User's Guide

AcuConnect®

Version 8.1.3

Micro Focus
9920 Pacific Heights Blvd.
San Diego, CA 92121
858.790.1900

E-01-UG-100501-AcuConnect-8.1.3

Contents

Chapter 5: Preparing the Client(s) in Distributed Processing

Chapter 6: Preparing the Client(s) in Thin Client

1 Introduction

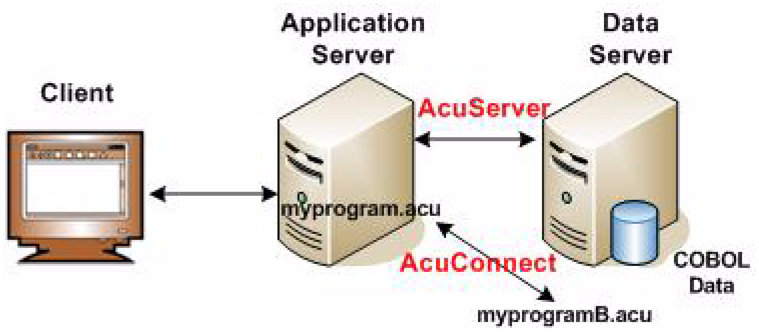
Key Topics

1.1 Overview

Welcome to AcuConnect®! Part of the ***extend***® family of solutions, AcuConnect is a client/server technology that can help you distribute your computing processes in the way that best suits your business needs, while optimizing system performance and your existing resources. AcuConnect lets you implement a client/server system in which the client component can be as "thin" or as "thick" as you need.

AcuConnect can be deployed in two ways: in a distributed processing environment or in an environment that uses our thin client technology.

With AcuConnect's distributed processing deployment, users can distribute application logic among client and server machines in a way that best suits their needs. AcuConnect users who take advantage of our thin client technology can run the user interface (UI) portion of an application on a Windows graphical display host while the rest of the application and data reside on a server.



This chapter introduces you to the basic concepts of these two AcuConnect environments.

1.1.1 Distributed Processing with AcuConnect

With AcuConnect in a distributed processing environment, 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. While some portions of your application continue to run on the client (for example, the user interface and some interactive programs), the resource-intensive portion runs where it is most efficient . . . on the server.

Should users decide to run their programs on server1 one day and server2 the next, they can do so without any changes to the application code. They can even run the application locally if they choose, or “daisy chain” applications together on multiple servers.

With server programs written accordingly, AcuConnect can also provide users access to Vision data. If the data is on the same remote server as AcuConnect, no special software is required to access the data from a local client. If the data is on a separate data server, AcuConnect works in tandem with our remote file server, AcuServer®, to provide seamless data access. AcuConnect can also work with Acu4GL®, our COBOL-to-RDBMS bridge, to provide seamless access to relational data wherever it resides.

By enabling you to distribute processing between the client and server, AcuConnect helps you leverage your computing resources to the fullest in a true distributed processing environment. For example, with AcuConnect, you can take advantage of the power of your server machines while using inexpensive client machines. AcuConnect

- Improves application performance by offloading major processing tasks to the server.
- Decreases network traffic, especially for I/O-intensive applications.
- Lengthens the life of older computing equipment; users of slower, older technology can use newer server components for the majority of their processing.
- Lets you take advantage of existing resources, including the Internet.

You can also use AcuConnect to execute remote COBOL objects from client applications developed in C, C++, Java, .NET, Delphi, or Visual Basic. These applications run on the client and use the C, .NET, or Java API contained in our COBOL Virtual Machine (CVM) to interact with the COBOL object on the server. As instructed by a configuration file on the client, AcuConnect executes the COBOL object remotely and shares data with the client application through the CVM on the client. For information on how to execute remote COBOL objects, refer to section 5.4.1, “Executing Non-COBOL Programs on the Client,” in this manual.

1.1.2 Thin Client Technology with AcuConnect

Our thin client technology enables you to display your ACUCOBOL-GT® graphical server-based application on graphical display hosts.

The thin client technology is designed for two main purposes:

1. To allow ACUCOBOL-GT programs running on a UNIX, Linux, MPE, or VMS server to present a full Windows graphical user interface (GUI) on PCs networked with TCP/IP. The application screens may require conversion to graphical, but your application stays in one piece on the server.
2. To allow you to host your application on a UNIX, Linux, MPE, VMS, or Windows server and enjoy the benefits of centralized application maintenance and to adopt the performance characteristics of a "thin" architecture. Many applications perform better when deployed in a thin fashion compared to other networking techniques, such as remote file access ("thick clients") or distributed processing. This is because thin client configurations execute COBOL programs on the server, where data access is local.

The benefits of our thin client technology are plentiful:

- UNIX users can enjoy the best of two worlds: the stability and security of the UNIX world in multi-user environments and the graphical nature of Microsoft Windows. Previously, UNIX users could use terminal emulation, move their applications to Windows and add a GUI or split their applications into two or more pieces.
- UNIX applications can use ActiveX controls, Automation Servers (OLE objects), and Windows Help systems on Windows clients. UNIX users can also interact with .NET assemblies in thin client. An ACUCOBOL-GT screen can include a mix of Win32, ActiveX, and .NET graphical controls, if desired.
- Users can print locally on their Windows machine or on the remote server. They can also choose whether to perform local or remote debugging.
- You don't have to separate your application into multiple tiers or deploy any of your application logic on client machines.

- You can usually improve application performance by executing COBOL programs on the server, where data access is local. Server processing eliminates the need to perform file operations over the network, which can be quite slow when many records are involved. This reduces network traffic and often improves response time. (The thin client model does perform screen I/O over the network, but usually at a lesser cost than the file I/O that it replaces.)
- Administration is simplified when an application resides solely on a server. Installation is easier, as are upgrades, distribution, and management.
- Security is enhanced when an application resides centrally.
- The total cost of ownership for enterprise information systems is reduced. With server-side processing, you remove many client requirements, providing end users easier, more cost-effective access to information.
- Users and enterprises enjoy increased flexibility with a choice of clients.
- The AcuConnect Thin Client solution can run on any TCP/IP network, including the Internet.

Our thin client solution can work in conjunction with other technologies from the ***extend*** family of solutions. If data will be stored on a separate data server in a three-tier architecture, your thin client solution can use AcuServer to provide transparent access to that data. If your application requires access to relational databases like Oracle, Informix, and Sybase, your thin client solution can use Acu4GL or AcuSQL® technologies on the server to provide automatic data translation. You can migrate your character-based application to ACUCOBOL-GT, display it on Windows hosts, and when you're ready, convert your screens to a full-featured graphical user interface. You can develop your GUI in Windows using our integrated development environment, AcuBench®.

1.2 Platform Support

Platform support for AcuConnect's distributed processing deployment is different from that in the thin client deployment. The following sections outline the specific platforms supported in each environment.

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.).

1.2.1 Distributed Processing Platform Support

AcuConnect’s distributed processing solution supports the following clients:

- UNIX
- Windows XP
- Windows Vista
- Windows 7

In this environment, AcuConnect supports the following servers:

- UNIX
- Linux
- Windows 2000, 2003, and 2008 R2

1.2.2 Thin Client Platform Support

The thin client technology supports the following client platforms:

- Windows XP
- Windows Vista
- Windows 7

The Thin Client technology supports the following server platforms:

- UNIX
- Linux
- VMS version 7.2 or later

- HP e3000 MPE
- Windows 2000, 2003, 2008 R2

1.3 How AcuConnect Works

This section describes in general terms how AcuConnect works in its various environments. In all cases, AcuConnect runs as an independent resident program (daemon) called **acurcl**.

1.3.1 Distributed Processing

In a distributed processing environment, AcuConnect is a runtime server that handles user requests to start new runtimes on server machines. AcuConnect supports multiple runtime instances so that users may start multiple applications on the server at the same time. It also allows programs on one server to start programs on another server, *ad infinitum*, as long as each server has a copy of AcuConnect installed.

In distributed processing, both the client and the server have a copy of the ACUCOBOL-GT runtime. The runtime version on the client should match the version of the runtime on the server. If your application interfaces with C-ISAM or other special library routines, you must relink the runtime with the library routines you are using.

Because both the client and the server have an ACUCOBOL-GT runtime, you decide how much of an application runs on the client and how much on the server. The client can be as “thin” or as “thick” as you like. With AcuConnect in distributed processing, the client is actually a “smart” client.

To launch a server program, the client uses standard COBOL CALL syntax. You embed a CALL in the client application, and AcuConnect launches the server application for you automatically. All application (and data) access is completely transparent to the end user.

Differentiation is achieved through configuration files and, more precisely, through definition of either a CODE_PREFIX variable or a “code name alias,” which defines the directory containing the object programs. Without modifying or recompiling the original code, the same object can operate on any server, client, or stand-alone machine.

The AcuConnect process can be described as follows:

1. A network administrator starts AcuConnect on the server. On Windows servers, this can be accomplished through an application service. On UNIX, the administrator uses the **acurcl** command along with options. For instance, he or she may type:

   ```
   acurcl -start -c acurcl.cfg -e server.err
   ```

2. The user starts the ACUCOBOL-GT client application on the client. On Windows, the user clicks an icon. On UNIX, the user types a **runcbl** command. For instance:

   ```
   runcbl -c client.cfg prog1.acu
   ```

3. The client application, “prog1.acu” in this example, performs a remote CALL to the server application. For instance:

   ```
   CALL "prog2.acu" using customer-info.
   ```

4. AcuConnect automatically starts the remote application, “prog2.acu”, on the server using the runtime flags and configuration file specified in “client.cfg”.
5. The remote application performs the requested task and returns a response to the client. If programmed accordingly, the remote application may perform a CALL to another application on the same or different server before sending back a response.
6. The client program processes the result, possibly displaying it to the user.

Synchronous or Asynchronous Operation

In the example shown above, the client application waits in a suspended state until the remote application performs its task and returns a result (that is, a “synchronous” CALL is performed). If desired, you can allow the client application to continue running by CALLing the C$ASYNCRUN library routine along with the remote application. In this case, you specify the "handle" of the remote application, as well as the application name itself. For instance:

```
CALL "C$ASYNCRUN" using handle-of-prog2 "prog2.acu" customer-info.
```

C$ASYNCRUN tells AcuConnect to allow asynchronous processing. If you use C$ASYNCRUN, you can check the status of the server application using the companion routine, C$ASYNCPOLL. See section 4.3.2, "Synchronous or Asynchronous Execution," for more details.

1.3.2 Thin Client

Our thin client computing is a network-based approach to information processing. With the thin client technologies, your software can display on Windows front-end clients, access COBOL programs in practically any server operating environment, permit local or remote printing, and allow remote debugging from Windows.

Different from AcuConnect's distributed processing environment, in which some application components are processed on the client and some on the server (and the application is engineered accordingly), thin client configurations perform all processing on the server and pass simple screen commands between the client and the server. Because server hardware is usually faster, and because data access on the server is local, application performance often improves when processing is performed on the server.

In a thin client configuration, your application comprises two logical layers: a 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 want to use ActiveX controls), it is considered to be truly "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 runtime portion that is installed on the thin client is known as the ACUCOBOL-GT Thin Client. The thin client can be installed from your product media or downloaded from the support section of Micro Focus's Web site (www.microfocus.com). Note that you must be a registered user to access the download portion of this site. The full ACUCOBOL-GT runtime is installed on the server.

To function, the split runtime makes use of AcuConnect as a "remote COBOL listener" on the server. The role of the listener is to wait for requests from clients to launch the ACUCOBOL-GT runtime on the server. When a request is received, the AcuConnect listener starts a new runtime on the server, connects that runtime with the client process, and removes itself from the communication stream to wait for requests from other enabled applications. Once initiated, the runtime on the server executes the COBOL application and sends screen commands back to the client.

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

1.4 Licensing

The licensing scheme for AcuConnect allows AcuConnect to track both types of client connection—distributed processing and thin client. The Activator license utility creates "acurcl.alc" when generating licenses for thin client connections and "acurcl.clc" when generating licenses for distributed processing connections. See the *Getting Started Guide*, *Section 2.1 License Files for Windows* for details on the Activator license utility.

AcuConnect is shipped with a product code and product key that defines the licensing capabilities specified in your purchase agreement. When you supply your key information during installation, the Activator creates a license file that contains information such as the product's version number, serial number, expiration date, and server count. After installation, AcuConnect must be able to locate the license file to function.

AcuConnect searches for "acurcl.alc" for thin client connections and "acurcl.clc" for distributed processing connections in the same directory as the **acurcl** executable. If you move the AcuConnect executable to a new directory, be sure to move a copy of the license file as well. If no license file is found, or if the information in the license file does not permit execution, AcuConnect exits with an error message.

When AcuConnect detects a client request to start a remote application runtime, it checks the connection count to see if another connection is permitted. If another connection is allowed, AcuConnect attempts to start a runtime on the server and connect it with the client process. The runtime, in turn, checks its usage count before it begins execution of the application. When an AcuConnect user disconnects from the remote server by exiting the COBOL program, the AcuConnect and runtime usage counts are decremented and those slots are made available to subsequent users. If the usage limit has been reached, the server runtime is not started and an appropriate message is returned to the end user.

The ACUCOBOL-GT Thin Client, which is installed on client machines, is available on your product media or as a download on Micro Focus's SupportLine Web site. You are required to accept a clickwrap license agreement on installation.

