is the directory that contains “acugtax.ocx”.

If a COBOL application requires access to other directories or disks on the user’s machine, you must instruct your *end users* to add those directories to the “acuauth.txt” file manually. In this way, end users can accept responsibility for specifying which directories on the client machine may be made accessible to Web runtime applications.

Note: You cannot change this file programmatically.

We recommend designing your application so that this is not necessary. Using AcuServer and/or AcuConnect, you can design your application to utilize remote application and data resources.

However, if you do need access to the client machine, you might want to use messages to help users manage the file. For example, if an application receives file status “37,07” (access denied) when trying to open a file on the local machine, the program can display a message that indicates the problem and instructs the user to add the directory name to the file.

You can also instruct users to add other information to the file, for example, to access restricted library routines. See **section 5.9.4.4** for more information.

Once the file is edited, applications executed by the Web controls are allowed to read and write data in these directories, as well as create and delete subdirectories. No other area of the user’s local or network file system will be accessible to the COBOL programs executed by the Web controls. This provides some measure of protection for the end user’s machine and network.

If the “acuauth.txt” file is empty or missing when the Web runtime is invoked, the COBOL application is denied access to the local machine’s file system.

5.9.4.1 FILE_PREFIX override

FILE_PREFIX is initially set to the list of directories specified in the “acuauth.txt” file. If you define an alternate FILE_PREFIX in the configuration file that you embed with your object library, that value overrides the “acuauth.txt” setting. The COBOL program may use ACCEPT FROM ENVIRONMENT to get the initial value of FILE_PREFIX and add to it or reset it. APPLY_FILE_PATH may come in handy for existing programs. The override is generally useful only if you set FILE_PREFIX with remote name notation for use with AcuServer. If you specify local directories in FILE_PREFIX, please note that your program will have permission to access these directories only if users add them to the “acuauth.txt” file on their machines. If the files being accessed are indexed files, the Web runtime must first obtain a valid license, which may be either located on the client or obtained by AcuServer.

5.9.4.2 Editing the authorization file

If your COBOL application requires access to other directories or disks on a user's machine, the user must add those directories to the "acuauth.txt" file. For example, if the application requires access to the client directory, "c:\myfiles", the user would add "c:\myfiles" to the end of the first line in the "acuauth.txt" file as follows:

```
"C:\Program Files\Acucorp\Acucblxx\acugt\plgndata"
"c:\myfiles"
```

All directories must be included on a single line, separated by spaces. If a directory name includes embedded spaces, you must enclose it in quotation marks. You may always use quotation marks if you choose. Up to 4096 characters can be included on the line.

Note that all files and folders of the directories listed in this file become accessible.

5.9.4.3 Restricted library routines

By default, COBOL programs that are run via the Web runtime are not allowed to call the following library routines:

C$CHAIN
C$MEMCPY
C$RUN
C$SYSTEM
M$ALLOC
M$FREE
M$GET
M$PUT
REG-CREATE-KEY
REG-CREATE-KEY-EX
REG-DELETE-KEY
REG-DELETE-VALUE
REG-SET-VALUE
REG-SET-VALUE-EX
SYSTEM

Calling any of these routines results in a failure. If the CALL statement has no EXCEPTION phrase, the Web control displays the message "call_name: Access denied", where "call_name" is the name of the library routine and where "call_name" terminates the COBOL program. C$CALLERR returns error code 23 if access was denied for the previous CALL.

5.9.4.4 Using the authorization file for access

If you want to allow programs executed by the Web runtime to call a library routine that is normally disallowed, you should instruct your users to add a line to the "acuauth.txt" file containing the name of the routine. In addition, if you want to allow programs executed by the Web control to call DLLs, instruct your users to add a separate line containing the word "DLL".

5.10 Troubleshooting

This section describes messages that users may encounter when installing or using the Web runtime.

A plug-in for this file type was not found.

Users receive this message if they do not have the Web runtime installed on their machine when they try to launch your application and, for some reason, cannot install one automatically using the CODEBASE implementation. They should obtain a copy of the ACUCOBOL-GT Web Runtime through another means (product media or Web site, for example) and install it on their machine. The installation script automatically makes a file type association for ACUCOBOL-GT object files (i.e., ".acu" extension) with the Web runtime.

You can try reinstalling the control, running **regsvr32** to register it, or accessing an HTML document where the CODEBASE attribute is correctly implemented.

Access denied.

If the “acuauth.txt” file is empty or missing from the client machine when the Web control is invoked, the COBOL application is denied access to the local machine’s file system. The user should edit or create an “acuauth.txt” file that contains the names of all required directories.

Blank screen on install.

If users receive a blank screen from Microsoft Internet Explorer when they try to use a Web control, they may not have all the necessary DLLs copied into the appropriate directory. Users should place a copy of “wrun32.dll” and “acme.dll” in the directory that contains the control, along with the license file and “ajpg32.dll”, and then try to run the install again. These files are part of the CAB file distribution.

Another possible cause of blank screens is a misspelling of the WIDTH or HEIGHT attributes in an EMBED statement. This results in a silent failure, where the misspelled attribute is set to zero.

Error Locating Object Handler. There is no viewer available for the type of object you are trying to open…

Internet Explorer users receive this message if the required file, “acme.dll”, is missing from the directory that contains the Web runtime control. Users should place a copy of “acme.dll”, the executable (“acugtax.ocx”), the license file, and “ajpg32.dll” in the Web runtime directory, and then try to use the Web runtime again. These files can be found in the AcuGT\bin directory on the ACUCOBOL-GT distribution media, and they are part of the CAB file distribution. You can get more information using the Web runtime AcuGetLastError method.

5.11 Migrating from the Web Plug-in to the Web Runtime

If you have already implemented the Web Plug-in control to provide Web access to your ACUCOBOL-GT application, read this section for information on how to upgrade your implementation to the Web runtime. You can add the appropriate HTML code to make the new Web runtime available to your users.

The major task is to invoke the Web runtime component from your HTML page. If you are distributing the Web runtime internally to users with no Internet access, you may need to supply it on your own media.

Although it is possible to invoke your application with the <EMBED> and <ANCHOR> tags as you have for the plug-in, we recommend that you invoke it with the <OBJECT> tag and the component's CLASSID. For more information, see **section 5.6.1**.

Changing your HTML code

To add the Web runtime to your current implementation, the minimal steps you need to take are:

1. Create the <OBJECT> tag in your HTML documents to invoke the Web runtime (acugtax.ocx) control.
2. Remove the <EMBED> or <ANCHOR> tag currently used to launch the Web Plug-in. Provided you add the <OBJECT> element to your Web page and include the CODEBASE attribute, your users will automatically download the Web runtime the next time they visit the Web page.

Adding the <OBJECT> element there causes the user to install the Web runtime component. If your users can invoke the application from more than one Web page, you need to revise all the pages to use the <OBJECT> element.

6 Other Internet Solutions

Key Topics

6.1 LAN, WAN, or Internet

Because the Internet is a large TCP/IP network, you can design an Internet configuration using any of our networking or data access technologies. The Web runtime and CGI technologies were designed for communicating with Web browsers, but you can use our standard technologies to access applications and data across the Internet outside of the World Wide Web.

- Our AcuServer® remote file server can provide access to Vision data over the Internet.
- Our AcuConnect® remote application server can provide access to server-resident COBOL programs over the Internet, even programs that are distributed across a number of different servers.
- Our AcuXDBC™ interface can give users of ODBC and JDBC applications access to Vision data over the Internet. AcuXDBC is combined with AcuXDBC Server for remote processing of SQL requests.
- Our Acu4GL® interface and AcuSQL® precompiler can provide access to relational databases over the Internet.
- Our AcuXML interface or C$XML library routine can be used to provide access to XML documents over the Internet.

The procedures for using these technologies on the Internet are not much different from the procedures outlined in the product user's guides for local- and wide-area networks (LANs and WANs). Special Internet considerations are discussed in this chapter.

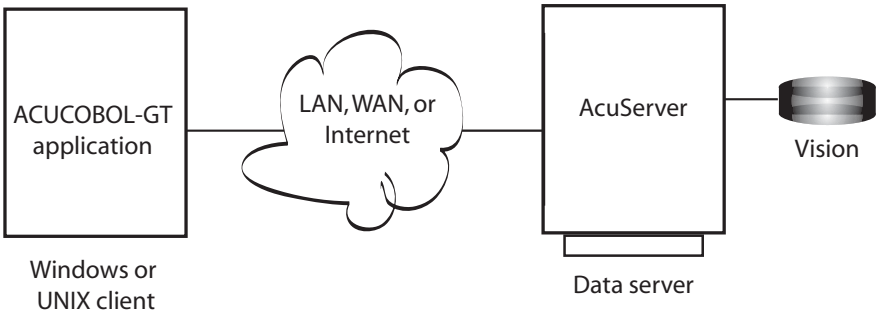
Please note that when accessing ***extend*** products on remote servers via the Internet or a virtual private network, products or technologies from third-party vendors such as Microsoft may be invoked. Carefully review any license agreements from these third-party vendors before proceeding with the remote connection.

6.2 Accessing Vision Data Over the Internet

AcuServer provides access to remote data and object files. Sites may choose to place files on a remote server because, for example, many users must access the files or the files may be updated frequently.

With AcuServer, ACUCOBOL-GT® programs on client machines have access to data and/or object files that are stored on remote servers and connected via TCP/IP. When the clients request data in a file system that resides somewhere in a network, AcuServer serves that request transparently, regardless of the client or server operating system.

A simple AcuServer environment looks something like this.



AcuServer configuration

As you can see from the illustration, the configuration is essentially the same whether you are using TCP/IP on a local- or wide-area network or using the Internet. This means that you can run your ACUCOBOL-GT application as usual on a Windows or UNIX client, store your data on a remote server, and access it over the Internet. AcuServer can give your users access to your Vision, relative, or sequential data and object files over the Internet as well as over a smaller network.

Using AcuServer is an easy way to take advantage of the Internet. In many cases, no reprogramming is required. You can indicate the location of the remote server through a runtime configuration variable. Your users do not require a Web browser to access remote data files.

Sample Scenario

You use AcuServer is to store your data in a central place. Your inventory file is updated frequently, and several people need to access the file over the course of the day. Having the file centrally located enables access by multiple users, with the current user having the most recent information. AcuServer supports record locking, so only one user at a time can edit a file. Finally, you may want your data or inventory files on a server that is backed up on a regular and frequent basis.

If you want to limit access to the inventory data, AcuServer can provide password protection for those instances where access to data must be restricted.

6.2.1 Internet Considerations for AcuServer

Although Internet implementations of AcuServer are configured the same as LAN or WAN configurations, there are some special considerations for providing remote file access over the Internet. For instance, how do you point to remote files over the Internet, and how do you keep your data files secure in this environment? This section provides the answers.