The **acushare** utility is required for all UNIX installations that do not have an unlimited license. The **acushare** utility monitors and enforces the site runtime usage limits.

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 like Microsoft may be invoked. Carefully review any license agreements with these third-party vendors before proceeding with the remote connection.

Nothing in this document 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 Micro Focus's licensing technology, which is required to operate the ***extend*** software. 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.

1.5 Security

AcuConnect system security is designed to address two fundamental issues:

1. Controlling access to data files

2. Preventing unauthorized use of client components to perform privileged activities (such as modifying privileged files)

The first issue, controlling access to data files, is addressed in two ways: first, via a server access file known as AcuAccess (the same access file used by AcuServer), and second, through the standard UNIX or Windows server file access provisions. Whether an AcuConnect user can access to a given file on the server depends on two things: (1) the user ID assigned the requester in the server access file, and (2) either the Windows security set up for your files, or the UNIX ownerships and permissions set on the particular file.

The second issue, preventing unauthorized privileged use, is addressed through strict enforcement of the security measures that you have established through the server's operating system.

Achieving sound system security depends on the configuration and management of the following security elements:

- The AcuAccess server access file (the database of authorized AcuConnect users)
- The Windows server security protections set up for the **acurcl** executable file, server configuration file, server access file, and remote data files and directories. Microsoft recommends that you use the NT file system (NTFS) for better security.
- The UNIX ownerships assigned to the **acurcl** executable file, server configuration file, server access file, and remote data files and directories
- The UNIX access permissions (read, write, and execute) set on the **acurcl** executable file, server configuration file, server access file, and remote data files and directories

When AcuConnect is running as a Windows service (NT/2000/2003/2008), it belongs to an implicit group called "SYSTEM." Make sure that the "SYSTEM" group is added to your file permissions with "Full Control." (This is not necessary if you are using Windows NT security via the SECURITY_METHOD configuration variable.)

UNIX ownerships and permissions can be set on key AcuConnect files. Note, however, that your site could jeopardize security if you include entries in the server access file that explicitly allow users running as root on the clients to run as root on the server. We strongly discourage the inclusion of such entries.

UNIX ownerships and permissions on the **acurcl** executable, server configuration file, and server access file are described in section 2.3, "Establishing System Security." These specifications must be strictly maintained. If the ownerships and permissions are more permissive than those specified, AcuConnect will not start, halting system operations.

In addition to the AcuAccess file, AcuConnect offers internal socket layer encryption to further enhance security. Encryption protects information while it is in transit across the network. For information about the configuration variables used to enable encryption, refer to section 3.4, "Creating a Runtime Configuration File for the Remote Server Component."

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 The AcuConnect Server

Key Topics

2.1 Getting Started with AcuConnect

This chapter describes some of the steps you need to take before you start using AcuConnect®. AcuConnect's installation and start-up instructions are included, along with detailed information about establishing system security. The latter topic includes the creation of the server access file and server alias file for thin client. Ownerships and permissions are also discussed. Configuration file entries are covered in Chapter 3.

2.2 Installing AcuConnect

To prepare your server, you must first install AcuConnect. You also need to install an ACUCOBOL-GT® runtime. The procedure for installing AcuConnect varies, depending on whether you are installing on a UNIX server or a Windows server. The following sections describe the process for these systems.

For thin client architectures, you can also install AcuConnect on a VMS or MPE/iX server. Some special considerations apply when you want to run AcuConnect on MPE/iX systems. Refer to section 2.2.3, "Installing AcuConnect in Thin Client on a VMS Server," for VMS procedures, and to section 2.2.4, "Running AcuConnect in Thin Client on HP MPE/iX Systems," for MPE/iX instructions.

For AIX version 5.1 and later, HP-UX version 11.11 and later, and Solaris version 7.0 and later, ACUCOBOL-GT products are distributed as shared object libraries. If you install AcuConnect as a shared object library, and you choose not to install to the default location (/opt/acucorp/8xx), you need to set an appropriate library search path variable specifying the location of the shared objects. The exact library path variable to set depends on the individual operating system. For example, on an AIX system, you would need to set the LIBPATH environment variable. Note that if you log in as root or superuser, this variable must also be set in root's environment for AcuConnect to start.

2.2.1 Installing AcuConnect on a UNIX Server

For UNIX platforms, AcuConnect software is shipped on the product CD-ROM. Use the following procedures to install AcuConnect on a UNIX server:

1. Create a directory on the server to hold the AcuConnect software. We recommend that you install AcuConnect directly into the ACUCOBOL-GT home directory.

2. Mount the product CD-ROM on your UNIX system and change directory ("cd") into the directory in which your product CD-ROM is mounted and execute the install script as described on the *Quick Start* card that came with your product.

   Follow the instructions on the card, entering your product code and product key when prompted.

   If you want to use the server configuration file in its default location, move "acurcl.cfg" into the /etc directory. This may require root or superuser privileges. The "acurcl.cfg" file may remain in the current directory or be copied or moved to some other directory. If "acurcl.cfg" is not located in the default directory (/etc), the name and location of the configuration file must be specified with the "-c" option each time AcuConnect is started. See section 8.2.1.9, "acurcl -start," for more information.

3. On UNIX servers, you must create a server access file (default name /etc/AcuAccess) before AcuConnect will start or establish connections with clients. AcuConnect includes a utility program for creating and maintaining the server access file. To create a minimal server access file that allows any user of any client to use AcuConnect, perform the following steps:

   a. Log on to the server as root or become superuser.

   b. Use the "**acurcl** -access" command to launch the server access file manager utility.

   c. When prompted for the path and name of the server access file, accept the default /etc/AcuAccess.

   d. Respond **yes** to the "Do you want to create it now?" prompt.

   e. From the main menu, select option [1], "Add a security record."

The manager presents a series of five prompts. Accept the default for each prompt by pressing **Return**. After you accept the umask field, the program reports that it has added the record.

f. Exit the utility by selecting option [5] from the main menu.

Please note that for the minimal server access file, the user should have an account on the server by the same name as the account on the client. For more details about the AcuAccess file and important information about security, refer to section 2.3.3, "The Server Access File."

AcuConnect is now fully installed and ready for configuration on the UNIX server. See section 3.3, "Creating a Server Configuration File," for the settings that can be configured for the server.

Note that **acushare** is installed automatically as part of AcuConnect installation.

2.2.2 Installing AcuConnect on a Windows Server

For Windows networks, AcuConnect software is also on the CD-ROM. Use the following procedures to install AcuConnect on a Windows server. Note that if you move or delete any installed ".dll" files, AcuConnect will not run.

1. Install and configure TCP/IP before installing AcuConnect.
2. Log on to your NT server using the Administrator account or an account that belongs to the Administrators group.
3. Insert the product CD-ROM into your disk drive. If the installation program does not start automatically, follow the steps below:

   a. Click the **Start** button, select **Run**, and enter the following:

   ```
   D:\setup.exe
   ```

   replacing *D* with the device designation of your CD-ROM drive.

   b. Follow the instructions on the screen, entering your product code and product key when prompted.

c. During the installation procedure, you will be prompted to specify an installation directory or to accept the default location. You will also be prompted to select a component to install. Select **AcuConnect** from the list of available products.

4. If you already have the files “c:\etc\AcuAccess” and “c:\etc\acurcl.cfg,” the setup utility detects them and asks if you want to overwrite them. **Do *not* overwrite them unless you have a backup copy.** The AcuAccess file contains one access record that gives all users access to AcuConnect. You can modify this file later, if desired. The file “acurcl.cfg” contains the server configuration variables; when the file is first installed, these variables are all commented out. You can also modify this file later if desired.

5. If you want AcuConnect to start automatically on boot, it must be installed and started as a Windows service. The setup program creates a sample Start Menu folder, if desired, and asks if you want to install and start these services.

   You can use the graphical control panel to install and start AcuConnect as a service. The Services tab in this interface lets you add, start, stop, and remove services.

   Please note that installing a service on a particular port resets all startup options for the service on that port. You can use all valid “-start” options when installing AcuConnect as a service. These options are stored so that the service will use them when starting.

   The default service is named “AcuConnect.” All other service names include the port on which to run.

   Note that service naming differences prevent AcuConnect from administering services created by pre-Version 6.2 AcuConnect, and vice versa.

   If you choose to start AcuConnect during the installation process, a Windows Command Prompt (DOS window) appears, showing the status of the Windows services being started or restarted automatically. You may see some error messages that can be ignored if the Windows services are not already installed and running. For example, you might see:

   ```
   acurcl –kill
   Open/Control Service failed
   ```

```
acurcl –remove
Open/Control Service failed
acurcl –install
AcuConnect service installed.
acurcl –start
STATE: START PENDING
```

6. The Windows services that are needed for AcuConnect can also be started manually. You can start the required Windows services from the command line ("**acurcl** -install", "**acurcl** -start") or from the control panel.

   In Windows, issuing the **acurcl** command without options causes a graphical control panel to appear. You can start AcuConnect from the Services tab in this interface.

   Refer to Chapter 8 for information about other **acurcl** command-line options.

   Please note that when AcuConnect is started as a Windows service, all paths used in the configuration file or on the command line are relative to the Windows server system directory (for example, c:\winnt\system32). For instance, if your current directory is c:\acucorp\acucbl8xx\AcuGT\bin and you start AcuConnect with the command "**acurcl** -start -le acurcl.log", the log file is not created in the current directory, but rather in the \winnt\system32 directory. If desired, you can use full pathnames, which has the effect of using an explicit file.

7. After the services are installed and running, there should be no reason to stop them. However, if you choose to stop them, you may do so from the command line ("**acurcl** -kill"). You can also stop services by clicking **Stop** in the Services tab on the graphical control panel.

   You may perform start and stop operations from the Service Control Applet. After the services are installed, there should be no reason to delete them. However, if you decide to delete them, use the following command from the command line:

   ```
   acurcl -remove
   ```

   or click **Remove** on the graphical control panel's Services tab. Refer to Chapter 8 for information about other **acurcl** command-line options.

This command deletes only the services and does not delete the executables. To help in resolving service problems, you can check some messages from these services in the Microsoft Event Viewer:

Start/Programs/Administrative Tools/Event Viewer/Log/Application

Note: This location may vary, depending on your version of Microsoft Windows.

2.2.3 Installing AcuConnect in Thin Client on a VMS Server

For VMS platforms (version 7.2 or later), AcuConnect software is shipped in BACKUP format. Select a directory on the server to hold the AcuConnect software, and extract the files from the tape. Follow the installation instructions on the *Quick Start* card that came with your product. You should be logged in to the “SYSTEM” account or an account with “SYSPRV” system privileges.

Please note that runtimes on VMS servers use RMS indexed files rather than Vision.

Install AcuConnect on VMS as follows:

1. Define a symbol so users can access AcuConnect. Set up a symbol for each AcuConnect user.

   ```
   acurcl == "$disk:[directory]acurcl.exe"
   ```

2. Use the “**acurcl** -access” command that launches the access file utility to create a server access file.

   Refer to section 2.3.3, “The Server Access File,” for information about creating and maintaining the server access file. Note that the server access file in VMS is an RMS indexed file rather than Vision.

   ```
   acurcl -access
   Enter the name of the Server Access File
   Filename [acuaccess.dat]: acuaccess.dat
   ```

3. Set up user account quotas to define each user's capabilities. You can always modify account quotas as needed, but we suggest the following settings if you encounter quota errors:

```
Maxjobs:       0    Fillm:       300   Bytlm:       100000
Maxacctjobs:   0    Shrfillm:      0   Pbytlm:           0
Maxdetach:     0    BIOlm:     10000   JTquota:       4096
Prclm:        40    DIOlm:     10000   WSdef:       100000
Prio:          4    ASTlm:       400   WSquo:       100000
Queprio:       0    TQElm:       400   WSextent:    100000
CPU:      (none)    Enqlm:      4000   Pgflquo:     100000
```

AcuConnect is now installed and ready for configuration on the VMS server. Refer to section 3.3, "Creating a Server Configuration File," for possible configuration variable settings. Refer to section 2.4, "Creating a Server Alias File in Thin Client," to learn how to create and maintain an alias file that holds all the information needed to start the desired application on the server. Section 2.6.4, "Starting AcuConnect in Thin Client on a VMS Server," describes how to start AcuConnect in this environment.

2.2.4 Running AcuConnect in Thin Client on HP MPE/iX Systems

A limitation in the MPE/iX operating environment requires that sites planning to use AcuConnect on an MPE/iX host carefully consider how they will deploy the service. The remainder of this section describes this situation and offers two management approaches for deploying on MPE/iX.

AcuConnect uses the same basic methods as AcuServer® to start the service, validate a service request, and fulfill a request. AcuConnect runs as an independent resident program (daemon) called **acurcl**. The running **acurcl** process takes the user ID of the account that starts the program. The service normally runs in the background, waiting for requests. When a service request is received, **acurcl** determines the user ID of the requester, checks an authorization file (AcuAccess) to determine if the requester is allowed to use the service, and determines the proxy user ID to be used for the requester (the requester user ID can be mapped to another ID; the mapped ID may be unique or may be shared by a number of users). If authorization is successful, **acurcl** temporarily assumes the proxy user ID to fulfill the request. On most UNIX systems, the SETUID system function is used to assume the correct user ID. A similar function is used in the Windows operating environment.