Note that end users must have a live Internet connection when they run the program that accesses AcuServer over the Internet. In addition, the server name in the client configuration file must be resolvable by the Internet name server used by their service provider. The Internet name server then resolves the name with its IP address. If the server name is not exposed to Internet name servers, you can enter the explicit IP address in the runtime configuration file on the client.

If the server is through a Virtual Private Network (VPN), the user must connect to the network before running the COBOL program.

For general information on setting up and using AcuServer, please see the *AcuServer User's Guide.*

6.2.1.1 Defining Internet pathnames

In both standard client/server and Internet configurations, enabling your applications to use AcuServer requires that your applications refer to remote files with *remote name notation*.

The easiest way to enable your applications to use AcuServer is to update the FILE_PREFIX and CODE_PREFIX runtime configuration variables to include paths with remote name notation. This holds true whether the remote path is a standard remote server address or an Internet address.

To add a remote search path, you simply append or insert the name and path of the remote directory to the variable's definition. For Internet configurations, you include the domain name of the AcuServer server that users are accessing over the Internet. For example, if the name of the server is "condor," you could define the configuration variables as follows:

```
FILE_PREFIX   @condor.XYZCorp.com:/usr/data
CODE_PREFIX   @condor.XYZCorp.com:/usr/objects
```

Of course, you could define "condor" as an alias for condor.XYZCorp.com, and simplify the remote name notation in these variables to "@condor:/usr/data" and "@condor:/usr/objects". This would not include the port number, but the port number is not always required.

If you want to enter the explicit IP address of the server, you may do so as shown in the following example:

```
FILE_PREFIX @128.110.121.42:/usr/data
CODE_PREFIX @128.110.121.42:/usr/objects
```

Alternatively, for highest performance, you can map the Vision filename directly to the Internet pathname by defining a file name alias in your runtime configuration file. A file name alias is a substitute string for the literal name that appears in the ASSIGN TO clause of a SELECT statement. Defining the alias shown below allows you to bypass FILE-PREFIX for "customer.dat":

```
CUSTOMER @condor.XYZCorp.com:/usr/data/customer.dat
```

By not requiring the runtime to connect to the server to see whether a file exists, you can open the file more quickly. For information on using name aliases, refer to section 7.2.4 of the *AcuServer User's Guide*.

6.2.1.2 Security and AcuServer

If you use AcuServer to give users access to data over the Internet, you will want to provide security measures to ensure that your data is safe from corruption or unauthorized access.

Setting up a firewall limits access to your data and enforces your organization's access control policy. When you set up a firewall, you'll need to indicate the "port number" through which applications gain access to your data. The default port number for AcuServer is 6523. You can indicate this setting through the ACUSERVER_PORT configuration variable.

Another way to secure your data is to encrypt it within your application before it is sent to AcuServer. Encryption provides an extra layer of security over and above the firewall your organization employs. Encryption is enabled with two configuration variables: ENCRYPTION_SEED and AGS_SOCKET_ENCRYPT.

AcuServer can also provide built-in password protection for access to files on the server. It provides a great deal of flexibility in assigning access permissions and password validation based on the client machine name, user name, or both. ACUCOBOL-GT runtimes provide built-in password-handling routines, or you can create your own password handling in your COBOL code using the *Acu-Client-Password* external variable.

In addition, AcuServer uses a security file called AcuAccess to support a wide range of access privileges, from very open to very restrictive. You choose the level of security best suited to your needs. The AcuAccess file is an encrypted Vision file. It contains one or more access records defining which users of which clients are permitted access to AcuServer. AcuAccess lets you specify individual user IDs that give users exactly the privileges they need, and no more.

6.3 Accessing COBOL Programs Over the Internet

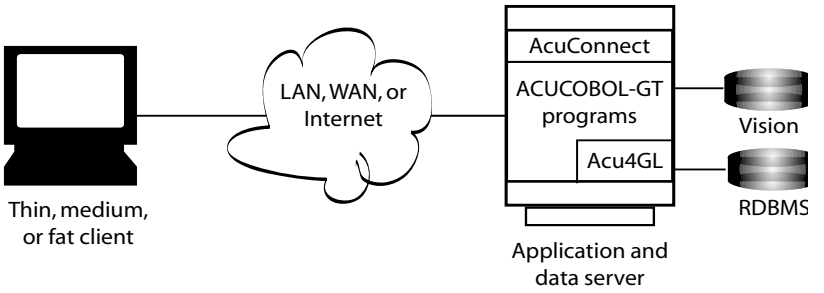
If you want users to be able to launch remote applications from their local machine, you can use our remote application server, AcuConnect, to accomplish your goal. Reasons for this may be that the application itself is revised frequently or the application is processing-intensive.

With AcuConnect, users or programs on client machines can launch applications on server machines, whether those servers are part of a local area network, wide area network, or global Internet. Some portions of your application can continue to run on the client while the resource-intensive portions run on the server, or all but the user interface can run on the server, in a thin client configuration.

In Chapter 3, we described how to launch a thin client from a Web page or over the Internet from the command line. But you can also use AcuConnect with the standard ACUCOBOL-GT runtime to distribute your application resources between the client and one or more servers on the Internet. Exactly how much processing is performed on the client and how much on the server is up to you.

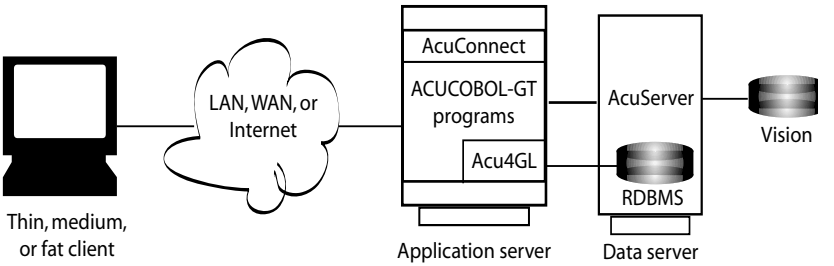
AcuConnect also provides users access to remote Vision data. If the data is on the same remote server as AcuConnect, data is considered local and no special software is required. If the data is on a third data server, AcuConnect can work in tandem with AcuServer to provide seamless data access. AcuConnect can also work with Acu4GL, our COBOL-to-RDBMS bridge, to provide access to relational data wherever it resides.

The following figures show AcuConnect in two-tier and three-tier environments. In a two-tier environment, applications and data are stored on the same remote server as follows:



AcuConnect in a two-tier environment

In a three-tier configuration, the data files reside on a different server than the application. Sites may do this for security purposes or because this may make for more efficient use of system resources. When the applications and data are on different servers, the data files are considered remote, and AcuServer is required.



AcuConnect in a three-tier environment

Consider using AcuConnect when your program is I/O-intensive or if you have reason to believe that the program will be updated frequently. You can determine how much reprogramming you want to do to facilitate remote application access for your users – either by embedding COBOL calls into your program or dividing it into client and server components. Your users do not require a Web browser for remote application access.

Sample Scenario

Your inventory program updates a file when products are sold, when products are returned, and when the supply is replenished. In addition, it interacts with a vendor's programs placing orders as supplies get low. In some circumstances, the program interacts with a payroll program to apply sales commissions to salaries. When performed over the network, these computationally intense activities are time consuming, and they sometimes produce bottlenecks. You decide to distribute the processing onto a fast application server that reduces network I/O and uses AcuConnect to launch the various program modules.

Because your inventory and payroll data is updated frequently, you decide to store it on a server that is devoted to data files. You use AcuServer to serve the requests for data. The user, at the client machine, starts an application that calls AcuConnect. The data files reside on a different server, so the files on the data server are accessed through AcuServer. After processing, the results are returned to the user via AcuConnect.

6.3.1 Internet Considerations for AcuConnect

Although Internet implementations of AcuConnect are configured the same as LAN or WAN configurations, there are special considerations for providing application and data access on the Internet: primarily, how you define the Internet application path and how you keep your applications and data secure in this environment. This section provides the answers.

Note that end users must have a live Internet connection when they run the program that accesses AcuConnect over the Internet. In addition, the server name in the client configuration file must be resolvable by the Internet name server used by their service provider. The Internet name server then resolves the name with its IP address. If the server name is not exposed to Internet name servers, you can enter the explicit IP address when configuring the client runtime.

If the server is accessed through a Virtual Private Network (VPN), the user must connect to the network before running the client program.

For general information on setting up and using AcuConnect, please see the *AcuConnect User's Guide*. For information on using the thin client or Web thin client on the Internet, refer to **Chapter 3** of this manual.

6.3.1.1 Defining an Internet application path

With AcuConnect, you define the pathname of your remote COBOL objects in a configuration file on the client. (This configuration file must be associated with the initial COBOL object.) In this file, you use the CODE_PREFIX variable to define the location of the object programs being CALLed.

In an Internet environment, the CODE_PREFIX variable should include the domain name of the AcuConnect server being accessed over the Internet. For example:

```
CODE_PREFIX *condor.XYZCorp.com:/usr/prog2
```

where *prog2* is the directory containing the ACUCOBOL-GT object code, and "condor" is the name of the application server running AcuConnect. If you define "condor" as an alias for condor.XYZCorp.com, you could simplify the remote pathname to "*condor:/usr/prog2". This would not include the port number, but the port number is not always required.

If you want to enter the explicit IP address of the server, you may do so as in the following example:

```
CODE_PREFIX *128.110.121.42:/usr/prog2
```

Notice that the name of the server is preceded by the asterisk character, "*". The asterisk indicates that the program is located on the server and that it should be run on the server as well. If you want the program to run on the client, you can simply change the "*" to "@". This emulates AcuServer notation.

When the client program executes a CALL, it determines which directory contains the program by looking in CODE_PREFIX, and then it executes the program on either the client or server as specified.

6.3.1.2 Security and AcuConnect

If you use AcuConnect to grant application access, you'll want to take the proper security measures to prevent unauthorized access.

One way to do this is with a firewall. When you set up a firewall, you'll need to indicate the "port number" through which clients gain access to the application on the server. The default port number for AcuConnect is 5632. You can indicate this setting through the ACURCL_PORT configuration variable.

In addition, AcuConnect uses a security file called AcuAccess to support a wide range of access privileges, from very open to very restrictive. You choose the level of security best suited to your needs. The AcuAccess file is an encrypted Vision file. It contains one or more access records defining which users of which clients are permitted access to AcuConnect. AcuAccess lets you specify individual user IDs that give users exactly the privileges they need, and no more.