In the MPE/iX environment, the operating system doesn't provide a way for a program (in this case, **acurcl**) to change its user ID. Therefore, the service always uses the ID of the account that started **acurcl**. Any action **acurcl** takes is performed with that ID. This inability to change IDs imposes some limitations and requires that MPE/iX sites carefully consider how they will deploy AcuConnect.

Because **acurcl** takes the user ID of the account that starts it, and because it uses that ID to access files and fulfill requests, that account must be able to service all anticipated requesters. There are two approaches to managing this issue; the approaches can be combined.

One approach is to start **acurcl** from an account that is accessible to all requesters (a "group" account). Of course, such an account must have all of the necessary access permissions to satisfy every requester. The limitation of this approach is that all requesters have the same proxy user ID on the server, and there is no way to identify a unique requester.

The second approach is to start a separate instance of **acurcl** for each unique requester, or group of requesters (multiple group accounts). This approach will work as long as the number of separate instances doesn't over-tax system resources (process space, processor capacity, and dynamic memory). The number of instances that each system can handle will vary depending on the resources of that machine. Some experimentation may be necessary to determine the limits of a given machine. Note that when **acurcl** is not executing a request, it waits on a socket in an efficient loop, consuming few resources.

2.2.5 Installing the ACUCOBOL-GT Runtime

The process for installing the ACUCOBOL-GT runtime on your UNIX or Windows server is basically the same as the process for installing AcuConnect.

On UNIX, follow the instructions in section 2.2.1, "Installing AcuConnect on a UNIX Server," taking care to insert the ACUCOBOL-GT runtime installation media and to select a target directory for the installation.

On Windows, follow the instructions for installing AcuConnect from the product CD-ROM, but make sure to select “ACUCOBOL-GT runtime” from the Product Selection screen.

2.2.6 Relinking the Server Runtime in Distributed Processing

If your applications don’t interface with any special library routines, you don’t need to relink your runtime. If, on the other hand, your applications interface with C-ISAM or other special library routines, you must relink the included runtime with these special routines. To execute the link, you must have the appropriate C compiler for your host machine:

- On Windows servers, use Microsoft Visual Studio .NET 2005. Refer to Chapter 5 in *A Guide to Interoperating with ACUCOBOL-GT* for more information.
- On UNIX servers, use the C compiler supplied by the vendor of your UNIX system. On some machines, this is supplied with UNIX; on others, it is an add-on option.

For C-ISAM, follow the relinking procedure described in the add-on module’s documentation.

For other special library routines, follow the procedures in Chapter 6 in *A Guide to Interoperating with ACUCOBOL-GT.*

Note: Before relinking, modify “direct.c” (*direct method*) or “sub.c” and “sub85.c” (*interface method*) to include your C routines (see Chapter 6 in *A Guide to Interoperating with ACUCOBOL-GT*).

2.3 Establishing System Security

Proper ownerships and permissions on the **acurcl** executable file, server configuration files, server access files, and existing data files and directories are essential to establishing a secure and functional system. Whether an AcuConnect user can access a given application depends on two things: (1)

the user ID assigned the requester in the server access file (discussed in detail in section 2.3.3, "The Server Access File"), and (2) either the Windows or UNIX native server security set up for your files, or the UNIX ownerships and permissions set on the particular file. UNIX ownerships and permissions on the **acurcl** executable, server configuration file, and server access file must be strictly maintained. If the ownerships and permissions are more permissive than those specified, AcuConnect does not start.

2.3.1 Windows permissions

After you set up your applications, you may set access permissions by using the Windows server security features. Please refer to your Windows documentation for more information about security procedures. Note that in Windows, the AcuAccess and "AcuAccess.vix" files should be readable and writable by "Administrator" and "System," with no other access. Make sure that the AcuAccess file and the "acurcl.cfg" file can be written only by those accounts and groups that you want to have write privileges.

Note: We recommend that you use native system security rather than AcuConnect system security. On Windows 2008 it is essentially required that you use system security. To use native security, you set the SECURITY_METHOD variable in *both* the runtime configuration file on the client and server configuration file on the server. You still create a server access file containing access records that define your user base, but the server access file is used only to check if the user connecting to the server is allowed to connect, and to check to which local account the connection should be mapped.

We recommend that you install and run AcuConnect on an NTFS drive, because FAT partitions offer no security to files or programs and are not supported by Micro Focus. If you install AcuConnect on an NTFS partition, be aware that the user connecting to AcuConnect needs all of the following:

- READ (RX) permissions on "wrun32.exe" and "acurcl.clc"
- READ on "wrun32.alc" (or other license files)
- The appropriate permissions to access any file

For example, if ACUCONNECT_RUNTIME_FLAGS contains “-e logfile”, AcuConnect attempts to write “logfile” in the same directory as “acurcl.exe”. In this case, the user would need CHANGE (RWXD) permissions to access that directory.

If the user connecting to AcuConnect is mapped to DEFAULT_USER, then DEFAULT_USER needs these permissions.

2.3.2 UNIX ownerships and permissions

Setting ownerships and permissions requires root privileges on UNIX systems or Administrator privileges on Windows systems. On UNIX, use the commands **chown**, **chgrp**, and **chmod** to set ownerships and permissions.

UNIX permission modes are specified by a series of three octal numbers. These three numbers assign access privileges to *user*, *group*, and *other,* respectively.

In each *user*, *group*, and *other* field, the following values provide the following privileges:

Value	**Permissions**
7	read, write, and execute
6	read and write
5	read and execute
4	read only

The UNIX command “ls -l” will return the permissions, ownerships, file size, and modification date of a file or directory. For example:

```
ls  -l  /usr2/bsmith/fio_seq
```

returns:

```
-rw-r--r--  1 bsmith  general  4870 Jan 18 2005
/usr2/bsmith/fio_seq
```

For details regarding UNIX file permission masks and the use of **chown**, **chgrp**, and **chmod**, see your UNIX operating system manuals.

Key file settings

UNIX ownerships and permissions must be assigned to key AcuConnect files as specified in the following table:

File name	**Owner**	**Permissions**
acurcl (executable file)	root	755
AcuAccess (server access file)	root	644
acurcl.cfg (server configuration file)	root	644

The permissions specified in the above table are the *least* restrictive (most permissive) settings allowed for each file. The specified permissions are the optimal permissions for most installations. However, more restrictive permissions may be assigned. (Note that more restrictive permissions could prevent some users from using some AcuConnect functions. For example, if the **acurcl** executable file were assigned permissions of 700, no user other than root could execute the "**acurcl** -info" command to generate a report of current AcuConnect system status.)

If the files named in the preceding table do not possess the specified ownerships and permissions (or more restrictive permissions), AcuConnect does not start.

You must also set appropriate ownerships and permissions on existing data files and directories. Appropriate ownerships and permissions are those that allow file access to the individuals and groups that require access and that disallow access to all others. See your UNIX operating system documentation for a discussion of file permissions and file security.

Caution: Your site could jeopardize security if you include entries in the server access file that explicitly allow users running as root on the clients to run as root on the server. We strongly discourage the inclusion of such entries.

Runtimes that are started by AcuConnect on the server inherit the environment of the user who started AcuConnect. Therefore, we recommend that you log on as the DEFAULT_USER and then use the "su" command to gain root privileges and start up AcuConnect. This ensures that any users mapped to the DEFAULT_USER account do not have any more privileges than you intend.

Ownerships and permissions on new files or processes

When a client application makes its initial request to AcuConnect for services, the requester is validated for permission to use AcuConnect. If the requester is permitted to use AcuConnect, the runtime invoked for that user runs in the context of that user. The runtime acts exactly as it would if the user logged onto the server machine and started the runtime from a shell. All access permissions and restrictions for the user apply to the runtime running for the user.

umask

The umask of the runtime process started for the user is the umask given in the AcuAccess file. Any new files created by the runtime have permissions based on that umask.

The umask is a variable having a three-digit octal value, similar to that used by **chmod**, but which describes the permissions that are *not* to be set on new files. The value of each digit, subtracted from seven, gives the corresponding **chmod** value. For instance a umask of 002 corresponds to a **chmod** value of 775 (however, because execute permission is not applicable to data files, AcuConnect actually sets the **chmod** value to 664). A umask of 002 grants read and write permissions to user and group, and read only permissions to other. Another common umask is 007, which sets read and write permissions for user and group and no permissions for other. For more information about umask, see your UNIX operating system documentation.

2.3.3 The Server Access File

The foundation of AcuConnect system security is the server access file, an encrypted Vision file named "AcuAccess" by default. This file contains a database of access records that determine which machines and which users are allowed to use AcuConnect. AcuConnect searches for this file in the /etc directory on UNIX servers and the operating system drive:\etc directory on Windows servers. You may rename the file if you like.

The server access file is designed to support a wide range of access security, from very permissive to very restrictive. You choose the level of security appropriate for your application.

Access records may include wild cards that allow all clients or all users (except root under UNIX and Administrator under Windows) access to AcuConnect. Or you can create individual access records for each user of each client. By having individual access records, you can restrict access to only those users specifically named in the access file.

The individual access records allow you to specify the user ID that AcuConnect will use when executing requests for users matching the given record. In this way, you can assign a user ID that has exactly the privileges needed and no more (typical of group access accounts).

In addition, every access record can include a password entry that the application or user must match before AcuConnect establishes a connection. (However, if the SECURITY_METHOD configuration variable is set to "LOGON", the local account password can be used instead.)

Note: We recommend that you use native system security rather than AcuConnect system security. On Windows 2008 it is essentially required that you use system security. To use native security, you set the SECURITY_METHOD variable in *both* the runtime configuration file on the client and server configuration file on the server. You still create a server access file containing access records that define your user base, but the server access file is used only to check if the user connecting to the server is allowed to connect, and to check to which local account the connection should be mapped.

The security system is almost completely transparent to the end user. Only when remote file access requires interactive password authentication is the user made aware of the security system.

On UNIX servers, the access file must be owned by root. The access file cannot be writable by anyone other than root. If the access file does not exist, is not owned by root, or is writable by users other than root, AcuConnect does not start.

On Windows servers, you should protect the access file by allowing only the administrator or someone in the administrators group to have write access to it. If the access file does not exist, is not owned by administrator or the administrators group, or is writable by users other than administrator or the administrators group, AcuConnect does not start.

Using an existing AcuAccess file

The server access file for AcuConnect is structured the same and named the same as the server access file for AcuServer®. If you are using AcuServer for UNIX, you can use your existing AcuAccess file in conjunction with AcuConnect, or you can set up a separate file for AcuConnect, if desired. If you do not want to modify your AcuServer AcuAccess file, you can set up a separate file for AcuConnect.

2.3.4 Access Record Composition

The server access file contains one or more access records. Each access record comprises the following fields:

Client Machine Name	The official host name by which the client machine is identified on the TCP/IP network. For distributed processing on Windows, this name is found in the Host field under Control Panel/TCP/IP Properties/DNS Configuration. Under UNIX, use the “hostname” or “**uname** -n” command to determine the name.
Client Username	The user’s login name on the client system.
Local Username	The local user name that AcuConnect will use when fulfilling requests for the client user

Password	Optional password protection. When used, the requester must supply a password that matches this field.
umask	A three-digit file creation mask

A typical server access record might look like the following:

Client Machine Name	**Client Username**	**Local Username**	**Password**	**umask**
starling	bernie	bsmith	. . . .	002

This record allows user *bernie* to connect from machine *starling*. AcuConnect uses the local user name *bsmith* (Bernie’s account on the file server) when executing requests for *bernie*. In this case, “. . . . . .” represents Bernie’s local Windows password, which is required if Windows security is used.

Four fields—Client Machine Name, Client Username, Local Username, and Password—each have a wild card value that is used to indicate a general behavior. These wild cards are:

Field name	**Wild card**	**Meaning**
Client Machine Name	*	Match all client machines
Client Username	(empty field)	Match all client users
Local Username	same as client	Use the Client Username
Password	*	Deny access to user

When the string “same as client” is specified in the Local Username field, certain conditions apply. If Client Username is not a valid name on the server, DEFAULT_USER is used. DEFAULT_USER is used also if the Local Username field is blank.

Again, if DEFAULT_USER is used to connect to AcuConnect on an NTFS partition under Windows NT, 2000 through 2008 be sure that DEFAULT_USER has both READ (RX) permissions on “wrun32.exe” and the appropriate permissions to access any file.

Common access records example

For illustrative purposes, here is a set of common access records:

Client Machine Name	**Client Username**	**Local Username**	**Password**	**umask**
support-pc		techie	<none>	002
warehouse-pc			<none>	002
president-pc	diamond	<same as client>	<none>	002
robin		<same as client>	<none>	002
starling	felice	<same as client>	<none>	002
starling	baxter		......	002
swallow	hartley	hartley	<none>	002
swallow		acct	<none>	002
raven			*	002

These entries are interpreted as follows:

- The entry for *support-pc* allows any user of *support-pc* to use AcuConnect. AcuConnect will use the local user name *techie* when executing requests for *support-pc*.
- The entry for *warehouse-pc* allows any user of *warehouse-pc* to use AcuConnect. Because the Local Username field is empty, AcuConnect will use the value of DEFAULT_USER as the local user name when executing requests for *warehouse-pc*.
- The entry for *president-pc* allows user *diamond* to access AcuConnect. Because the Local Username field holds “same as client,” AcuConnect will attempt to use *diamond* as the Local Username. If *diamond* is not a valid local user name, the value of DEFAULT_USER will be used.
- The entry for *robin* allows all users of *robin* to access AcuConnect. If the requester has an account on the server by the same name, AcuConnect will use that name; otherwise, AcuConnect will use the value of DEFAULT_USER.
- The first entry for *starling* allows user *felice* to access AcuConnect. AcuConnect will follow the same rules as the previous entry to assign a local user name.