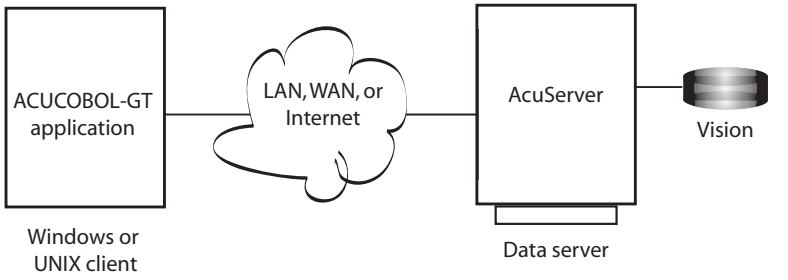
Every access record can include a password entry that the application or user must match before AcuConnect establishes a connection.

6.4 Accessing Vision Data from ODBC Applications

AcuXDBC can be configured to give users of popular Windows applications access to Vision data over the Internet. With AcuXDBC, you can retrieve and update ACUCOBOL-GT® indexed or relative data files from many Windows-based applications, including Microsoft Word, Excel, Access, Query, and Crystal Reports. These applications can access your files whether they reside on the same Windows PC, on a different computer in an enterprise network, or on a UNIX or Windows server in a LAN, WAN, or Internet configuration.

To gain access to remote Vision files over the Internet, you must have AcuXDBC Server installed on the server.

With AcuXDBC Server, the SQL query is processed on the server and just the result is returned to the client. This reduces network traffic and often leads to performance gains.



AcuXDBC w/AcuXDBC Server configuration, for remote SQL processing

Sample Scenario

Your inventory program is written in ACUCOBOL-GT and run on a UNIX server. The data is stored on the server in ACUCOBOL-GT's Vision file system. Frequently, field support managers create sales and inventory reports on their laptops using Seagate's Crystal Reports. While field reps use the thin client to interact with the inventory application on a daily basis, the managers use AcuXDBC to connect their Windows-based software with the COBOL data. Using the Windows application they are familiar with, they are able to import COBOL Vision data and create a business report in no time. If they need to work with any of the numbers in Microsoft Excel, they can do this as well. AcuXDBC works with any ODBC-compliant Windows application, including all the applications in Microsoft Office. They can even update records in the Vision file system if necessary.

All users need is a laptop, an Internet connection, and AcuXDBC. The server takes care of the rest.

6.4.1 Internet Considerations for AcuXDBC

Although Internet implementations of AcuXDBC are configured the same as LAN or WAN configurations, there are some special considerations for accessing data over the Internet. For instance, how do you point to remote files over the Internet, and how do you keep your data files secure in this environment? This section provides the answers.

Note that end users must have a live Internet connection when they try to import Vision data into their applications from the Internet. In addition, the server name indicated in the AcuXDBC configuration manager (described below) must be resolvable by the Internet name server used by their service provider. The Internet name server then resolves the name with its IP address. If the server name is not exposed to Internet name servers, you can enter the explicit IP address when setting up the AcuXDBC configuration.

If the server is through a Virtual Private Network (VPN), the user must connect to the network before running the COBOL program.

For general information on setting up and using AcuXDBC, please see the *AcuXDBC User's Guide.*

6.4.1.1 Defining Internet pathnames: AcuXDBC Server configuration

In an AcuXDBC Server configuration, where data is both accessed and processed remotely, you define the location of the server using the AcuXDBC configuration manager on the client machine, and then you define the data pathname when you set up the DSN on the server.

On the client machine, you define only the Data Source Name field, the name or IP address of the remote AcuXDBC Server host, and the port number on the General tab of the AcuXDBC DSN Setup screen. For Internet configurations, you might enter the following Host Name:

```
condor.XYZCorp.com
```

or IP Address:

```
128.110.121.42
```

6.4.1.2 Security and AcuXDBC

If you decide to use AcuXDBC to give users access to data over the Internet, you will want to provide security measures to ensure that your data is safe from corruption or unauthorized access.

Setting up a firewall limits access to your data and enforces your organization's access control policy. When you set up a firewall, you'll need to indicate the "port number" through which applications gain access to your data. To define the port number for AcuXDBC Server, use the dialog box. The default port number for AcuXDBC Server is 20222.

Another way to secure your data is to encrypt it within your application before it is sent to AcuXDBC Server. Encryption provides an extra layer of security over and above the firewall your organization employs.

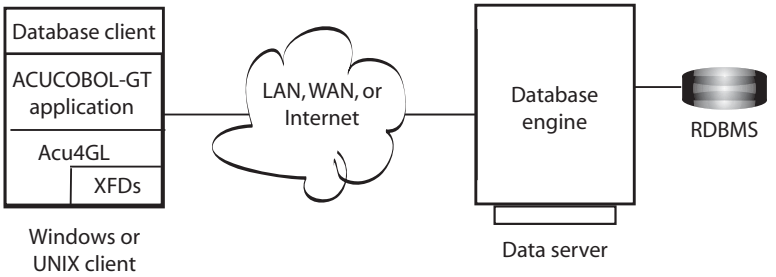
AcuXDBC Server uses typical relational database table permissions using the GRANT SQL syntax.

6.5 Accessing Relational Data Over the Internet

If you want users of ACUCOBOL-GT programs to access a relational database over the Internet, you can do so using our Acu4GL or AcuSQL technologies.

Acu4GL is a COBOL-to-RDBMS interface that automatically translates COBOL file I/O into database-specific SQL requests and vice versa. You do not need to know SQL to use this technology, because the translation occurs "behind the scenes" using a data dictionary map that is created at compile

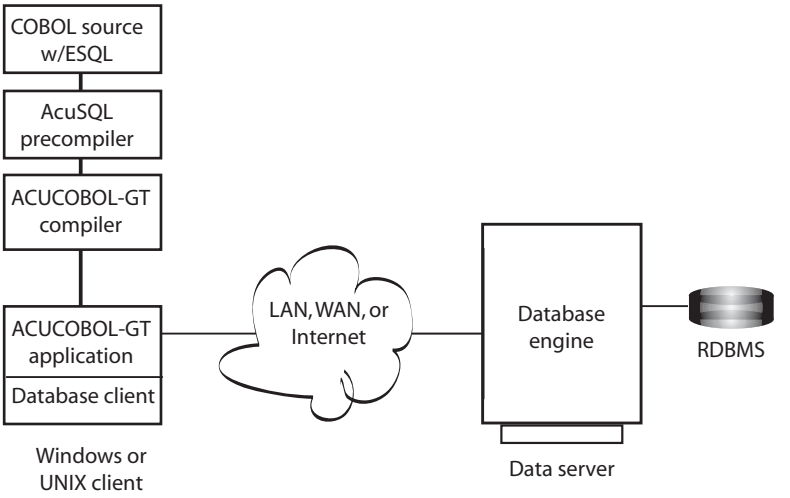
time. Typically, you don't even need to change your source code. With Acu4GL, COBOL users gain transparent access to RDBMS data in any TCP/IP network.



Acu4GL on the Internet

AcuSQL is an embedded SQL (ESQL) precompiler that lets you embed database-specific SQL directly into your ACUCOBOL-GT program. You control the precise fourth generation Structured Query Language commands that are sent to the database, and the precompiler automatically translates the

SQL into COBOL CALL statements that your program understands. Compiled again with the ACUCOBOL-GT compiler, your COBOL code is ready to access relational data wherever it resides.



AcuSQL in Internet environments

Each of these technologies gives users of ACUCOBOL-GT applications access to information stored in Relational Database Management Systems (RDBMS) such as Oracle, Informix, Sybase, and Microsoft SQL Server as well as ODBC-compliant data sources. The data can be stored on the same machine as the ACUCOBOL-GT application, or on a different machine in a TCP/IP network, including the Internet.

Sample Scenario

At one time, the data for your inventory program resided in a COBOL indexed file system. You decided to move your files to a centralized relational database—Oracle—that would give end users improved access to data and the ability to easily create ad hoc reports.

To provide access to the Oracle data from COBOL, you could do one of two things: embed SQL commands into your COBOL program and run it through the AcuSQL precompiler, or use the Acu4GL interface to translate

COBOL I/O statements into SQL instructions that the Oracle database understands. The first solution gives you more power and flexibility, but the second frees you from having to learn SQL and can be applied more quickly. You choose to use the Acu4GL interface.

Now, users of your inventory application can access Oracle data on any server in your TCP/IP network, even across the Internet. With a live connection to the Internet, users can launch the ACUCOBOL-GT program on their local machines or laptops, and access real-time inventory and distribution information on the designated server.

6.5.1 Internet Considerations for Acu4GL and AcuSQL

With relational database systems, the database manages remote connections. Therefore, when working with an RDBMS, you perform all remote path definition, including Internet path definition, using the database client software—for instance, the Oracle client or DB2 Connect client.

Please refer to your database guide for instructions on how to configure the client software for Internet environments. Pay close attention to database security features when providing data access across the Internet.

Refer to the *Acu4GL* or *AcuSQL User's Guides* for instructions on using these database technologies.

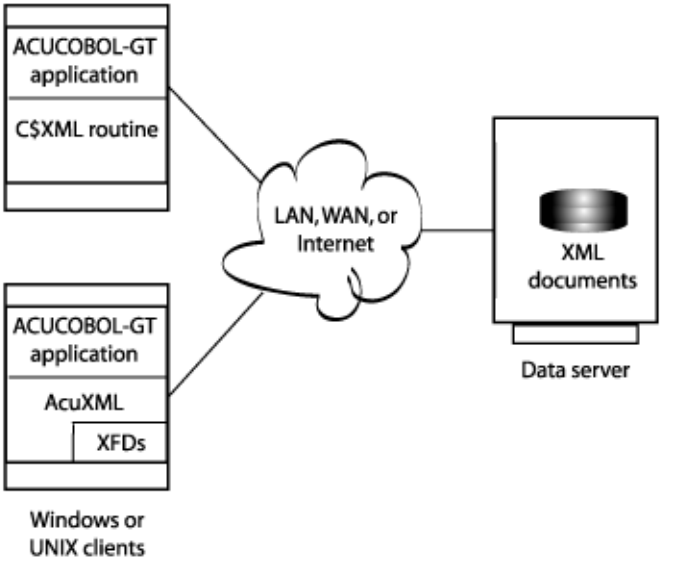
6.6 Accessing XML Data Over the Internet

There are two ways to access XML data from ACUCOBOL-GT:

1. Via the C$XML library routine, which gives you low-level control over the XML data that is parsed.

2. Via AcuXML, which provides a transparent file system interface between ACUCOBOL-GT applications and XML documents.

With either method, an ACUCOBOL-GT program can read whatever XML files a user indicates, and/or output data in XML format. It doesn't matter whether the data is in a LAN, WAN, or Internet configuration. Both methods support Internet notation for remote filenames.



Accessing XML data on the Internet

Although AcuServer can be used to provide remote file access to XML documents, it is not required. ACUCOBOL-GT applications can process XML streams directly on socket connections.

For more information on using AcuXML, please refer to the *ACUCOBOL-GT User's Guide*, section 6.11. For information on using the C$XML library routine, refer to Appendix I.