- The second entry for *starling* allows user *baxter* to access AcuConnect. AcuConnect will use the value of DEFAULT_USER when executing requests for *baxter*. *baxter* will need to provide a password before a connection will be established.
- The first entry for *swallow* allows user *hartley* to access AcuConnect. AcuConnect will use the local user name *hartley* when executing requests for *hartley*.
- The second entry for *swallow* allows all users of *swallow* to access AcuConnect. AcuConnect will use the local user name *acct* for all users of *swallow*, except *hartley* (or other records for *swallow* that explicitly name a client user).
- The entry for *raven* denies any user of *raven* access to AcuConnect.

The most permissive access record that can be created is:

Client Machine Name	**Client Username**	**Local Username**	**Password**	**umask**
*		<same as client>	<none>	002

This record allows any user of any client to use AcuConnect, as long as the user has an account on the server by the same name, or DEFAULT_USER is defined with a valid user name (DEFAULT_USER cannot be defined to be root). A client user running as root will be mapped to DEFAULT_USER.

2.3.5 Using the Access File Manager

Create and maintain the server access file with the access file manager utility. To start this utility, use the "**acurcl** -access" command or the Access tab in the AcuConnect graphical control panel in Windows. To use the access file manager, you must be logged on to a UNIX server as root or superuser or on to a Windows server from the administrator account or from an account that belongs to the administrators group.

Note: To use the access file utility on Windows 2008 where User Access Control (UAC) security is turned on (as it is by default), any user must choose "Run as Administrator" in order to use the various AcuServer utilities. UAC can be turned off, in which case the user must merely be a member of the administrators group in order to fully operate the utility.

The access file manager is an interactive, menu-driven utility that allows you to do the following:

- Create an access file
- Add an access record
- Remove an access record
- Modify an access record
- Display an access record(s)

When the access file manager displays a value inside a pair of square brackets ([]), that value is the default value for the field. To accept the default value, press **Return**. In the following example, */etc/AcuAccess* is the default value.

```
Enter the name of the Server Access File
Filename [/etc/AcuAccess]:
```

2.3.5.1 Starting the access file manager

To launch the access file manager, use the “**acurcl** -access” command, as shown:

```
acurcl -access
```

2.3.5.2 Creating or opening an access file

When the access file manager starts, you are prompted for the path and name of the server access file:

```
Enter the name of the Server Access File
Filename [/etc/AcuAccess]:
```

- To open an existing access file

  Enter the path and file name of interest. If the specified server access file is not found, the manager asks if you want to create the file:

  ```
  'access-file' does not exist.
  Do you want to create it [N]?
  ```

- To create a new access file

  Enter “Y”. After opening or creating the access file, the manager displays a menu of five options:

```
Server Access File Option:

1. Add a security record
2. Remove a security record
3. Modify a security record
4. Display one/all security records
5. Exit

Enter choice [4]:
```

2.3.5.3 Adding an access record

1. Start the access file manager.
2. Select option [1] from the main menu.

   The manager presents a series of five prompts, one for each field in the record. The first field is the Client Machine Name field. The prompt looks something like this:

   ```
   A value of "*" for client machine name means that
   this record will match all clients for which there
   are no other records.  You cannot use alias names.
   The name must be the official machine name.  Enter
   the official machine name [*]:
   ```

   Enter a client machine name or accept the default value.

3. The second prompt is for the Client Username. If this field is blank, any user name will match.

   ```
   If no client user name is entered it implies any user.
   Enter client user name []:
   ```

   Enter a client user name or accept the default value.

4. The third prompt fills in the Local Username field. The Local Username is the name that AcuConnect will use when executing access requests for requesters that match the first two fields of this record. Note that if the Local Username is not a valid name on the server, the server will attempt to use the value of the server configuration variable DEFAULT_USER (if defined). If DEFAULT_USER is not defined, the connection will be refused (AcuConnect returns an error 9D,103).

```
A value of 'same as client' for local user name
means to use the client user name.  If no local
user name is entered DEFAULT_USER is used.
Enter the local user name [same as client]:
```

Enter a local user name or accept the default value.

5. The fourth prompt allows you to specify a password that must be supplied by requesters who match this record.

   Enter a password up to 64 characters long. The set of allowable characters includes upper and lower case letters, numbers, the space character, and most special characters (all ASCII characters numbered 32 to 126). Delete, escape, and other non-printable characters are not allowed.

   The password characters are not echoed on the screen when entered. You are asked to enter the password a second time to verify that it was entered correctly.

   ```
   If no password is entered it implies none.
   Enter password []:
   Retype password for verification:
   ```

   If the password verification fails, you see the following message:

   ```
   Mismatch - try again.
   If no password is entered it implies none.
   Enter password []:
   ```

6. The final prompt allows you to specify a umask. Enter the umask of the runtime process started for the user. Section 2.3, "Establishing System Security," provides more information on determining this value.

   ```
   The umask defines the file creation mask for all
   files created by this user.  It must be an octal
   value between 000 and 777.
   Enter umask [002]:
   ```

   Enter a umask value or accept the default value. If you enter an invalid umask value, you see this message:

   ```
   Invalid value for umask - try again.
   ```

After you specify a valid umask, the access file manager adds the record to the server access file.

```
Record added.
Press <Return> to continue...
```

If you accept all of the defaults when creating the record, the entry looks like this:

Client Machine Name	**Client Username**	**Local Username**	**Password**	**umask**
*		<same as client>	<none>	002

This record entry matches any client and allows any user to connect to the server, provided that:

- The user has an account of the same name on the server

  or

- The DEFAULT_USER variable is defined with the name of a valid user

Through inclusion or exclusion of wild cards, named entries, passwords, and umasks, it is possible to construct a server access file that allows open, unrestricted access; rigid, tightly controlled access; or almost any level in between.

2.3.5.4 Removing an access record

1. Select option [2] from the main menu.

   The manager presents two prompts:

   ```
   Enter official client machine name:
   Enter client user name:
   ```

2. At the first prompt, enter the official client machine name.
3. At the second prompt, enter the client user name.

   If a matching record is found, it is removed and the following message appears:

```
Record removed
Press <Return> to continue...
```

2.3.5.5 Modifying an access record

1. Select option [3] from the main menu.

   The access file manager presents the following prompts:

   ```
   Enter official client machine name:
   Enter client user name:
   ```

2. At the first prompt, enter the official client machine name.
3. At the second prompt, enter the client user name.

   If a matching record is found, the following message appears:

   ```
   Here is the record that was found:
   Client machine name:
   Client user name:
   Local user name:
   Password
   Umask:

   Do you want to modify the local user name [N]:
   ```

4. Type “Y” to modify the local user name.

   If you have a different local user name and want to change the record to “same as client,” you can type “same as client” at this prompt. Upper/lower case is ignored, but you must put a single space between the words. Don’t enter brackets or quotes.

   If you enter “Y” or “same as client” at this prompt, you are asked:

   ```
   Do you want to modify the password [N]:
   ```

   If you want to modify the password, enter a new one now.

5. After the password value is entered, you are prompted:

   ```
   Enter a new umask [002]:
   ```

   If you want to change the umask, enter the new value here.

After this series of prompts is answered, the record is updated.

```
Record modified
Press <Return> to continue...
```

2.3.5.6 Displaying an access record

1. Select option [4] from the main menu.

   You can display records to the screen, or you can output all of the records to a file. The manager first asks if you want to display the records to the screen.

   ```
   Display to the screen [Y]?
   ```

2. To display the records to the screen, type “Y”. The manager responds with the following prompt:

   ```
   Display all records [Y]?
   Displaying all records...
   ```

   a. To view an individual record, respond “N” (“no”) to “Display all records?” and the manager displays the following prompts:

      ```
      Enter official client machine name:
      Enter client user name:
      ```

      If a match is found, the record appears; otherwise, the manager states that no match was found and returns to the main menu.

      ```
      No matching record found.
      Press <Return> to continue...
      ```

   b. To copy all of the records to a file, respond “N” to the “Display to the screen?” prompt. The manager then prompts for a file name.

      ```
      Enter the name of the Server Report File
      Filename [AcuAccess.rpt]:
      ```

   c. Enter a file name or accept the default. The manager copies the access records to the named file.

      ```
      Creating 'AcuAccess.rpt'
      ```

2.4 Creating a Server Alias File in Thin Client

Before using AcuConnect for thin client operations, you must create a single association file to hold all the information that will be needed to invoke the appropriate application on the server. This file is in XML format.

Note: Alias files created prior to Version 7.0 are in ".ini" format. These files are automatically converted to XML format on first use with Version 7.0 and beyond. If you would prefer your alias file to be in ".ini" format, set the ALIAS_FILE_IS_XML variable to "FALSE" in your environment when executing "**acurcl** -alias" (or "**acurcl**" with the graphical control panel on Windows). Alias files in XML format are incompatible with all AcuConnect versions prior to 7.0.

---

1. To create the server alias file, log on to the server as the superuser (UNIX) or administrator (Windows) and type:

   ```
   acurcl -alias
   ```

2. When you execute "**acurcl** -alias", you are prompted for the necessary information. First, you are asked for the file to modify or create:

   ```
   Enter the name of the Alias file: [/etc/acurcl.ini]
   ```

   In VMS, the following prompt appears:

   ```
   Enter the name of the alias file: [acurcl.ini]
   ```

   Type the name of the file you want to use, or press **Return** to use the default file. If the file does not exist, AcuConnect asks if you want to create it. If you answer "Yes", it continues with processing as below, and if you answer "No", it asks for a new file name. The menu looks like this:

   ```
   RCL Alias file options
   1) Add an alias entry
   2) Remove an alias entry
   3) Modify an alias entry
   4) Display alias entries
   5) Exit
   Enter Choice [4]:
   ```

In Windows, you can also perform these tasks in the graphical control panel's Alias tab.

2.4.1 Adding aliases

1. To add an alias to the alias file, choose item [1] from the alias menu. You will receive the following prompt:

   ```
   Add an alias
   Enter the alias name:
   ```

2. Enter the name of the alias you want to add. If the named alias already exists in the alias file, a message displays:

   ```
   Alias already exists
   ```

   and you are returned to the alias menu. Otherwise, you are asked for the working directory for the alias:

   ```
   Enter the working directory:
   ```

3. Type an existing directory name. (Note that AcuConnect does not verify that this directory actually exists.) You are then asked for a command line for the runtime:

   ```
   Enter the command line:
   ```

4. Type the command line that you would normally type when executing the runtime, except do NOT include the runtime name. The command line is parsed and the alias entry is added to the alias file. If there are incorrect parameters in the command line, the parsing fails and the alias is not added to the alias file. Note that if you inadvertently add a space, tab, or special character to the name of your program, it may not be found.

AcuConnect keeps track of when the file was last modified, and if the modification date of the alias file is later than the tracked date, the file is reread. This means that you can add information to the alias file while AcuConnect is running, and the next time that alias is requested by a client it uses the new information.

Example

To add an alias entry named “tour” that runs the **tour** program with the command “**runcbl** -le err tour”, you would have the following interaction:

```
C:\Acucorp\AcuCBL8xx\AcuGT\bin> acurcl -alias
Enter the name of the Alias file: [/etc/acurcl.ini]
  myalias.ini

'myalias.ini' does not exist. Do you want to create it [N]?
  y

RCL Alias file options
1) Add an alias entry
2) Remove an alias entry
3) Modify an alias entry
4) Display alias entries
5) Exit

Enter Choice [4]: 1
Add an alias
Enter the alias name: tour
Enter the working directory:
C:\acucorp\acucbl8xx\acugt\sample
Enter the command line:
-le err tour
Press <Return> to continue...

RCL Alias file options
1) Add an alias entry
2) Remove an alias entry
3) Modify an alias entry
4) Display alias entries
5) Exit

Enter Choice [4]: 5

C:\Acucorp\AcuCBL8xx\AcuGT\bin>
```

2.4.2 Removing aliases

1. To remove an alias from the alias file, choose item [2] from the alias menu.

   You receive the following prompt:

   ```
   Remove an alias
   Enter the alias name:
   ```

2. Enter the name of the alias that you want to remove.

   If the named alias does not exist in the alias file, a message displays:

   ```
   Alias does not exist
   ```

   and you are returned to the alias menu. Otherwise, you see the current working directory and command line for the alias, and are asked to confirm deleting the entry:

   ```
   Are you sure you want to remove this alias?
   ```

3. Enter “Y” or “N”.

   In either case, you are returned to the alias menu. If you answered “Y”, the alias is no longer an entry in the alias file.

2.4.3 Modifying aliases

1. To modify an alias in the alias file, choose item [3] from the alias menu.

   You receive the following prompt:

   ```
   Modify an alias
   Enter the alias name:
   ```

2. Enter the name of the alias that you want to modify.

   If the named alias does not exist in the alias file, a message displays:

   ```
   Alias does not exist
   ```

   and you are returned to the alias menu. Otherwise, you see the current working directory and are asked for a new one.

3. Press **Return** to leave the working directory unchanged, or type the name of a different directory.
4. After you enter a directory, you see the command line as it currently exists in the alias file, and are asked for a new command line. Press **Return** to leave the entry unchanged, or enter a new command line. You are then returned to the alias menu.