Sample Scenario

In your company, your inventory application is managed at your warehouse location. It is written in COBOL and running on a local Windows 2000 server. Data is stored in the Vision indexed file system. The customer

service application, on the other hand, is managed by the sales organization. It is written in Java and run on a UNIX server on the other side of the country. Data is output in XML for use in a Web-based self-service application.

Occasionally, your inventory application must access the customer service data for delivery status. In the past, the only way this was feasible was to create a redundant database in COBOL. Now, using AcuXML, you can connect the worlds of COBOL and XML, and eliminate the need for redundancy. And you can take advantage of the Internet to traverse your company's geographical boundaries.

With a live Internet connection, your end users can make a standard COBOL request, and a runtime configuration file on the client tells the application the location and type of data being requested—XML. AcuXML translates the XML to input that the COBOL program understands.

6.6.1 Internet Considerations for AcuXML and C$XML

Both AcuXML and C$XML make use of Internet notation to provide direct access to XML data on the Internet. The procedures for using Internet notation are provided in the next sections.

If you prefer to use AcuServer to access the XML files over the Internet, you can combine it with AcuXML or C$XML. In this case, you include the domain name of the file server in the FILE_PREFIX variable, but you are required to have AcuServer installed on the machine hosting the XML files. The procedure for using AcuServer is also provided below.

Whichever method you choose, make sure that the server name that you specify is resolvable by the Internet name server that will be used. Of course, end users must have a live connection to the Internet or VPN to make use of the remote files.

6.6.1.1 Using Internet notation with C$XML

To parse an XML file with the C$XML routine, you call C$XML using the CXML-PARSE-FILE op-code, and pass the filename and path of the XML document to parse. If the document is located on the Internet, you pass the filename with URL syntax. For example:

```
CALL "C$XML" using CXML-PARSE-FILE, "http://myserver.mycomp.com/xmldata/bookfile.xml"
```

Refer to Appendix I of the ACUCOBOL-GT manual set for more information on using this routine.

6.6.1.2 Using Internet notation with AcuXML

With AcuXML, the file path is specified in the runtime configuration file, "cblconfig."

To read a file from the Internet, you set up a filename alias in the runtime configuration file, and use Internet notation to map the data files directly to a URL. For example, to read "bookfile.xml" over the Internet, you might define a name alias for "bookfile" in your configuration file like this:

```
BOOKFILE  http://myserver.mycomp.com/xmldata/bookfile.xml
```

Whenever your application refers to "bookfile" in a SELECT statement, the alias "http://myserver.mycomp.com/xmldata/bookfile.xml" is substituted.

To specify that "bookfile" should be treated as XML data for translation by AcuXML, you must also set the *filename*_HOST variable to "XML" as follows:

```
BOOKFILE_HOST XML
```

With AcuXML, you include a separate entry for each XML document name.

Please refer to Book 1, section 2.7.1 of the ACUCOBOL-GT manual set for more information on setting up file aliases. Refer to Book 1, section 6.11.4 for more information on creating configuration files for use with AcuXML.

6.6.1.3 Using AcuServer with AcuXML or C$XML

To access remote XML files using AcuServer, you use the FILE_PREFIX configuration variable to define the location of the remote files. You simply append or insert the name and path of the remote directory to the variable's definition.

For Internet configurations, you include the domain name of the server that users are accessing over the Internet. (The remote server that contains the XML documents must be running AcuServer.) For example, if the name of the server is "condor," you could define the configuration variables as follows:

```
FILE_PREFIX   @condor.XYZCorp.com:/usr/data
```

Of course, you could define "condor" as an alias for condor.XYZCorp.com, and simplify the remote name notation in this variable to "@condor:/usr/data". Or if you want to enter the explicit IP address of the server, you may do so as in the following example:

```
FILE_PREFIX @128.110.121.42:/usr/data
```

If you are using AcuServer with AcuXML, you must use the *filename*_HOST variable to designate the data files as XML files, as in the following example:

```
file1-HOST XML
file2-HOST XML
file3-HOST XML
```

This lets the ACUCOBOL-GT runtime know to use the AcuXML file interface to translate the XML input for the COBOL application.

6.6.1.4 Security and XML

If you decide to use AcuXML or C$XML to give users access to XML data over the Internet, you will want to provide security measures to ensure that your data is safe from corruption or unauthorized access. If you are using AcuServer, you can take advantage of AcuServer's access privileges and password protection to aid in the process. Refer to the *AcuServer User's Guide* for more information.

A Building and Hosting a Web Site

Key Topics

A.1 Setting Up a Web Site

A Web site is typically comprised of an HTML document or set of HTML documents and the Web server software. Sometimes it contains XML documents and PDF files instead of or in addition to the HTML documents.

How exactly you create a Web site depends largely on the tool you choose. For this reason, this chapter will not try to exhaust the subject. It will simply provide some guidelines and general instructions to help orient you to your task. This chapter divides the task of setting up a Web site into seven main steps:

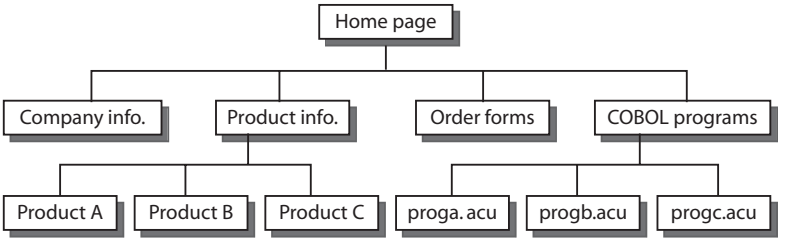
1. **Designing Your Site**
2. **Finding a Host or Building a Web Server**
3. **Creating Your Web Pages**
4. **Creating a Link to COBOL Programs**
5. **Posting Your Web Documents**
6. **Promoting Your Site**
7. **Registering a Domain Name**

A.2 Designing Your Site

The first step to setting up a Web site is to design the site. What kind of material do you want to publish on the Web? Do you plan to include company information…product information…order forms…applications…data? How do you want your home page to look? With most Web authoring tools, you can choose from a wide array of templates, customize a template, or design your own page from scratch. The best way to decide on your design is to surf the Web. Which sites do you find to be the most effective? Which ones grab your interest and compel you to read on, compel you to buy?

If you're designing a site from scratch, you may find it useful to make a thumbnail sketch of your home page. Once you have the basic design of your home page, you need to decide what subsequent pages to include and how to

navigate your visitor to those pages. It is often helpful to create a flow diagram similar to the one you create when designing an application. For instance:



Once you have a good idea of how you want the site to look and flow, you're ready to move to the next step.

A.3 Finding a Host or Building a Web Server

Before you create your site, you need to find or create a Web server to host it. Initially you might decide to have an Internet service provider (ISP) host the site. ISPs provide space on their Web server for your site and charge for it based on the type of information that your site will contain and the amount of disk space and network I/O that it requires. The best way to learn about ISPs is to check computer circulars, computer stores, or even your local Yellow Pages. Once you find an ISP that serves your needs, you simply set up an account with that provider and you’re ready to go.

To build your own Web server, you must make a minor investment in hardware and software, and you should prepare to spend hundreds or thousands each month for a dedicated connection to the Internet. You must also be prepared to dedicate time and personnel to the cause.

A.3.1 Selecting Web Server Software

If you plan to build your own Web server, you must select the operating system and Web server software to use. Web servers can operate on a variety of operating systems, the most popular being Windows and UNIX. When

setting up your Web server, be sure to select the operating system that you are most comfortable with. Whichever operating system you choose, you will have a variety of packaged Web server suites to choose from. Both Netscape and Microsoft offer complete Web server suites for supporting applications and data on the Internet.

Once you install the Web server software on the machine you intend to use as a Web server, you may also need to configure the Web server for your particular deployment method. Instructions for configuring the Web server can be found in the following sections:

Deployment Method	Information on Web Server Configuration
HTML/CGI	**Section 4.7, "Configuring the Web Server"**

A.4 Creating Your Web Pages

To create a Web page, you can use any of several HTML or XML authoring tools or you can use a simple programmer's editor. Some authoring tools come bundled with Microsoft Office others are available over the Internet free of charge. Many tools let you customize a site template so you don't have to begin from scratch. Most provide a WYSIWYG (what-you-see-is-what-you-get) screen painter-type interface, so you don't even have to learn the markup language.

If you choose to create the more dynamic XML documents, you will have to "publish" the documents in HTML or PDF form before a Web browser can display them. XML documents are typically transformed to HTML or PDF through the use of style sheets and a style sheet transformation language (XSLT). XML tools like the Cocoon Project from Apache provide a Web publishing framework for creating XML-based Web sites.

Because every Web authoring tool is different, this section cannot provide specifics. It can, however, provide some guidelines. When authoring your Web pages, you should:

- Create a separate folder to hold your HTML, XML and PDF documents. This will make it much easier when it comes time to "post" the site. It is also a good idea to separate logical sections into separate subdirectories.
- Work from an existing template if available.
- Start with your home page, then move to subsequent pages.
- Perform all formatting before including clickable image maps or setting up links.
- If you are working with an ISP, find out what name to save your home page under, such as "index.html" or "index.htm".
- Keep it simple. You can always edit your files later to add more "glitz."
- Manage your HTML and XML files the same way you manage your COBOL source code files. (After all, they are plain text like the source code files you work with daily.) You can even use version control tools.
- Use server-side includes, SSI, whenever possible to reduce redundancy on your Web site. For example, suppose you want to include your e-mail address at the bottom of every page. Without SSI, if your site contains 100 Web pages and your e-mail address changes, you need to edit 100 individual files. With SSI you simply edit the "included" file containing your e-mail address.

A.5 Creating a Link to COBOL Programs

Once you've created your HTML or XML documents, you can easily establish a link to a COBOL program, whether it is your main application, a client or server subprogram, or a CGI program. In most cases, this involves using an HTML source editor or word processor to edit the HTML document you created with your authoring tool. You can use this same editor to "tune" your HTML tags later, once the site is operational and you have enhancement requests.

The procedure for establishing a link to COBOL programs varies with Web server platform and Web deployment method. You'll find instructions for your method in the appropriate chapter in this book.

Method	Where to find information on linking COBOL programs
Thin client	**Section 3.5.2, "Using Anchor Tags"**
Web runtime	**Section 5.6, "Invoking Your COBOL Application with the Web Runtime"**
Helper application/viewer	**Section C.5, "Creating a Link to Your COBOL Object"**

A.6 Posting Your Web Documents

Once you've created your Web document, you need to "post" it to your Web server to make it available on the World Wide Web. Posting is nothing more than uploading or copying the documents onto the Web server.

If you have created HTML files, you can post the files directly. If, however, you have created XML files, you must transform the files to HTML, PDF, or WML before posting them. You do not post the XML page, but rather a published form of the page.