2.4.4 Displaying aliases

To display an alias in the alias file, choose item [4] from the alias menu. You receive the following prompt:

```
Display aliases
Enter the alias name:
```

Enter a valid alias name, or enter nothing to see a list of valid aliases contained in the file. If you enter a valid alias name, you see the working directory and the command line for that alias. You are then returned to the alias menu.

Note: AcuConnect rewrites the command line before putting it into the alias file, so when it is displayed, it might not be exactly the same as what was originally entered. For example, displaying the "tour" alias above shows that the command line is "-el err tour".

2.5 Installing Server Programs

The last step in preparing your server for use with AcuConnect is copying the server components of your COBOL application onto the server into the desired directory. If you created a runtime configuration file for the server application, you must copy this onto the server as well.

AcuConnect supports the following server platforms:

- UNIX
- Windows NT Server
- Windows 2000, 2003, 2008 Server

2.6 Starting AcuConnect

For a client process to access programs on the server, AcuConnect must be running. To start and stop AcuConnect (as well as to perform several system administration functions such as checking system status, maintaining the server access file, or installing AcuConnect as a Windows service), you issue the **acurcl** command on the server at the command line. In Windows, using the **acurcl** command with no options brings up a graphical control panel. You can start AcuConnect from the Services tab of this screen.

This section provides specific instructions for issuing the "**acurcl** -start" command on the server. Chapter 8 describes AcuConnect's systems administrator functions and provides a complete reference to **acurcl** command formats and options.

2.6.1 Starting AcuConnect on a UNIX Server

1. Log in as root or superuser.
2. Issue the "**acurcl** -start" command on the server command line. Valid options to "-start" are listed in the following table:

Option	**Description**
-c	Specifies the name and path of the server configuration file. The "-c" must be followed by a space and then the path and name of the server configuration file. When "-c" is not used, **acurcl** looks for the configuration file in its default location: /etc/acurcl.cfg for UNIX or \etc\acurcl.cfg for Windows.
-d	Starts AcuConnect on the server machine in debug mode. AcuConnect runs in the foreground. Note that you are limited to one UNIX client connection when you use this option.
-e	Causes error output from **acurcl** to be placed in the named file. Follow "-e" with a space and then the path and name of the error output file (use this same syntax for the "-ee" and "+e" options listed below). • If "-e" is not specified, **acurcl** attempts to direct error output to /dev/console. • If /dev/console cannot be opened, **acurcl** attempts to append to a file named "acurcl.err" in the current directory. • If the "acurcl.err" file doesn't exist, or the file append fails, **acurcl** prints the message "acurcl: can't open error output file" to standard output, and **acurcl** terminates.

Option	Description
	You can use “-ee” to redirect the contents of **stderr** into the new error file. You can use “+e” to append error output from **acurcl** to the named file.
-f	By default, **acurcl** runs in the background. Use the “-f” option to run **acurcl** in the foreground. When run in the oreground, the **acurcl** process traps normal keyboard signals, such as **Ctrl+C**. If combined with the “-t” option, the “-f” option causes **acurcl** to display tracing and transaction messages directly to the screen. However, if the “-e” switch is used, all messages are placed in the named log file.
-g	Causes the file specified by the “-e” option to be compressed using the gzip compression method. Because the error file name is not modified to reflect the compression, we recommend that you specify a different file extension for the error file (for example, “.gz”) when you use this option. To read the file, you must decompress it with gzip.
-l	Causes a listing of the server configuration file to be printed to standard error output. This can be helpful when you are debugging problems that may be related to configuration variables. When this option is combined with the “-e” option, the listing is captured in the error output file.
-n	Assigns a port number to one instance of the **acurcl** daemon. The “-n” must be followed by a space and then an integer, for example, “5633”. This overrides the ACURCL_PORT value set in the server configuration file. Thin client applications can be assigned to a particular instance of the **acurcl** daemon via the <server:port> notation on the **acuthin** command line. Thin clients launched from a Web page can have the port number specified by the atc-port variable in the thin client command-line file (see section 7.3.1, “Thin Client Command-line Files.”) The **acurcl** daemon can work with privileged port numbers (from 0 to 1023), and with non-privileged port numbers (1024 and higher). Privileged port numbers are useful for external, secure applications.

Option	Description

	Note: If you start **acurcl** on two ports, you must also specify all start-up arguments, including the configuration file, as in:
	`acurcl -start -n 5632 -c c:\etc\config1 [other options]`
	`acurcl -start -n 5633 -c c:\etc\config2 [other options]`

-t #	Turns on the tracing function. When combined with the “-e” option, trace information is placed in the named error file. The “#” symbol represents the type of tracing or logging to be performed, as described below: • “1” provides information about access file match attempts. The trace information buffer is flushed to the error file when the buffer is filled or **acurcl** terminates. • “2” provides information about runtime requests. The buffer is flushed to the error file when the buffer is filled or **acurcl** terminates. • “3” provides the information described for “1” and “2”. • “5” is equivalent to “1”, but the tracing buffer is flushed to the error file each time an access file match is requested. (File trace flushing can also be controlled with the FILE_TRACE_FLUSH server configuration variable. See section 3.3, “Creating a Server Configuration File.”) • “6” is equivalent to “2”, but the tracing buffer is flushed to the error file each time a runtime is requested. • “7” provides the information described for “5” and “6”.

You can stop AcuConnect at any time by issuing the “**acurcl** -kill” command. See section 8.2.1.6, “acurcl -kill,” for a discussion of the “-kill” command options.

AcuConnect returns a status code of “0” after a successful start. When AcuConnect fails to start, it returns one of the following status codes to the operating system, indicating the reason for the failure:

Code	**Reason for failure**
1	Invalid argument
2	Missing license
3	Expired license

4	Unable to open the error log file
5	Unable to open the configuration file (writable by other than root)
6	Unable to open the AcuAccess file (missing or writable by other than root)
7	Unable to create a child process
8	Too many servers running
9	Unable to create a socket, because the socket is used by something other than AcuConnect
10	AcuConnect is already running on this port.

How you get this status depends on your shell. When you use the Bourne shell (or a compatible shell), echo $? to see the return status of any program executed from the shell.

Setting the user environment in UNIX

With the use of USE_UNIX_SHELL configuration variable, you can direct AcuConnect to attempt to read your UNIX shell login files based on the shell named in /etc/passwd when AcuConnect starts a runtime. Setting this variable to “true” means that AcuConnect can start a runtime through your login files, and the environment variables you set in these files are therefore available to the runtime.

AcuConnect tries to convince the shell to read the startup files by making the first character of argv[0] = ‘-’, just as login(1) does. As a result, the following environment variables are set:

- HOME (from /etc/passwd)
- SHELL (from /etc/passwd)
- PATH /usr/local/bin:/bin:/usr/bin:
- TERM dumb
- LOGNAME $USER

Your startup files may modify these variables or set other variables.

AcuConnect starts the shell as shown:

```
    shell -c "'runcbl' . . ."
```

where *shell* is the user's shell and the ellipsis (". . .") designates the position of the arguments that AcuConnect would normally use to start the runtime. The full path to **runcbl** is also given. Make sure that the shell used supports the "-c" option and the ability to set argv[0] to "-". For example, the following shells can be used: sh, ksh, and bash. The csh and tcsh shells are not supported.

2.6.2 Starting AcuConnect on a Windows Server

Issue the "**acurcl** -start" command from the administrator account or from an account that belongs to the administrators group. On Windows servers, it's best to specify "**acurcl** -start" with no options. Alternatively, you can click the **Start** button on the Windows graphical control panel's Services tab.

Note: To use the AcuServer Control Panel (UAC) on Windows 2008, where User Access Control security is turned on (as it is by default), any user must choose "Run as Administrator" in order to use the various AcuServer utilities. UAC can be turned off, in which case the user must merely be a member of the administrators group in order to fully operate AcuServer.

You can stop AcuConnect at any time by issuing the "**acurcl** -kill" command or by clicking **Stop** on the control panel's Services tab. See section 8.2.1.6, "acurcl -kill," for a discussion of the "-kill" command options.

2.6.3 Starting AcuConnect in Thin Client on MPE/iX

To start the server process on an MPE/iX host, you need to run the process as an MPE batch job. The job file might be called "JACURCLD" and might look something like this:

```
!JOB JACURCLD,MGR.ACUCOBOL;PRI=CS
!RUN /ACUCOBOL/bin/acurcl;&
!  INFO='-start -c /ACUCOBOL/etc/acurcl.cfg -le
                      /ACUCOBOL/bin/acurcl.err -t3';&
!  PRI=CS
!EOJ
```

- To stream the job, you could do the following, assuming you are in the MPE group where the job file is:

  MPE:

  ```
  :STREAM JACURCLD
  ```

  POSIX:

  ```
  shell/iX> callci "STREAM ./JACURCLD"
  ```

- To stop the job, you would do the following:

  MPE:

  ```
  ABORTJOB #[job-number]
  ```

- Use SHOWJOB to obtain the job number, if necessary.

  POSIX:

  ```
  shell/iX> /ACUCOBOL/bin/acurcl -kill
  ```

2.6.4 Starting AcuConnect in Thin Client on a VMS Server

To start AcuConnect on VMS (version 7.2 or later) as a detached process, use the following command line:

```
run/detached sys$system:loginout /input=acurcl.com -
 /output=acurcl.out /process_name=acurcl
```

2.6.5 Starting AcuConnect at System Startup

On a UNIX server, to start AcuConnect whenever the server boots, add the "**acurcl** -start" command to the system boot file. Your entry might be similar to the following:

```
#
# If the acurcl executable is present,
# start acurcl
if  (test -f  /acucobol/acurcl)  then
   echo Starting acurcl > /dev/console
   /acucobol/acurcl -start -e /acucobol/acurcl.log
fi
```

On a Windows server, we recommend that AcuConnect be installed as a Windows service. This causes the service to be started automatically each time the system is booted. If you don't install it as a service, it will stop whenever you log out.

We provide an installation script that gives you the option to install the **acurcl** daemon as a Windows service. If you choose not to install the service from the installation script, you can install it as a service in the following way. You must be logged on to an account that belongs to the administrators group. From the command line, type:

```
acurcl -install
```

You can also click **New** in the graphical control panel's Services tab. The service will now be started automatically each time the system is booted.

Immediately following the installation, you can start the service from the graphical control panel's Services tab by clicking **Start**. As an alternative, you can use the Start/Settings/Control Panel/Services menu option.

Please note that installing a service on a particular port resets all start-up options for the service on that port. You can use all valid "-start" options when installing AcuConnect as a service. These options are stored so that the service will use them when starting.

2.7 AcuConnect Connection Logic

How the client connects to AcuConnect depends on the client application that is attempting to connect. However, some common steps are involved with any attempt, regardless of the client application. These are described here to clarify the use of the server access file and the DEFAULT_USER configuration variable.

To validate a requester's access privileges, AcuConnect does the following when a client process first makes a request to AcuConnect:

1. Opens the server access file.
2. Searches for a record that matches both the client machine name and the client user name.
3. (If no match is found) searches for a record that matches the client machine name and a "match all" (blank) client user name.

4. (If no match is found) searches for a record that has the "match all" ("*") client machine name and the client user name.
5. (If no match is found) searches for a record that has the "match all" ("*") client machine name and the "match all" (blank) client user name.
6. (If no match is found) refuses the connection.

When a match is found:

1. If the Local Username is valid, it is used.
2. If the Local Username is not valid, DEFAULT_USER is used.
3. If the Local Username is not valid and DEFAULT_USER is not valid, the connection is refused.
4. If the Local Username is valid and the password field is defined, a message is sent back to the requester asking for a password.

When a connection is established, AcuConnect invokes the runtime based on the information provided by the client making the request. AcuConnect connects that child runtime with the client and then removes itself from the communication loop between the two processes.

2.7.1 Passwords

When a password is assigned to an entry in the server access file, requesters who match that entry must return a matching password to AcuConnect. In a distributed processing environment, the client application has two options for acquiring and sending a password back to AcuConnect. In thin client, only option 2 applies.

Option 1: Program Variable

The requesting application may include code that checks for the program variable *acu_client_password*. If defined, its value is considered an unencrypted password, which is then encrypted and sent to AcuConnect for verification. If the value does not match the value in the access record, the connection is refused. Using *acu_client_password*, the COBOL programmer has a great deal of flexibility in setting and acquiring the password. The programmer can supply a password to AcuConnect without requiring any user interaction (the user may remain unaware that a password is required).

To use *acu_client_password*, declare an external pic x variable named *acu_client_password* in Working-Storage:

```
ACU_CLIENT_PASSWORD     PIC  X(64) IS EXTERNAL
```

Assign a value to the variable before the program's first access to a remote file (or better, before the program's first access to any file). Note that the value of *acu_client_password* should be terminated by LOW-VALUES.

Option 2: User-entered Password

If *acu_client_password* is not defined, the client runtime opens a dialog window requesting that the user enter a password.

```
A password is required to connect to host hostname.
Please enter a password:
```

The user must enter a password. The characters do not echo on the screen.

The password is then encrypted and sent to the server for verification. If the password matches, a connection is established. If the password doesn't match, or if the user enters a blank password (for example, **Enter** or **OK**), the user is prompted again to enter a password:

```
Invalid password
Please enter a password:
```

The password verification cycle is repeated until a valid password is entered, or the value of the server configuration variable PASSWORD_ATTEMPTS is exceeded (the default value is "3"). The text displayed by the runtime to prompt for a password and report a failed verification can be modified with the TEXT runtime configuration variable. See section 3.3, "Creating a Server Configuration File," for more information about these configuration variables.