Authoring tools like the one that comes with Netscape Navigator/ Communicator provide "one-button publishing," meaning you can post your entire site to an ISP's server with the click of a button. If this is not available, you can use FTP to post your site. Your ISP will provide instructions.

If you have your own Web server, posting your site simply involves copying the files over to the server machine. Your URL is formed from the path to the files relative to the server's root directory. Your Web server software will provide instructions.

A.7 Promoting Your Site

If you are using your new Web site for business use, you will no doubt want to promote the site once it's up and running. Since most people use search engines to find sites on the WWW, it is a good idea to add your site to several prominent search databases. To do this, you can navigate to the site, www.submit-it.com. There you specify which search databases to submit your information to (e.g., Yahoo, WebCrawler, and InfoSeek), then you define keywords, categories, and your URL. Submit-it lets you submit your information to many databases at the same time, so no matter which search engine users use, they have a good chance of discovering your site if they enter your keywords.

A.8 Registering a Domain Name

After you create your Web site, you can register a domain name for the site with the Internic. A domain name associates your site with a specific Internet Protocol (IP) address. Web site domain names, such as "www.company-name.com" provide a series of benefits leading to the success of your Web site.

For instance, by registering a domain name, you can move your Web site anywhere on the Web and your visitors can access the new location the same as always simply by updating the registered address. If you plan to include your Web site in the many search engines, a registered domain name allows you to move your Web site without having to update the search engine entries. Also, a domain name that matches, or closely matches, your company name makes it easy for visitors to locate your Web site.

To register a domain name, talk to your Internet service provider or visit www.InterNIC.com.

B Adding Internet Features to Your Program

Key Topics

B.1 WEB-BROWSER Control

The ACUCOBOL-GT® Development System includes an Internet-related graphical control called WEB-BROWSER. This control broadens the usefulness and scope of your COBOL programs by giving them all of the capabilities of Microsoft Internet Explorer's Web browser control Version 4.0 and later (this control is designed for Windows users only). For instance, the WEB-BROWSER control lets you:

- Facilitate seamless Web browsing from your COBOL application.
- Display Web pages containing HTML, scripting, ActiveX controls and Java applet content.
- Display HTML pages distributed with your COBOL application.
- Include a variety of graphical and multimedia file types in your COBOL application.
- Invoke e-mail, telnet, and FTP services from your COBOL application.
- Display word processing, accounting, or presentation documents from your COBOL application.
- Display Windows objects such as folders and files from your COBOL application.
- Display Windows dialog boxes such as "Print," "Print Preview," and "Page Setup," allowing users to print the contents delivered by the control.
- Display the Windows "Save As" dialog box allowing the user to save the current control content to a file.
- Perform "Select All" and "Copy" clipboard operations.

ACUCOBOL-GT's WEB-BROWSER control is used just as any other graphical control in ACUCOBOL-GT is used, except that it opens a resource such as an HTML page, graphical image, video, audio, e-mail program, file folder or any other resource that a Web browser can open. When you include

the WEB-BROWSER control in your source code, your application launches Microsoft Internet Explorer (MS IE) on your user's machine and displays the resource you specified.

Note: For the control to work, your users must have Microsoft Internet Explorer Version 4.0 or later on their machine.

You specify the resource using standard URL naming conventions with any protocol that Internet Explorer recognizes, such as http:, ftp:, mailto:, file:, or javascript:.

For example to display our Web page on the user's machine, you might create a WEB-BROWSER control identified by the name BROWSER-1, then add the following lines to a Screen Section item:

```
03  BROWSER-1 WEB-BROWSER, VALUE "www.acucorp.com"
    COLUMN 5, LINE 5, SIZE 60, LINES 20.
```

or add the following Procedure Division code:

```
DISPLAY WEB-BROWSER, VALUE "www.acucorp.com"
COLUMN 5, LINE 5, SIZE 60, LINES 20.
```

You can then close the screen, close the window, or destroy the control as you would destroy any other ACUCOBOL-GT control.

The VALUE of the WEB-BROWSER control determines which resource to display. Therefore, if you include:

```
DISPLAY WEB-BROWSER, VALUE "mailto:info@acucorp.com".
```

your application launches your user's default e-mail program and creates a new message with the "To:" field prefilled. This allows your user to quickly compose and mail a message to the specified address.

The default settings used by Internet Explorer when it performs these Web browser functions (such as the "home page," security options, e-mail program) are included in the user's control panel under the Internet option. Users who want to alter the behavior of the browser controls should modify their control panel Internet settings.

As with any ACUCOBOL-GT control, you can add the WEB-BROWSER control to user interface screens using the AcuBench® Screen Designer. The control can be found on the screen design control palette along with the other controls. Click the control, fill in its value clause, and the Screen Painter will show you what the screen will look like to the user.

Your WEB-BROWSER control can be as simple or involved as you like. In its simplest form, the WEB-BROWSER control is activated in a single DISPLAY statement. If desired, you can specify the size and color of the resulting browser window; you can activate the browser control's back, forward, home, and search buttons; or you can capture events such as download progress, status bar text changes or resource title change so you can communicate them to your user.

For information on the methods, properties, events, and event properties of the WEB-BROWSER control, refer to Chapter 3 of ACUCOBOL-GT Book 2: *User Interface Programming.*

B.1.1 Adding Web Browsing to Your COBOL Applications

To add Web browsing to your COBOL applications, you include the Web address (or URL) in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example:

```
display web-browser, value "www.acucorp.com"
   column 5, line 5, size 60, lines 20.
```

Using this method, your COBOL program can display Web pages containing HTML, scripting, ActiveX controls, Java applet content and more.

Alternatively, your COBOL program can send Internet Explorer to the user's predefined home page (the one defined in the user's control panel, Internet Properties setting). To do this, you use the GO-HOME feature of the WEB-BROWSER control, as in:

```
display web-browser, go-home = 1
```

Or, if you want to enable your users to search the Web, you can send the browser to Microsoft's Web portal site using the GO-SEARCH feature, as in:

```
display web-browser, go-search = 1
```

If desired, you can dynamically change the browser's location using the MODIFY verb. For instance, if your program displays a combo-box offering many different URL choices, you may need to MODIFY the location of the browser based on a user response (mouse click).

```
Modify web-browser, value  "www.cobol.com"
```

B.1.2 Displaying HTML Pages Distributed With Your Application

To display HTML pages distributed with your application, you include the HTML filename in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example:

```
display web-browser, value "myfile.htm".
```

This opens and displays the file "myfile.htm" inside the WEB-BROWSER control.

Using this method, you can display an HTML forms-based front end for your application. To pass user input values from the HTML form to your COBOL application, you must write a CGI program as described in **section 4.5, "Writing a CGI Program."** Then the HTML program can interact with the CGI program on the Web server, and the CGI program can interact with your COBOL code.

Note: If you want the <Enter> key to be active on an HTML page, add the USE-RETURN style to the WEB-BROWSER control. This permits users to press <Enter> after filling out a form to submit the form. If you do not add the USE-RETURN style, pressing the <Enter> key will normally terminate input (e.g., generate a button-pushed event for the default push-button).

B.1.3 Including Graphical and Multimedia Files in Your Applications

Using the WEB-BROWSER control, you can include many types of files in your COBOL applications, including those with compressed graphical formats, compressed sound formats, movie-file formats, animated GIF formats, PDF formats, and more.

To include a graphical or multimedia file in your program, you include the filename in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example:

```
display web-browser, value "movie1.avi".
```

This example would launch the user's default movie viewer and play the movie on the user's screen.

B.1.4 Invoking e-mail, telnet, and FTP Services From Your Applications

To invoke e-mail, telnet, or FTP services from your COBOL application, you include a standard mailto, telnet, or FTP path in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example, to open the user's default e-mail program and create a new message addressed to "info@acucorp.com", you would include:

```
display web-browser, value "mailto:info@acucorp.com".
```

To open the login screen to a server called "sun" over telnet, you would include:

```
display web-browser, value "telnet://sun".
```

If desired, you can even use this interface to log on to a UNIX system from your Windows environment.

To go to the FTP site "ftp://ftp.acucorp.com", you would include:

```
display web-browser, value "ftp.acucorp.com".
```

This opens a directory view of Acucorp's FTP server.

B.1.5 Displaying Word Processing, Accounting, or Presentation Documents From Your Applications

To display word processing, accounting, or presentation documents that have been distributed with your COBOL program, you include a path and filename in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example, you can "embed" a Word document, Excel spreadsheet, or PowerPoint slide presentation in your application, as in the following example:

```
display web-browser, value "c:\presentations\training.ppt".
```

This example opens PowerPoint inside the browser on the user's machine and displays the presentation named "training.ppt". Note that the menus and toolbars normally available to PowerPoint users do not appear in the browser. Users can access a subset of menu options by right-clicking in the control.

B.1.6 Displaying Windows Objects Such as Folders and Files

To display Windows objects such as directory folders and files (as in the Windows Explorer directory structure), you include a pathname in the VALUE clause of the DISPLAY WEB-BROWSER statement. For example:

```
display web-browser, value "C:\Program
Files\Acucorp\Acucbl800\acugt\bin".
```

This displays the folders and files contained in the "\acucorp\acucbl5x\acugt\bin" directory on the user's local machine in a manner similar to Windows Explorer. The user can open files by double-clicking them. To view the options normally displayed in the Windows Explorer menu bar or toolbar, users can right-click in the control. They will see a pop-up menu that contains many useful options for browsing directories.

B.1.7 Performing Print, File, and Clipboard Operations

The WEB-BROWSER control includes special properties that allow users to perform standard Windows print, file, and clipboard operations from your COBOL application.

There are five special properties relating to the Windows Print function. They are designed to let users print the contents of the WEB-BROWSER control (i.e., the Web page that is being displayed). These are:

Special Property	Function
PRINT	Displays the “Print” dialog
PRINT-NO-PROMPT	Prints without displaying a dialog first
PRINT-PREVIEW	Displays the “Print Preview” dialog
CUSTOM-PRINT-TEMPLATE	Defines the name and location of the custom template to use for printing and print previewing. (For use with IE versions 5.5 and later)
PAGE SETUP	Displays the “Page Setup” dialog

To display the “Print” dialog so that users can choose a printer, page range, and so on, set the Print property equal to “1” as such:

```
display web-browser PRINT = 1.
```

To assign a custom print template to use when printing with the dialog, add a pathname to the CUSTOM-PRINT-TEMPLATE property, then set PRINT equal to “1”. For example:

```
display web-browser CUSTOM-PRINT-TEMPLATE = "\templates\mytemplate.html", PRINT = 1.
```

In addition to the print properties, the WEB-BROWSER control has several special properties relating to the Windows file and clipboard operations. These are:

Special Property	Function
COPY-SELECTION	Copies current selection to clipboard
CLEAR-SELECTION	Clears current selection from clipboard

Special Property	Function
COPY-SELECTION	Copies current selection to clipboard
SELECT-ALL	Selects the active HTML page, frame, or entry field, depending on cursor location
SAVE-AS	Displays the “Save As” dialog
SAVE-AS-NO-PROMPT	Saves frame contents to disk without prompting (for early version of IE only)
FILE-NAME	Defines path of file to be used in Save As operation
PROPERTIES	Displays the “Properties” dialog

To copy the current selection to the clipboard, set the COPY-SELECTION property to “1” as shown below:

```
display web-browser, COPY-SELECTION = 1.
```

For complete information on the WEB-BROWSER control special properties, refer to Book 2, section 5.19.2 of the ACUCOBOL-GT manual set.