2.7.2 Exiting the Access Manager

To exit the manager, select option [5] from the main menu.

3 Server Configuration

Key Topics

3.1 Configuring the AcuConnect Server

After you install AcuConnect, you need to configure the server. At a maximum, this can entail configuring the environment, creating a server configuration file for AcuConnect, and creating a runtime configuration file for the server component of your COBOL program. If you accept the default security file locations and other settings, however, no special configuration is required.

The client machine also requires configuration; this is discussed in Chapter 5, "Chapter 5: Preparing the Client(s) in Distributed Processing" and Chapter 6, "Chapter 6: Preparing the Client(s) in Thin Client."

3.2 Configuring the Environment

Environment variables are values maintained by the host operating system that can be changed by the user. These values can be used to store information such as the location of executable programs and header files. Following is an environment variable that you may choose to set on your AcuConnect server if you want to specify the format of your alias file. For more information on environment variables, refer to the *ACUCOBOL-GT User's Guide*, section 1.5, "Environment Variables."

ALIAS_FILE_IS_XML

Alias files are stored on disk as XML files by default. If you want the alias files to be in INI format rather than in XML, set the ALIAS_FILE_IS_XML environment variable to "FALSE".

Note: Once the alias file is in XML format, setting this variable to FALSE won't revert it.

If you want to use this variable, you must set it in the environment when you are executing "**acurcl** -alias" (or **acurcl** with the graphical control panel on Windows), because AcuConnect reads no configuration files in this mode.

See your operating system documentation for how to set environment variables.

3.3 Creating a Server Configuration File

The server configuration file, called “acurcl.cfg” throughout this book, tells AcuConnect the location of the AcuAccess file, defines the default user, and supplies other information. The server configuration file is completely optional if you have stored AcuAccess in the default /etc or operating system drive:\etc directory (\etc for UNIX/Linux).

When shipped, AcuConnect comes with a sample “acurcl.cfg” file. If you choose to use a server configuration file, you can use the sample file as is (complete with its default values), or you can modify the file to your specification. By default, AcuConnect looks for “acurcl.cfg” in /etc or operating system drive:\etc on the server. If the file is given another name or is located in another directory, you must specify the full location and name of the file with the “-c” option when you start AcuConnect. For more information, see the description of the “-start” command in section 8.2.1.9, “acurcl -start.”

The server configuration file can contain any or all of the following variables:

Name	**Default Value**
ACCESS_FILE	/etc/AcuAccess
ACURCL_PORT	5632
AGS_BAD_SOCKET	undefined
ALIAS_FILE_IS_XML	TRUE
CHILD_WAIT	50
DEFAULT_USER	undefined
FILE_TRACE	0
FILE_TRACE_FLUSH	FALSE
FILE_TRACE_TIMESTAMP	FALSE
PASSWORD_ATTEMPTS	3
PROVIDE_PASSWORD_MESSAGES	FALSE
SECURITY_METHOD	NONE
SERVER_ALIAS_FILE	/etc/acurcl.ini (thin client only)

Name	**Default Value**
SERVER_IP	undefined
SERVER_NAME	undefined
SERVER_RUNTIME	wrun32.exe or runcbl in the **acurcl** directory
TEXT_015	"A password is required to connect to host %s."
TEXT_016	"Please enter a password: "
TEXT_017	"Invalid password."
USE_SYSTEM_RESTRICTIONS	FALSE
USE_UNIX_SHELL	FALSE
WINNT_LOGON_DOMAIN	undefined
WINNT_EVENTLOG_DOMAIN	NULL

Note that values assigned to variables contained in the server configuration file are applied solely to operations performed by AcuConnect. To gain a context for using these variables, read section 2.6, "Starting AcuConnect."

ACCESS_FILE

The ACCESS_FILE variable (along with DEFAULT_USER and PASSWORD_ATTEMPTS) affects AcuConnect access security.

ACCESS_FILE must hold the full path and file name of the server access file, if it is other than the default location (/etc/AcuAccess on UNIX systems; \etc\AcuAccess on the root drive on Windows servers).

For example:

```
    access_file c:\security\AcuAccess
```

ACURCL_PORT

The ACURCL_PORT variable specifies a particular port number to be used for accessing AcuConnect on the server host machine. This is especially helpful when the server host machine has a security firewall, because firewalls generally allow access only to specific ports. With this variable, the site can ensure that firewall restrictions are satisfied.

The **acurcl** daemon can work with privileged port numbers (from 0 to 1023) and with non-privileged port numbers (1024 and higher, up to 32767).

The value of this variable is overridden by any port number assigned on the command line (via the “-n” option) when the daemon is started.

Note: The ACURCL_PORT variable may be specified in the server configuration file. It may also be specified on the server command line with the “-n” option. If you specify a non-default value, you must use the same value on the client side (using the <server:port> notation).

AGS_BAD_SOCKET

AcuConnect depends on the ability to pass an open socket descriptor to the runtime for the client to communicate with the server. Some UNIX shells don’t pass all file descriptors to a child process. You may be experiencing such a problem if users can sometimes connect to the server, but other times can no longer connect, even though there are sufficient licenses.

If you detect such problems, look for a line in the server runtime error file that shows what descriptor the socket is using, as in the example below:

```
Runtime version 8.2.0
Configuration file = 'config'
Thin client socket uses descriptor 6
Loading map file
Unable to open TC-MAP-FILE '', error 0
Try loading 'testit.acu'...
testit.acu loaded
```

Set the AGS_BAD_SOCKET variable to this socket descriptor to alleviate the problem. You can set this variable multiple times, once for each bad descriptor that you detect. Each time you set it, however, you should set it to a single value. For example, if you find that descriptors “18” and “62” are causing problems, you can add the following two lines to the runtime configuration file:

```
AGS_BAD_SOCKET 18
AGS_BAD_SOCKET 62
```

Note: If the socket descriptor is in use at the time you attempt to set it, you will not be able to use the “-config” option of **acurcl**.

Also, please note that there is a chance that the setting will fail when using “-config”. You can determine whether this setting had any effect by looking at the error file. You will see a line something like:

```
7431977 - CONFIG request - SET AGS_BAD_SOCKET to n
Set parameter 'AGS_BAD_SOCKET' to n
```

Instead of *n*, you will see the value you are trying to set. The most important line is the second. If you *don’t* see that line, it means that **acurcl** has that descriptor value open for some other purpose (perhaps another client is trying and failing to connect). You should continue trying until you get the setting.

CHILD_WAIT

The CHILD_WAIT configuration variable lets you determine how long AcuConnect waits for a child runtime to start successfully. This variable is valid for Windows servers only. The default value (in milliseconds) is “50”. When you set this variable to “0”, AcuConnect does not wait at all. The maximum value is “32767”, or approximately 32 seconds.

This variable helps you determine whether you have reached a Windows limit on the number of simultaneous child runtimes. Refer to section 8.7.3, “Connection Refused,” for more information about this Windows limitation and how to work around it.

DEFAULT_USER

DEFAULT_USER holds the default user name given to AcuConnect requesters who are not specifically mapped to a local user name in the server access file (as when the Local Username field of the access record is empty, or the value of the field is an invalid user name). Definition of DEFAULT_USER is optional. DEFAULT_USER cannot be defined as "root" on UNIX systems or administrator (on Windows systems). For a description of when AcuConnect assigns DEFAULT_USER to a requester, see section 2.7, "AcuConnect Connection Logic."

FILE_TRACE

This variable allows you to start the tracing function without specifying a "-t #" option on the command line when you start AcuConnect. Setting this variable to a value between "1" and "7" saves connection information. The default value is "0". Refer to section 2.6, "Starting AcuConnect," for more information about trace values.

FILE_TRACE_FLUSH

Setting FILE_TRACE_FLUSH to "1" (on) causes AcuConnect to flush the error trace buffer to disk after every alias program execution. This function also ensures that in the event that the **acurcl** daemon terminates abnormally, all trace information, up to the last alias program execution, is captured in the error file. If this configuration variable is not used (set to 0, false, or off), and AcuConnect terminates unexpectedly, an undefined amount of trace information is lost. FILE_TRACE_FLUSH is set to "false" by default. Note that you can also enable trace flushing when you start the **acurcl** daemon by including the "-t #" command-line switch in the start command (see section 2.6, "Starting AcuConnect," for more information).

FILE_TRACE_TIMESTAMP

Setting this variable to "true" allows you to turn on trace file timestamping. The timestamp is added to the trace file at the beginning of every line. The default value is "false". Note that setting this variable to "true" may have a negative impact on performance.

PASSWORD_ATTEMPTS

The variable PASSWORD_ATTEMPTS (along with DEFAULT_USER and ACCESS_FILE) affects AcuConnect access security.

PASSWORD_ATTEMPTS holds a positive integer value specifying the total number of password validation attempts a requester is allowed before a connection attempt is terminated. The default value is "3". If a value of less than "1" is given, the value "1" is used. If a non-integer value is given, the default value is used.

PROVIDE_PASSWORD_MESSAGES

This variable controls whether the server sends password messages to the client when a remote connection is requested and passwords are required. When you set this variable to "true", the password messages defined in the TEXT_nnn configuration variable are sent to the client when it requests a connection to the server. If PROVIDE_PASSWORD_MESSAGES is set to the default value of "false", the client uses its own default values.

SECURITY_METHOD

This variable lets you determine the security method AcuConnect employs for user logon. AcuConnect can use the operating system's native logon facility or the security provided by the AcuAccess file. Users can use their regular passwords when they connect to servers, instead of having a different password in the AcuAccess file or remembering to keep the password in the AcuAccess file coordinated with the native password.

Note: We recommend that you use native system security rather than AcuConnect system security. On Windows 2008 it is essentially required that you use system security. To use native security, you set the SECURITY_METHOD variable as described in this section. You still create a server access file containing access records that define your user base, but the server access file is used only to check if the user connecting to the server is allowed to connect and to check to which local account the connection should be mapped.

This feature works for Windows servers and for UNIX servers. For UNIX servers, passwords can be stored in /etc/passwd or in /etc/shadow (in other words, the machine uses shadow passwords). It does not use the newer pluggable authentication module (PAM) libraries.

The default value of SECURITY_METHOD is "NONE", which means that AcuConnect's AcuAccess file security is used. For information about using the AcuAccess file, refer to section 2.3, "Establishing System Security."

When this variable is set to "LOGON", the operating system's native logon capability is used.

Windows

In Windows, AcuConnect attempts to log the user on to the domain specified in the WINNT_LOGON_DOMAIN configuration variable. AcuConnect first uses the password in the AcuAccess file to log the user onto the server. If the AcuAccess password matches the user's Windows domain password, the login completes and the user is never prompted for a password. If the password doesn't match, or if the password field in the AcuAccess file is empty, the user is prompted to supply a password. The password provided must match the user's network domain password on the Windows server. The number of attempts the user has to supply the correct password is limited by the value of the configuration variable PASSWORD_ATTEMPTS. A successful logon grants users all the access rights they would have if they were directly logged onto the server. AcuConnect allows Windows servers to manage all issues pertaining to access permissions.

UNIX

In UNIX, the AcuAccess password is checked against the password in the system files. If the password matches, the login is completed. If it doesn't match, the user is asked for a password that is then checked against the system password. If you want your UNIX machine to be able to restrict access to the machine based on various parameters, set the USE_SYSTEM_RESTRICTIONS configuration variable.

In distributed processing, SECURITY_METHOD must be set in both the client and the server configuration files, and the values must match. If the values don't match, the security method reverts to "NONE", which uses only the AcuAccess password.

In a thin client environment, the initial behavior is different from that of distributed processing. When the thin client establishes its configuration, it has a default setting of "LOGON". This value must match the SECURITY_METHOD setting on the server. If the values don't match, the security method reverts to "NONE", which uses only the AcuAccess password.

A value of "NAMED-PIPE" is for AcuServer use only and we do not recommend its use here.

Note: The SECURITY_METHOD configuration variable replaces the NT_SECURITY variable (which has been removed). If you are upgrading a Windows server, you must change your configuration file, replacing the NT_SECURITY variable with the SECURITY_METHOD variable.

SERVER_ALIAS_FILE

This variable applies only in a thin client environment. SERVER_ALIAS_FILE holds the name of the alias file to be used with the thin client. When the thin client is invoked, it is not given the name of a COBOL program to run. Rather, it is given the name of an alias to use. This alias holds all the information for invoking the runtime, which will be used to run the COBOL program. Refer to section 2.4, "Creating a Server Alias File in Thin Client," for information on creating the server alias file.

SERVER_IP

This variable is necessary only if your server interfaces with more than one network. In this instance, you can use SERVER_IP to specify the exact Internet Protocol (IP) address of the network interface card that you want AcuConnect to use. For example:

```
SERVER_IP 192.215.170.107
```

starts AcuConnect using IP address 192.215.170.107. If you have already assigned a host name to the desired IP address, you can use the SERVER_NAME variable instead of SERVER_IP to specify the precise network card to use.

Note that if both the SERVER_IP and SERVER_NAME configuration variables are defined in the configuration file, AcuConnect uses the value of the first valid entry in the file.