B.1.8 Sample Web Browser Program

The following sample program, “browser.cbl”, demonstrates usage of the ACUCOBOL-GT WEB-BROWSER control. Additional samples have been provided in the sample directory on your ACUCOBOL-GT distribution media. See “webbrows.cbl” for another useful example.

```
identification division.
program-id.  Browser.

* Copyright (c) 1988 - 2003 by Acucorp, Inc.  Users of ACUCOBOL
* may freely modify and redistribute this program.

remarks.
This program illustrates the WEB BROWSER control type.

*****************************************************************
data division.
working-storage section.
copy "acucobol.def".
```

```
copy "acugui.def".

77  key-status
         is special-names  crt status    pic 9(4) value 0.
         88  exit-button-pushed          value 27.

01  event-status
         is special-names event status.
    03  event-type                       pic x(4) comp-x.
    03  event-window-handle              usage handle.
    03  event-control-handle             usage handle.
    03  event-control-id                 pic x(2) comp-x.
    03  event-data-1                     usage signed-short.
    03  event-data-2                     usage signed-long.
    03  event-action                     pic x comp-x.

78  event-occurred                       value 96.
78  go-btn-pressed                       value 707.
78  back-btn-pressed                     value 708.
78  forward-btn-pressed                  value 709.
78  home-btn-pressed                     value 710.
78  refresh-btn-pressed                  value 711.
78  search-btn-pressed                   value 712.
78  stop-btn-pressed                     value 713.
77  ef-url                               pic x(1000).
77  wb-1-url                             pic x(1000).
77  wb-1-title                           pic x(100).
77  wb-1-status                          pic x(100).
77  wb-1-progress                        pic 9(7).
77  wb-1-max-progress                    pic 9(7).
77  progress-percent                     pic 9(9).

77  gt-bitmap                            pic s9(9) comp-4.

01  configuration-data.
  05 current-lines                       pic s99999V99 value 25.
  05 current-size                        pic s99999v99 value 73.


*****************************************************************
screen section.
01  screen-1.

    03  entry-field, column 5, line 4, size 55 max-text = 0
        value ef-url.

    03  push-button, "&Back",
        column 5, line 2, size 9
        self-act
```

```
    termination-value = back-btn-pressed.

03  push-button, "&Forward",
    column + 2, size 9
    self-act
    termination-value = forward-btn-pressed.

03  push-button, "&Home",
    column + 2, size 9
    self-act
    termination-value = home-btn-pressed.

03  push-button, "&Refresh",
    column + 2, size 9
    self-act
    termination-value = refresh-btn-pressed.

03  push-button, "&Find",
    column + 2, size 9
    self-act
    termination-value = search-btn-pressed.

03  push-button, "&Stop",
    column + 2, size 9
    self-act
    termination-value = stop-btn-pressed.

03  push-button, "&Go", default-button
    column 55.2, line 4,
    termination-value = go-btn-pressed.

03  status-frame-1 frame, line 22, column 5,
    lines 2 size 65 cells lowered.

03  status-text-1 label, line 22.5, column 5.5,
    size 64 cells
    value wb-1-status.

03  progress-meter-1 frame line 24 column 5
    lines 1.5 size 24
    fill-color = red, fill-color2 = white,
    fill-percent = 0, lowered.

03  exit-1 push-button, "E&xit",
    cancel-button, line 24, column 32, size 11.

03  busy-bitmap bitmap bitmap-handle = gt-bitmap,
    size 39, bitmap-start = 1, bitmap-end = 15,
    bitmap-timer = 0,
```

```
        line 2, column 65.

    03  wb-1 web-browser
        column 5, line 6,
        lines 16 cells, size 65 cells
        event procedure is browser-event-handler.

*****************************************************************
procedure division.
main-logic.
    display standard window,
         title "Web Browser Sample - browser.cbl"
         lines current-lines, size current-size,
         resizable
         background-low.

    call "w$bitmap" using wbitmap-load, "gtanima.bmp",
         giving gt-bitmap.

    display screen-1.

    perform, with test after, until exit-button-pushed
         accept screen-1
         evaluate key-status
           when go-btn-pressed
             move ef-url to wb-1-url
             modify wb-1 value=wb-1-url
           when back-btn-pressed
             modify wb-1 go-back=1
           when forward-btn-pressed
             modify wb-1 go-forward=1
           when home-btn-pressed
             modify wb-1 go-home=1
           when search-btn-pressed
             modify wb-1 go-search=1
           when refresh-btn-pressed
             modify wb-1 refresh=1
           when stop-btn-pressed
             modify wb-1 stop-browser=1
           when event-occurred
             if event-type = ntf-resized
               divide event-data-1 by 100 giving current-lines
               divide event-data-2 by 100 giving current-size
               modify wb-1
                 lines current-lines - 9
                 size current-size - 8
               modify status-frame-1
                 line current-lines - 3
                 size current-size - 8
```

```
                modify status-text-1
                  line current-lines - 2.5
                  size current-size - 9
                modify progress-meter-1
                  line current-lines - 1
                modify exit-1
                  line current-lines - 1
              end-if
          end-evaluate
      end-perform.
      stop run.

  browser-event-handler.
      evaluate event-type
          when msg-wb-navigate-complete
              inquire wb-1 value in wb-1-url
              if wb-1-url is not = ef-url then
                  move wb-1-url to ef-url
                  display screen-1
              end-if
          when msg-wb-progress-change
              inquire wb-1 progress in wb-1-progress
              inquire wb-1 max-progress in wb-1-max-progress
              move wb-1-progress to progress-percent
              multiply 100 by progress-percent
              divide wb-1-max-progress into progress-percent
              if progress-percent = 100
                  move 0 to progress-percent
              end-if
              modify progress-meter-1,
                  fill-percent = progress-percent
              if progress-percent = 0
                  modify busy-bitmap bitmap-timer = 0
                      bitmap-number = 1
              else
                  modify busy-bitmap bitmap-timer = 10
              end-if
          when msg-wb-status-text-change
              inquire wb-1 status-text in wb-1-status
              display status-text-1
          when msg-wb-title-change
              inquire wb-1 title in wb-1-title
              display wb-1-title upon global window title
      end-evaluate.
```

C Use the Runtime as a Helper Application or Viewer

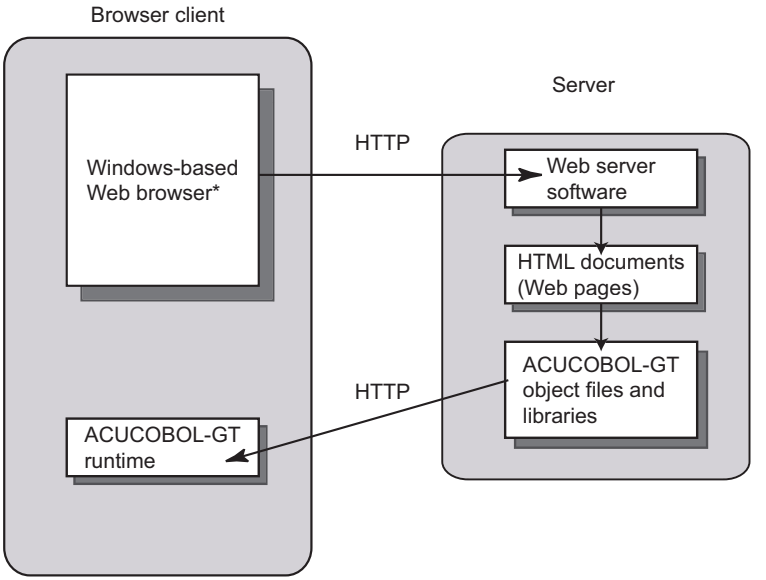
Key Topics

C.1 What Are Helper Applications and Viewers?

If your users already have a licensed copy of the ACUCOBOL-GT® runtime on their machine and they want to be able to access COBOL applications on a Web site, one way to do this is to set up their runtime as an Internet helper application or viewer inside their browser.

Helper application and *viewer* are browser terms referring to user-based software that can read and process files of a given type. Netscape uses the term "helper application" to refer to such software. Microsoft Internet Explorer uses the term "viewer."

Since the ACUCOBOL-GT runtime has the ability to read and process COBOL objects, it allows users to run COBOL programs that they encounter when browsing your Web page.



The main difference between this and a standard runtime configuration is that your COBOL object files are on the Web server and transmitted to the client machine via HTTP.

Keep in mind that the helper application/viewer is a fully-featured runtime; it has full access to your users' machines, including system calls, memory, disk and network access. Therefore, it should be used only with programs from trusted sources, or in conjunction with the Internet security you have in place.

Note: If your users do not already have an ACUCOBOL-GT runtime installed, a thin client or Web runtime may be a better option for them. These client technologies are available on Acucorp's Web site, and they provide additional capabilities designed for accessing remote COBOL resources in Internet environments. Refer to **Chapter 3** and **Chapter 5** for more information on these Web deployment methods.

Sample Scenario

The runtime as a helper application makes it very easy for your mobile work force to "stay in touch." With the runtime running as a helper application on their laptops, remote staff can access the appropriate page on your Web site and click on the link to start your inventory program on their local machines. To access Vision data on the server, the runtime makes use of the AcuServer® file system interface. This enables the sales person to access the inventory data files and update them with new information from a remote location.

C.2 Deploying Applications with the Runtime as a Helper Application or Viewer

To deploy your application on the Web using the helper application/viewer method, you (the developer) have three tasks, described in the following sections:

1. **Setting Up a Web Site**
2. **Preparing Your ACUCOBOL-GT Application**
3. **Creating a Link to Your COBOL Object**

Once your work is done, runtime users also have two tasks, described in these sections:

1. **Defining the Runtime as a Helper Application or Viewer**
2. **Launching the Application**

C.3 Setting Up a Web Site

Appendix A gives general information about setting up a Web site, including information on Web servers, posting a site, and promoting a site. Many different tools are available to help you create a Web page quickly and easily. Refer to **Appendix A** for guidelines.

C.4 Preparing Your ACUCOBOL-GT Application

There are no special coding considerations for using this method of Web deployment. However, runtime configuration and packaging do require some thought.