SERVER_NAME

You can use this variable in place of SERVER_IP to specify which network interface card to use when starting AcuConnect. It is necessary only if your server interfaces with more than one network, and it can be used only if you have already assigned a host name to the desired IP address. In this instance, you can use SERVER_NAME as follows:

```
SERVER_NAME <host-name>
```

where <*host-name*> is the host name assigned to the IP address of the desired network card. For example:

```
SERVER_NAME nts2
```

starts AcuConnect using host name nts2.

Note that if both the SERVER_IP and SERVER_NAME configuration variables are defined in the configuration file, AcuConnect uses the value of the first valid entry in the file.

SERVER_RUNTIME

This variable is used to specify the name of the runtime to use. The default is to use the runtime that is in the same directory as AcuRCL.

TEXT_*nnn*

You can use a set of TEXT configuration variables to control the text of selected AcuConnect messages. We have included three configuration variables to support AcuConnect password handling. The default text for each message is:

```
TEXT_015  "A password is required to connect to host %s."
TEXT_016  "Please enter a password:  "
TEXT_017  "Invalid password."
```

Note that in message 15, the name of the AcuConnect host will be substituted for the “%s” characters.

To change the text associated with a given message, perform the following steps:

1. Place the word “TEXT” at the beginning of a line in the configuration file
2. Follow this with:
   a. An underscore
   b. The message number
   c. An “=” character (or a blank space or tab)
   d. The text to be used when that message number is displayed.

For example:

```
TEXT_016=Enter your password now:
```

Note that to enable the display of the password messages, you set the PROVIDE_PASSWORD_MESSAGES variable to “true” (1, on, yes).

The use of password protection is optional. If password protection is not used, these messages will not be displayed.

USE_SYSTEM_RESTRICTIONS

This configuration variable applies only to UNIX systems. Some UNIX machines can restrict access to the machine based on various parameters. If you want to include those restrictions in AcuConnect, set the USE_SYSTEM_RESTRICTIONS configuration variable to “true”. The default value is “false”. This configuration variable is available only on UNIX servers and is used only if the SECURITY_METHOD configuration variable is set to “LOGON”. The server uses the **loginrestrictions()** function to implement this feature. Ask your system vendor for information about this function and how to establish restrictions.

USE_UNIX_SHELL

This configuration variable determines whether AcuConnect attempts to start a runtime through the user’s UNIX login shell. When USE_UNIX_SHELL is set to “true” (1, on, yes), AcuConnect reads the user’s login shell files based on the shell named in /etc/passwd when it starts a runtime for that user. As a result, the runtime has access to all the environment variables that the user has set in the startup file. The default value is “false”.

Note: Running AcuConnect in debug mode when USE_UNIX_SHELL is “on” in a distributed processing deployment may render the UNIX debugger display unreadable. This behavior occurs because the TERM environment variable is automatically set to “dumb” when USE_UNIX_SHELL is on. To avoid this behavior, you can add a line to your shell’s login script file (for example, “.profile”, “.bash_profile”, or “.cshrc”) that sets either the TERM or A_TERM environment variable to another value.

WINNT_LOGON_DOMAIN

This configuration variable applies only to Windows servers. When the SECURITY_METHOD configuration variable is set to “LOGON”, you can specify a domain name for user logon via another configuration variable. Setting WINNT_LOGON_DOMAIN to the name of a Windows domain logs the user on to that domain. The user logs on to the NULL domain if this variable is not set. Note that the WINNT_LOGON_DOMAIN setting applies only when SECURITY_METHOD is “LOGON”.

WINNT_EVENTLOG_DOMAIN

This configuration variable applies only to Windows servers. Set WINNT_EVENTLOG_DOMAIN to the Universal Naming Convention (UNC) name of the computer to which event log messages should be sent. This variable must be set in the configuration file and is not changeable. If this variable is not set, system logging information is sent to the local machine on which the server is executing. For information about event logging, refer to section 8.5, “Event Logging.”

Sample “acurcl.cfg” File

The following file is included in the /sample directory on your AcuConnect installation CD. Use this file as a starting point, then modify it for your needs.

```
"acurcl.cfg"
# This is a sample AcuConnect configuration file
# You should edit it to match your needs
# The following lines are commented out to show you the
# default values.  If you want to use a different value,
# then uncomment the line and change the value.
#Access_file /etc/AcuAccess
#default_user
#password_attempts 3
#server_port -1
# Windows NT specific configuration variables
#server_start_6997 -l -e acuconnect.log -c \etc\server2.cfg
```

3.4 Creating a Runtime Configuration File for the Remote Server Component

Because at least some application logic executes on the server, you may want to create a runtime configuration file for the remote server component of your application. If you create this file, you will install it on the server machine along with the remote application component. This file defines how the remote application should behave at runtime.

To tell AcuConnect where to find this file in a distributed processing environment, you must specify its full path using the ACUCONNECT_RUNTIME_FLAGS variable in the client configuration file, “client.cfg”.

Because the remote application is run using an ACUCOBOL-GT runtime, you can use any runtime configuration file variable described in the *ACUCOBOL-GT User’s Guide*. The following variables are significant for the distributed processing environment:

Name	**Default value**
AGS_MAX_SEND_SIZE	16000
AGS_RECEIVE_BUFFER_SIZE	16384
AGS_SEND_BUFFER_SIZE	16384
AGS_SOCKET_COMPRESS	NONE
AGS_SOCKET_ENCRYPT	0
AGS_TCP_NODELAY	TRUE

Runtime configuration files have a similar function in a thin client environment. The file is stored on the server along with the application, and its name is specified in a server alias file that you establish when first setting up the system. (See section 2.4, “Creating a Server Alias File in Thin Client,” for details). In addition to the variables in the previous table, the following variables are important for a thin client environment:

Name	Default value
FREEZE_AX_EVENTS	1
TC_AUTO_UPDATE_FAILED_MESSAGE	ACUCOBOL-GT Thin Client: Automatic update was unsuccessful
TC_AUTO_UPDATE_FAILED_TITLE	ACUCOBOL-GT Thin Client
TC_AUTO_UPDATE_NOTIFY_FAIL	TRUE
TC_AUTO_UPDATE_QUERY	1
TC_AUTO_UPDATE_QUERY_MESSAGE	See section 7.4.4, "Informing the User When an Update Is Needed."
TC_AUTO_UPDATE_QUERY_TITLE	ACUCOBOL-GT Thin Client
TC_AX_EVENT_LIST	undefined
TC_CHECK_ALIVE_INTERVAL	300
TC_CHECK_INSTALLER_TIMESTAMP	0
TC_CONTINUITY_WINDOW	0
TC_CONTROL_SYNC_LEVEL	1
TC_DELAY_ACTIVATE	1
TC_DELAY_PRE_EVENT_OPS	0
TC_DISABLE_AUTO_UPDATE	0
TC_DISABLE_SERVER_LOG	FALSE
TC_DOWNLOAD_CANCEL_MESSAGE	Please wait while the download is being cancelled . . .
TC_DOWNLOAD_DESCRIPTION	Downloading installation file . . .
TC_DOWNLOAD_DIALOG	1
TC_DOWNLOAD_DIALOG_TITLE	ACUCOBOL-GT Thin Client Automatic Update
TC_EVENT_LIST	undefined
TC_EXCLUDE_EVENT_LIST	0
TC_INSTALLER_ARGS	undefined

Name	**Default value**
TC_INSTALLER_CLIENT_FILE	<APPDATA>\ACUCOBOL-GT\ <*installer_server_filename*>
TC_INSTALLER_RUN_ASYNC	0
TC_INSTALLER_SERVER_FILE	<*runtime_path*>\acuthin.msi
TC_INSTALLER_TARGET_DIR	undefined
TC_INSTALLER_UI_LEVEL	DEFAULT
TC_MAP_FILE	undefined
TC_NESTED_AX_EVENTS	0
TC_QUIT_MODE	-1
TC_REQUIRES_BUILD_NUMBER	0
TC_RESTRICT_AX_EVENTS	0
TC_SERVER_LOG_FILE	autoupdate.%c.%p.log
TC_SERVER_TIMEOUT	20
TC_TV_SELCHANGING	1
THIN_CLIENT_ENCRYPT	0

AGS_MAX_SEND_SIZE

This variable lets you tune socket performance by determining the maximum size of each data packet sent across the network. The default value is “16000”. Tips on how to use this variable in a thin client environment can be found in section 8.8.2.1, “Buffer size considerations.”

AGS_RECEIVE_BUFFER_SIZE

This variable helps you tune one element of low-level socket communication between a client and server. AGS_RECEIVE_BUFFER_SIZE controls the size of the low-level receive buffer. The variable must be set before the creation of any sockets. The default value is “16384”.

Note that because ACUCOBOL-GT does not control this buffer, changes in its value are not displayed in response to a U debugger command to list runtime memory usage. Changes to the default value may have little, if any, noticeable effect. The default value is probably adequate in most situations. Tips on how to use this variable in a thin client environment can be found in section 8.8.2.1, "Buffer size considerations."

AGS_SEND_BUFFER_SIZE

This variable helps you tune one element of low-level socket communication between a client and server. AGS_SEND_BUFFER_SIZE controls the size of the low-level send buffer. The variable must be set before the creation of any sockets. The default value is "16384".

Note that because ACUCOBOL-GT does not control this buffer, changes in its value are not displayed in response to a U debugger command to list runtime memory usage. Changes to the default value may have little, if any, noticeable effect. The default value is probably adequate in most situations. Tips on how to use this variable in a thin client environment can be found in section 8.8.2.1, "Buffer size considerations."

AGS_SOCKET_COMPRESS

This variable controls data compression by the internal socket layer. Setting this variable to "ZLIB" means that data is compressed using the same algorithm as the gzip compression utility. A value of "RUNLENGTH" means that a simple compression based on counting repeated bytes of data is performed. RUNLENGTH compression tends to be faster than ZLIB compression, but tends not to compress as well, especially with large blocks of data. The default value for this variable is "NONE".

AGS_SOCKET_COMPRESS must be set *before* any socket communication is done, and it cannot be changed via the SET ENVIRONMENT statement. Tips on how to use this variable in a thin client environment can be found in section 8.8.2.2, "File compression."

Note that Windows supports ZLIB compression, but not all UNIX machines do. If ZLIB compression is not supported on a particular machine, a variable value of “ZLIB” will be ignored. When the system must negotiate the compression algorithm to be used with a server, the method that both machines support is used.

AGS_SOCKET_ENCRYPT

The AGS_SOCKET_ENCRYPT variable lets you encrypt any data passed over a socket. Set this variable to “1” (on, true, yes) if you want the internal socket layer to perform encryption. The default value is “0”. You must set this variable before any socket communication occurs, and you cannot change it via the SET ENVIRONMENT statement. Note that if the THIN_CLIENT_ENCRYPT variable is set to “1”, AGS_SOCKET_ENCRYPT is also set to “1”. You can still use the ENCRYPTION_SEED configuration variable with AGS_SOCKET_ENCRYPT.

AGS_TCP_NODELAY

This variable helps you tune one element of low-level socket communication between a client and server. The setting of AGS_TCP_NODELAY determines whether the Nagle algorithm is used to affect the frequency of socket communication. The Nagle algorithm is a method by which the transmission of small socket packets is briefly delayed so several can be sent at the same time. Setting this variable to “false” causes the algorithm to be used.

The default value is “true”, which results in immediate individual packet transmissions. The default value is probably adequate in most situations.

ENCRYPTION_SEED

The value of this variable initializes the encryption algorithm. You can change the default value of this variable to any text string of your choice. Note that AGS_SOCKET_ENCRYPT must be set to “true” for this variable to have an effect. It is not required for encryption to be enabled.

FREEZE_AX_EVENTS

This variable applies only in a thin client environment. During the processing of an ActiveX event, the Windows and thin client runtimes attempt to suspend subsequent ActiveX events until the first event has completed. By default, the thin client runtime also attempts to suspend ActiveX events whenever the application is not processing an ACCEPT statement. To suspend and resume events, the runtime calls the ActiveX function **IOleControl::FreezeEvents()**.

You might want to disable calls to "FreezeEvents" for ActiveX controls that discard events while in a "FreezeEvents" state. For example, if a user double-clicks in an ActiveX control, the control might generate three events: mouse-down, mouse-up, and double-click. If the COBOL program terminates an ACCEPT statement in response to the mouse-down event, the runtime calls **FreezeEvents()**, and the ActiveX control might discard the mouse-up and double-click events.

You can disable the **FreezeEvents()** logic by setting the FREEZE_AX_EVENTS runtime configuration variable to "0" (off, false, no) in the configuration file or programmatically with the SET verb. The default value of FREEZE_AX_EVENTS is "1" (on, true, yes).

Note that when this variable is set to "0", the setting of TC_RESTRICT_AX_EVENTS is ignored.

Note: The **FreezeEvents()** logic protects against unexpected nesting of ActiveX events and against event procedures running unexpectedly during a CREATE, DISPLAY, MODIFY, INQUIRE, or other operation that waits for results from the thin client. Turning this feature off can cause unexpected behavior.

TC_AUTO_UPDATE_FAILED_MESSAGE

This configuration variable applies only to the thin client automatic update feature. If this process fails, the following message appears:

```
ACUCOBOL-GT Thin Client: Automatic update was
 unsuccessful
```

You can use the TC_AUTO_UPDATE_FAILED_MESSAGE configuration variable to change the text of this message.