C.4.1 Configuring the Runtime

When used as a helper application or viewer, the ACUCOBOL-GT runtime is invoked to execute a single file. If your application contains more than a single object file, it must be packaged in a library file that "packages" all of the objects and associated resources of your application into a single file (see **section C.4.2, "Packaging Your Application and Resources."**)

There are two ways to configure the runtime for use as a helper application:

- Programmatically, using the SET CONFIGURATION or SET ENVIRONMENT variables, or
- Bundling a configuration file into the library file along with the COBOL object files and resources.

If you choose the library file method and the library contains a configuration file, the client machine must be set up to specify the name of the configuration file when users define the runtime as a helper application in their browser. Be aware, however, that this will affect all COBOL applications that your users run as a helper application. (See **section C.6.1, "Defining the Runtime as a Helper Application or Viewer,"** for instructions.)

If you choose to configure the runtime programmatically, you should use the SET CONFIGURATION (or SET ENVIRONMENT) phrase in your source code.

For example, if your application uses a configuration file with the following entries:

```
FILE_PREFIX @hal:/u2/serverfiles
COMPRESS_FILES 1
KEYSTROKE EDIT=PREVIOUS EXCEPTION=52 kl
```

Add the following lines to your COBOL initialization code:

```
SET CONFIGURATION "FILE_PREFIX" TO "@hal:/u2/serverfiles".
SET CONFIGURATION "COMPRESS_FILES" to "1".
SET CONFIGURATION "KEYSTROKE" TO "EDIT=PREVIOUS EXCEPTION=52 kl"
```

Note that MAX_FILES, MAX_LOCKS, and LOCKS_PER_FILE cannot be modified using the SET verb and that the COBOL program can read environment variables using ACCEPT FROM CONFIGURATION (or ACCEPT FROM ENVIRONMENT).

C.4.2 Packaging Your Application and Resources

In order to be used as a helper application or viewer, your COBOL application must be packaged in a single library file that contains all of the necessary COBOL objects, bitmap resources, and if desired, a configuration file. If you will be interfacing your application with a relational database through Acucorp's Acu4GL® product, you can package the ".xfd" files into the library file as well.

To create your library "package," you use the **cblutil** utility, the **COPY RESOURCE** statement, or a combination of the two. An overview of these two techniques is provided below. Refer to the *ACUCOBOL-GT Reference Manual* for additional information.

C.4.2.1 Using cblutil

The **cblutil** utility lets you embed *resources*, which are defined as pieces of static data, directly into an object file. For the purposes of the helper application, these resources can be applications, bitmaps, wave files, configuration files, and extended file descriptors (".xfd" files). The program treats the resource as if it were a disk file, but the resource is not actually a separate file in the target environment.

Using "cblutil -lib", you can specify any type of file as an input file. If an input file is a COBOL object, **cblutil** includes it in the resulting library as a COBOL object. If an input file is another library, each component of the library is individually added to the resulting library. Any other file is included as a resource.

To use the **cblutil** utility program, type "cblutil -lib" followed by the desired options, main program name, and all the modules you want included, separated by a space. Be sure to add the main or initial program to the library first, because the runtime executes the first program it encounters in the library.

Syntax:

```
cblutil -lib [options] main_program modules
```

Example:

```
cblutil -lib -v -o mylib.acu prog1.obj prog2.obj logo.bmp cblconfi data1.xfd data2.xfd
```

Note: Since the helper application identifies content with the ".acu" extension, the output library file must have the ".acu" extension. For this reason, be sure to use the "-o" option to specify the name of the output file when using "cblutil -lib". In the example above, "mylib.acu" is the specified output file.

C.4.2.2 Using COPY RESOURCE

If you are creating a simple, single object library without ".xfd" files, you can use the COPY RESOURCE statement to package your applications, bitmaps, and configuration file instead of or in addition to the **cblutil** utility.

Use the COPY RESOURCE statement as follows:

```
COPY RESOURCE resource-name [ {IN} path-name ] .
                             {OF}
```

where r*esource-name* and *path-name* identify a resource file to be included in the resulting object file.

The effect of a COPY RESOURCE statement is to add *resource-name* to a list of resources that the compiler embeds into the resulting COBOL object file. The resources are added to the end of the COBOL object in the same order as the corresponding COPY statements. Since the resources are added to the end of the object, the location of the corresponding COPY RESOURCE statement in the COBOL program is irrelevant. Conventionally, COPY RESOURCE statements are placed either in Working-Storage or at the end of the program, but any location is acceptable.

If *resource-name* resolves to a COBOL object or library file, the compiler includes this object or library in the resulting object in a manner similar to "cblutil -lib". These are not considered resources, but are embedded COBOL objects.

If you are creating a library containing multiple COBOL objects, we recommend using "cblutil -lib" instead of using COPY RESOURCE. Using **cblutil**, you do not need to worry about the order in which COBOL objects are compiled (if you use COPY RESOURCE, you must ensure that the copied object is compiled first), and **cblutil** also checks for duplicated program names while COPY RESOURCE does not.

Although technically you can include ".xfd" files in a library using COPY RESOURCE, these files must exist before you compile with the COPY RESOURCE statement. Because ".xfd" files are created at compile time, this means you must compile more than once. For this reason, the **cblutil** method is also preferred for including ".xfd" files in your object library.

C.5 Creating a Link to Your COBOL Object

To place your COBOL application on your Web page for use with a helper application or viewer, you create a hyperlink by placing the HTML Anchor tags in your HTML document. End users who visit your Web page can then run the program by clicking on the program's link.

The HTML Anchor tags, <A> and </A>, are closed elements that, when combined with the HREF attribute, highlight text or images, making them clickable. When users click on a highlighted item on your Web page, they are transferred to the linked document. Since the link in this case is a COBOL program, when users click on the highlighted item, the COBOL program is automatically invoked. (See any commercially available HTML text for more information on anchors and hypertext links.)

To turn text into a hypertext anchor, enclose the clickable text in the Anchor tags. The browser usually displays this text underlined and in a different color. For example, your HTML document may include this:

```
<A HREF="myprog.acu">Click here to run the application</A>
```

The HREF attribute is used within the starting anchor tag to specify the document to be linked (or retrieved). Then when the user clicks on the highlighted text, "Click here to run the application," on your Web page, the COBOL object "myprog.acu" is retrieved from the specified location on the Web server and run in a normal ACUCOBOL-GT window.

To use images as hypertext anchors, you place the <IMG> element within the Anchor tags. For example:

```
<A HREF="myprog.acu">
<IMG SRC="myprog.gif" ALT="Click here to run the application">
</A>
```

Then when the user clicks on the "myprog.gif" image on your Web page, the COBOL object "myprog.acu" is retrieved from the specified location on the Web server and run in a normal ACUCOBOL-GT window. While the graphic file is loading (or if the user's browser does not support images), the browser displays the alternate text, "Click here to run the application". Clicking this text invokes the application as before.

C.6 The User's Job

Once your work is done, runtime users have two tasks:

1. **Defining the Runtime as a Helper Application or Viewer** in Their Browsers
2. Visiting Your Web site and **Launching the Application**

C.6.1 Defining the Runtime as a Helper Application or Viewer

The process for defining helper applications or viewers varies according to the browser. While many browsers support helper applications or viewers, this section describes the most common: **Netscape Communicator**, **Netscape Navigator**, and **Microsoft Internet Explorer**.

To define the runtime as a helper application in Netscape Communicator

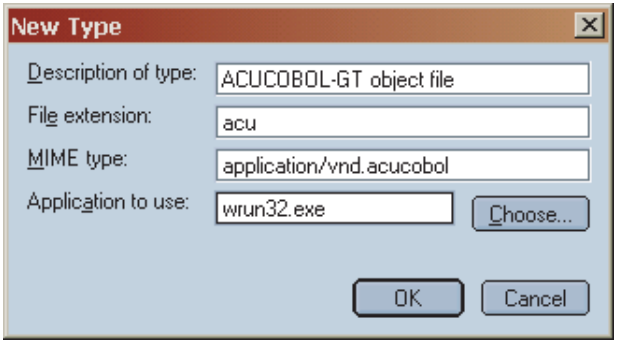
The following procedure was written for newer versions of Netscape Communicator. Your version of Netscape may operate differently. (Refer to the documentation that comes with the browser for instructions on setting up helper applications.) To configure Netscape Communicator to use ACUCOBOL-GT as a helper application, do the following:

1. Start Netscape Communicator.
2. From the Edit menu, choose **Preferences**.
3. Under the **Navigator** category, click **Helper Applications** (Netscape 7) or **Applications** (Netscape 4.5).
4. To add a helper application to the list, and click **New Type** (or **New** on UNIX).

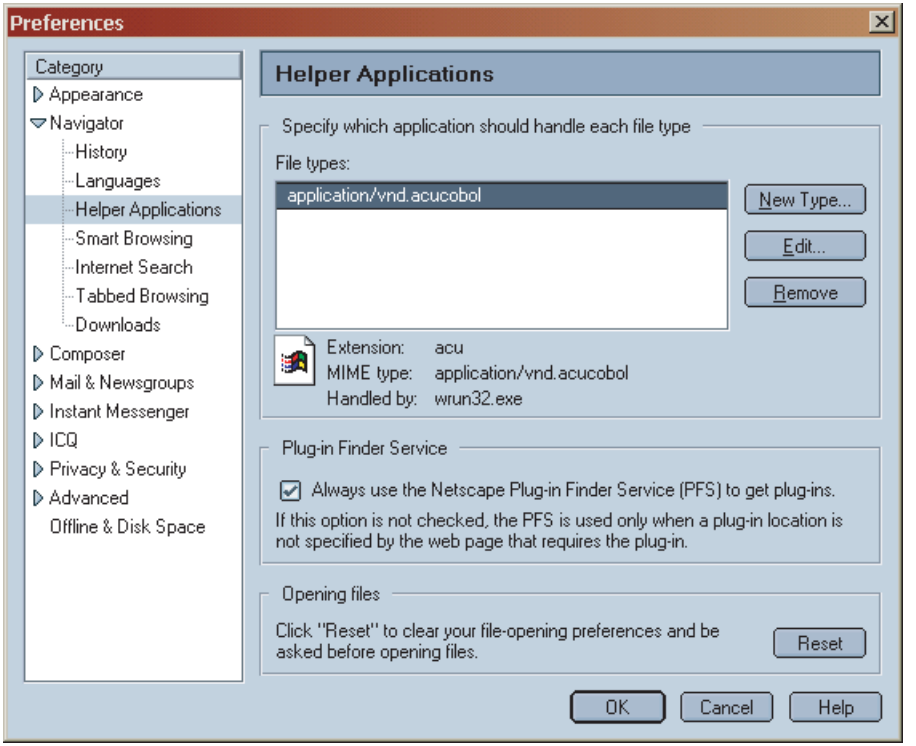
5. Click inside each field in the box and do the following:

Field	Action
Description of Type	Type “ACUCOBOL-GT object file”
File Extension	Type “acu”
MIME Type	Type “application/vnd.acucobol”
Application to use	Type “wrun32.exe” or browse to and select the ACUCOBOL-GT runtime file “wrun32.exe”. This is the command line used to launch the application. If users want to specify command line options, they may do so by typing the option after the executable name.

The New Type dialog box should now look similar to this:



6. Click **OK** to return to the Preferences dialog box, which should now look something like this:



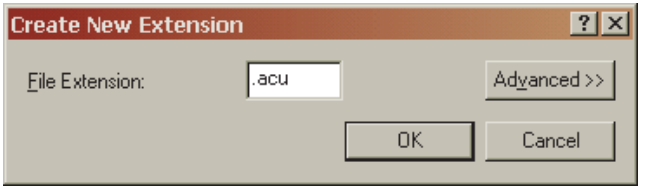
7. Click **OK** to accept the new helper application set up for the ACUCOBOL-GT runtime.

To define the runtime as a viewer in Internet Explorer

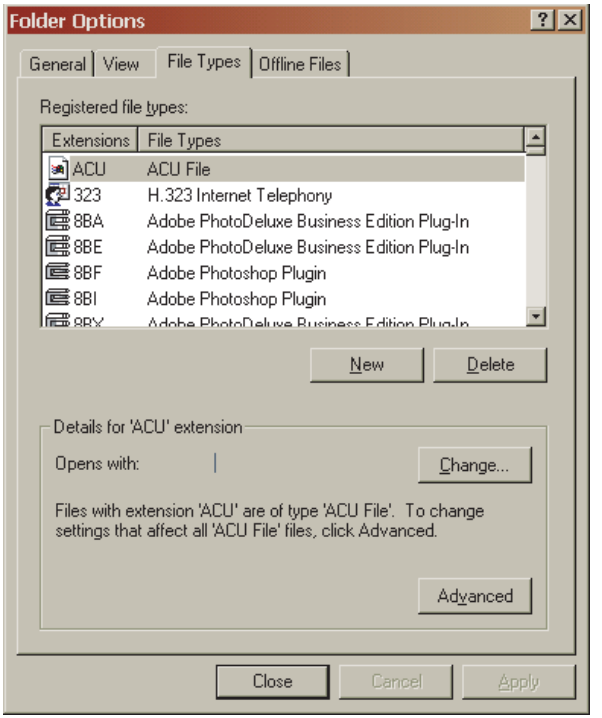
To invoke viewers, Internet Explorer uses the file type associations that users have set up for their Windows configurations. The procedure to add a file type association for ACUCOBOL-GT object files depends on your operating system.

Windows NT, 2000, ME, and XP users do the following:

1. Start Windows Explorer using the Start/Programs/Accessories menu.
2. From the Tools menu, select **Folder Options** and click the File Types tab.
3. Click **New** to open the Create New Extension dialog box.
4. Type ".acu" for extension and click **OK**.



5. On the File Types tab, click **Advanced**.

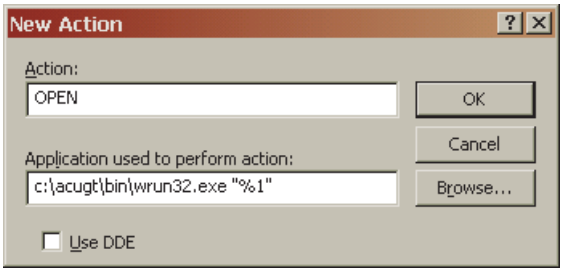


6. From the Edit File Type box, select **New...** from the Actions field.

7. Click inside each field in the box and do the following:

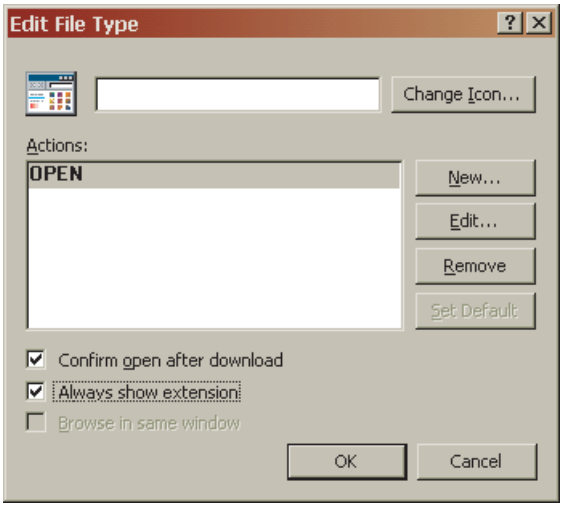
Field	**Action**
Action:	Type "OPEN"
Application used to perform action:	Browse to and select ACUCOBOL-GT runtime file "wrun32.exe". Click at the end of this field and type "%1" preceded by a space. Include the quotation marks. This indicates the complete path to the COBOL program file. **Note:** If users want to specify command line options, they may do so by typing the option after the executable name.

The New Action dialog box should look similar to this:



8. Click **OK** to accept the new action.

9. Change the icon for the application as desired. The Edit File Type box should look similar to this:



10. Click **OK** to accept the new file type.

C.6.2 Launching the Application

Now for the easy part. Once the runtime is installed on the local machine and set up as a helper application or viewer in the browser, users just start their browsers and visit your Web site. Since your application was included as a hyperlink, users simply click the highlighted text or image associated with the hyperlink to invoke your COBOL program.

The browser recognizes ACUCOBOL-GT object files, invokes the ACUCOBOL-GT runtime as a helper application, and runs your application on the end user's machine. The application runs just as if the user had invoked it directly on a local machine, using the full screen and ACUCOBOL-GT user interface. The browser continues to run in the background.

C.7 Security and the Helper Application or Viewer

It is important to realize that the ACUCOBOL-GT runtime is a fully enabled runtime, even when used as a helper application or viewer. Unlike the ACUCOBOL-GT Web Runtime, it does not contain any extra Internet security mechanisms; it has full access to your users' machines, including system calls, memory, disk and network access.

For this reason, you should warn your users not to download COBOL applications from other sources to use with this runtime, unless the applications are from a trusted source. When your users try to download a file of a new type under Netscape, they are warned that the file may contain malicious code or scripting instructions, and they are given a chance to decline the operation. Nevertheless, users should be reminded that their helper application is a full-featured runtime without access limitations.

Glossary of Terms

Browser

The user interface that one uses to "browse" the World Wide Web. The browsers discussed in this book are Netscape Navigator/Communicator and Microsoft Internet Explorer, although ACUCOBOL-GT applications can be accessed by most Web browsers.

CGI

Common Gateway Interface. A standard interface for running external programs, or gateways, on a Web server. The Internet Server Application Programming Interface (ISAPI) standard is an alternative to CGI. CGI programs can be written in any language, including COBOL.

Domain name

A domain name associates your site with a specific Internet Protocol (IP) address. Web site domain names, such as "www.company-name.com" provide a series of benefits leading to the success of your Web site.

Headers

The information included in the beginning of client requests and server responses using the HTTP model. Request headers contain file type requirements, usernames and passwords, URLs, etc. Response headers contain Web server information, date information, MIME types, etc.

Helper Application

Software installed on a user's machine that has the ability to read and process files of a given type. Synonymous with "viewer." The ACUCOBOL-GT runtime can be set up as a helper application in Netscape or a viewer in Internet Explorer.

HTML

HyperText Markup Language. A language that tells browsers how to present Web pages. HTML uses a header and tags to pass structure, formatting, hyperlink, and form description information to the browser.

HTTP

HyperText Transfer Protocol. The protocol or set of rules used by the World Wide Web to govern the transfer of documents.

Hyperlink

A hot spot in a document or Web site that when clicked, takes you to another location in the same or different document. Also called a Link.

Internet

The largest TCP/IP network in the world. A network that allows global users to exchange information instantly and seamlessly. The infrastructure behind the World Wide Web.

Internet Explorer

Microsoft's browser program. Has become one of the two browser standards used today, alongside Netscape Navigator and Netscape Communicator.

MIME

Multipurpose Internet Mail Extensions. A set of standards for encoding/decoding and identifying different types of files for transmission across multiple platforms on the Internet.

Netscape Navigator/Netscape Communicator

Netscape's browser program. Has become one of the two browser standards used today, alongside Microsoft Internet Explorer. Navigator is an older version of Communicator.

SSI

Server-Side Includes. Commands in an HTML document that are interpreted by the Web server. SSIs are used when you want to include the contents of another file in the current HTML document or to execute a script whose output will be included in the current HTML document before being sent to the browser client.

Tags

Codes that you use to mark up HTML documents in order to format the document's appearance. Tags indicate to the Web browser how to display and handle portions of the document.

TCP/IP

Transmission control protocol/Internet protocol. The protocol or set of rules used by the Internet and most client/server networks to facilitate communication between computers.

Templates

HTML documents that can be used to provide standard formatting for other HTML documents. Templates save hours of work. They can be used to store common HTML shared by multiple Web pages on a single Web server. This decreases the amount of replication and simplifies the task of maintaining a Web site.

Thin client

A configuration in which your application is composed of two logical layers: a user interface (UI) layer on the display host (client) and a COBOL layer on the application host (server). The UI layer handles screen, mouse, and keyboard activity, and the COBOL layer performs application processing. With the ACUCOBOL-GT Thin Client, you do not have to split your application into these layers, because a special runtime does it for you. Because no application components are required on the client (unless you want to use ActiveX controls), it is considered to be "thin."

URL

Uniform resource locator. It contains the address or location of a resource on the Internet. The resource may be located on a server or locally on the client machine. The resource can be a file, command, or query that is handled using a protocol or access method indicated in the URL prefix. Some common URL prefixes are http, ftp, file, news, javascript, telnet, wais, and gopher.

Viewer

Software installed on a user's machine that has the ability to read and process files of a given type. Synonymous with "helper application." The ACUCOBOL-GT runtime can be set up as a helper application in Netscape or a viewer in Internet Explorer.

Web site

A Web page or a set of related Web pages on the World Wide Web.

Web thin client

The ACUCOBOL-GT Web Thin Client is a special 32-bit version of the thin client that is based on Microsoft's ActiveX technology. It is itself an ActiveX control that you can embed on your Web page. It makes your existing thin client applications accessible through browsers that support ActiveX, particularly Microsoft Internet Explorer.

WAP

Wireless Application Protocol. The communications protocol commonly used for wireless devices like mobile phones and Personal Digital Assistants (PDAs).

WML

Wireless Markup Language. Language standard of Wireless Application Protocol (WAP).

World Wide Web

Also known as WWW. A collection of sites or pages that provide access to text, graphics, multimedia, and more over the Internet.

XML

eXtensible Markup Language. Language standard for business-to-business data exchange, Web services, and dynamic content Web publishing.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X