For more information about this variable, refer to section 7.4.7, “Automatic Update Failure.”

TC_AUTO_UPDATE_FAILED_TITLE

This configuration variable applies only to the thin client automatic update feature. If this process fails, the message specified either by default or by the TC_AUTO_UPDATE_FAILED_MESSAGE configuration variable appears with the following in the title bar:

```
ACUCOBOL-GT Thin Client
```

You can use the TC_AUTO_UPDATE_FAILED_TITLE configuration variable to change this title bar text.

For more information about this variable, refer to section 7.4.7, “Automatic Update Failure.”

TC_AUTO_UPDATE_NOTIFY_FAIL

This configuration variable applies only to the thin client automatic update feature. If this process fails, a message appears informing the user of the failure. In some cases, you may not want the thin client to inform the user that the automatic update failed. If you don’t want this message to appear, set the TC_AUTO_UPDATE_NOTIFY_FAIL configuration variable to “false” (0, off, no). The default value of this variable is “true” (1, on, yes).

Note that the message box does not appear if the user cancels the Windows installer download process.

For more information about this variable, refer to section 7.4.7, “Automatic Update Failure.”

TC_AUTO_UPDATE_QUERY

This configuration variable applies only to the thin client automatic update feature. When an event triggers the update process, the thin client displays a message box informing the user that an upgrade is required. The default setting of “1” (on, true, yes) for the TC_AUTO_UPDATE_QUERY configuration variable enables the display of that message box. Setting this variable to “0” (off, false, no) prevents the message box from appearing.

For more information about this variable, refer to section 7.4.4, “Informing the User When an Update Is Needed.”

TC_AUTO_UPDATE_QUERY_MESSAGE

This configuration variable applies only to the thin client automatic update feature. When an event triggers the update process, the thin client displays a message box informing the user that an upgrade is required. The value of the TC_AUTO_UPDATE_QUERY_MESSAGE configuration variable determines the message displayed in that message box. The default value of the variable depends on the circumstances that triggered the automatic update. For example, if the automatic update is initiated by a version or protocol number mismatch, the default message displayed is the following:

```
Incompatible server version
Server version: <srvvers>, client <clntvers>
Server protocol: <srvproto>, client <clntproto>
Press OK to automatically correct this problem
```

where <*srvvers*>, <*clntvers*>, <*srvproto*>, and <*clntproto*> are replaced by the server version, client version, server protocol number, and client protocol number, respectively.

For more information about this variable, including other default values for this configuration variable, refer to section 7.4.4, “Informing the User When an Update Is Needed.”

TC_AUTO_UPDATE_QUERY_TITLE

This configuration variable applies only to the thin client automatic update feature. When an event triggers the update process, the thin client displays a message box informing the user that an upgrade is required. You use the TC_AUTO_UPDATE_QUERY_TITLE configuration variable to specify the title bar text in that message box. The default value of this variable is

```
ACUCOBOL-GT Thin Client
```

For more information about this variable, refer to section 7.4.4, “Informing the User When an Update Is Needed.”

TC_AX_EVENT_LIST

This configuration variable applies only in a thin client environment. TC_AX_EVENT_LIST lets you control which events your program receives, giving you more control over the rate of network traffic. It contains the numeric value of a single .NET or ActiveX event type or a list of .NET or ActiveX event types separated by non-numeric characters like spaces or commas. Whether your program receives these events depends on the value of TC_EXCLUDE_EVENT_LIST. If its value is “0”, your program receives the events listed in TC_AX_EVENT_LIST. If the value of TC_EXCLUDE_EVENT_LIST is “1”, the events listed in TC_AX_EVENT_LIST are not sent to your program.

Note that this variable must be set in the configuration file and cannot be changed programmatically via the SET statement.

An AX-EVENT-LIST common control property performs the same function as this configuration variable. For more information, refer to section 8.8.2.5, “Graphical control event handling.”

TC_CHECK_ALIVE_INTERVAL

This variable applies only in a thin client environment. Use it to specify how long the runtime should wait (in seconds) for an inactive client (a “dead” **acuthin**) before exiting the server runtime process. If you enter a value of

"60", the runtime checks for client activity for 60 seconds. Client activity includes regular client user interaction or "ping" messages sent automatically from the client to the server. If no client activity is detected during the specified interval, the server runtime process exits. Valid values range from 1 to 32767. Set this variable to "0" to disable client checking. The default value is "300" (5 minutes).

TC_CHECK_INSTALLER_TIMESTAMP

This configuration variable applies only to the thin client automatic update feature. The value of the TC_CHECK_INSTALLER_TIMESTAMP configuration variable determines whether the thin client compares the modification times of the installer files on the client and on the server. If this variable is set to "1" (on, true, yes) and the modification time of the client file is older than the time of the server file, the automatic update process is initiated. If the installer file does not exist on the client, the comparison is made with the modification time of the thin client executable (**acuthin**) currently running. The default value for this variable is "0" (off, false, no).

For more information about this variable, refer to section 7.4.1, "Automatic Update Overview."

TC_CONTINUITY_WINDOW

This variable applies only in a thin client environment. TC_CONTINUITY_WINDOW can ensure that a given application does not lose focus in Windows 2000 and Windows XP clients because the application has destroyed all of its windows. Setting TC_CONTINUITY_WINDOW to "1" (on, true, yes) causes the thin client to create an invisible, independent window, which ensures that your application maintains focus. The default value is "0".

TC_CONTROL_SYNC_LEVEL

This variable applies only in a thin client environment. TC_CONTROL_SYNC_LEVEL lets you control which VALUE data items are updated after an embedded procedure is executed. The setting of this variable can affect performance.

Setting	**Effect**
1 (the default)	Only the VALUE data item associated with the current field is updated when its AFTER or EXCEPTION procedure executes.
2	Only the VALUE data item associated with the current field is updated when its BEFORE, AFTER, or EXCEPTION procedure executes.
3	All VALUE data items are updated when any BEFORE, AFTER, or EXCEPTION procedure executes. Note that this setting affects only BEFORE, AFTER, and EXCEPTION procedures. Values of all variables are made current any time an ACCEPT terminates

For best performance, we recommend that this variable be set to the default of “1”. You can directly INQUIRE the value of a control in an embedded procedure if necessary.

TC_DELAY_ACTIVATE

This variable applies only in a thin client environment. TC_DELAY_ACTIVATE lets you control whether the thin client delays sending a CMD-ACTIVATE event to the server until after the Windows notification that caused the event is complete. Note that ActiveX events are never delayed, regardless of the setting of this variable. The default value of “1” (on, true, yes) enables this behavior. Setting TC_DELAY_ACTIVATE to “0” (off, false, no) turns off the delay mechanism.

TC_DELAY_PRE_EVENT_OPS

This variable applies only in a thin client environment. Using this configuration variable, you can direct the thin client to buffer some requests received from the server and process them later. When you set this variable to “1”, the thin client buffers the requests received between the time that the client sends an event to the server and the time that the server informs the client that it has started the related event procedure. The events are processed only after the event procedure starts to prevent the thin client from processing requests that generate more events before the first event procedure has started. The default value of TC_DELAY_PRE_EVENT_OPS is “0”.

Note: The buffering behavior described for this configuration variable was introduced as the default behavior in Version 6.1. Beginning with Version 7.2, the buffering behavior is turned off by default.

TC_DISABLE_AUTO_UPDATE

This configuration variable applies only to the thin client automatic update feature. You can disable the automatic update process by setting the TC_DISABLE_AUTO_UPDATE configuration variable to “1” (on, true, yes). The default value of this variable is “0” (off, false, no).

For more information about this variable, refer to section 7.4.3, “Enabling or Disabling the Automatic Update Feature.”

TC_DISABLE_SERVER_LOG

This configuration variable applies only to the thin client automatic update feature. If this process fails, a log file may be created on the server. This file contains a log of the update operations and details about the failure. To prevent the creation of this log file, set the TC_DISABLE_SERVER_LOG configuration variable to “true” (1, on, yes). The default value of this variable is “false” (0, off, no).

For more information about this variable, refer to section 7.4.7, “Automatic Update Failure.”

TC_DOWNLOAD_CANCEL_MESSAGE

This configuration variable applies only to the thin client automatic update feature. During the automatic update installer file download process, a progress dialog box appears. You can cancel the download at any time from this dialog. Use the TC_DOWNLOAD_CANCEL_MESSAGE configuration variable to specify the message that appears when the download is cancelled. The default value for this variable is

```
Please wait while the download is being cancelled . . .
```

For more information about this variable, refer to section 7.4.5.4, “Download progress dialog.”

TC_DOWNLOAD_DESCRIPTION

This configuration variable applies only to the thin client automatic update feature. During the automatic update installer file download process, a progress dialog box appears. You use the TC_DOWNLOAD_DESCRIPTION configuration variable to specify the text that appears in the middle of the download progress dialog. Its default value is

```
Downloading installation file. . .
```

For more information about this variable, refer to section 7.4.5.4, “Download progress dialog.”

TC_DOWNLOAD_DIALOG

This configuration variable applies only to the thin client automatic update feature. During the automatic update installer file download process, a progress dialog appears. The default value of “1” (on, true, yes) for the TC_DOWNLOAD_DIALOG configuration variable allows the appearance of this dialog box. If you set this variable to “0” (off, false, no), the progress dialog does not appear.

For more information about this variable, refer to section 7.4.5.4, “Download progress dialog.”

TC_DOWNLOAD_DIALOG_TITLE

This configuration variable applies only to the thin client automatic update feature. During the automatic update installer file download process, a progress dialog appears. The TC_DOWNLOAD_DIALOG_TITLE configuration variable is used to specify the title bar text in this dialog. The default value of this variable is

```
ACUCOBOL-GT Thin Client Automatic Update
```

For more information about this variable, refer to section 7.4.5.4, "Download progress dialog."

TC_EVENT_LIST

This variable applies only in a thin client environment. TC_EVENT_LIST lets you control which events your program receives, giving you more control over the rate of network traffic. It contains the numeric value of a single event type or a list of event types separated by non-numeric characters like spaces or commas. Whether your program receives these events depends on the value of TC_EXCLUDE_EVENT_LIST. If its value is "0", your program receives the events listed in TC_EVENT_LIST. If TC_EXCLUDE_EVENT_LIST is "1", the events listed in TC_EVENT_LIST are not sent to your program. Note that this variable must be set in the configuration file and cannot be changed programmatically via the SET statement.

An EVENT-LIST common control property performs the same function as this configuration variable. For more information, refer to section 8.8.2.5, "Graphical control event handling."

TC_EXCLUDE_EVENT_LIST

This variable applies only in a thin client environment. The value of TC_EXCLUDE_EVENT_LIST determines whether the events listed in TC_AX_EVENT_LIST or TC_EVENT_LIST are sent to your program. A value of “1” means the specified events are not sent to your program. The default value is “0”. Note that this variable must be set in the configuration file and cannot be changed programmatically via the SET statement.

An EXCLUDE-EVENT-LIST common control property performs the same function as this configuration variable. For more information, refer to section 8.8.2.5, “Graphical control event handling.”

TC_INSTALLER_ARGS

This configuration variable applies only to the thin client automatic update feature. The thin client uses the value of the TC_INSTALLER_ARGS configuration variable as the command-line options passed to the installer executable. For example, if you want “msiexec.exe” to log all of its operations to a file named “msi.log”, then you could set TC_INSTALLER_ARGS to /log msi.log. TC_INSTALLER_ARGS has no default value.

For more information about this variable, refer to section 7.4.5, “Accepting the Automatic Update.”

TC_INSTALLER_CLIENT_FILE

This configuration variable applies only to the thin client automatic update feature. You use the TC_INSTALLER_CLIENT_FILE configuration variable to specify the path and file name of the installer file that you want to create on the client. The default value of this variable is

```
<APPDATA>\ACUCOBOL-GT\<installer_server_filename>
```

where <*APPDATA*> is a special directory name for C:\Documents and Settings\<user>\Application Data.

For more information about this variable and special directory names like <APPDATA>, refer to section 7.4.5, “Accepting the Automatic Update.”

Note: If TC_INSTALLER_CLIENT_FILE is set to a non-existent directory or to one on which the autoupdate process doesn't have sufficient permissions, no log file is written if the autoupdate process fails.

TC_INSTALLER_RUN_ASYNC

This configuration variable applies only to the thin client automatic update feature. You use the TC_INSTALLER_RUN_ASYNC configuration variable when you want to prevent the thin client from restarting after an automatic update or when your installer file handles the automatic update process to completion. When you set this variable to "1" (on, true, yes), the thin client starts the installer process asynchronously and then exits immediately. It does not wait for the automatic update process to complete and does not restart the application. The default value is "0" (off, false, no).

For more information about this variable, refer to section 7.4.5, "Accepting the Automatic Update."

TC_INSTALLER_SERVER_FILE

This configuration variable applies only to the thin client automatic update feature. You set the TC_INSTALLER_SERVER_FILE configuration variable to the path and file name of the server installer file. Its default value is

```
    <runtime_path>/acuthin.msi
```

where <*runtime_path*> is the directory that contains the **wrun32** or **runcbl** runtime executable.

For more information about this variable, refer to section 7.4.5, "Accepting the Automatic Update."

TC_INSTALLER_TARGET_DIR

This configuration variable applies only to the thin client automatic update feature. You use the TC_INSTALLER_TARGET_DIR configuration variable to specify the location where you want the updated thin client to be installed. This variable has no default value.

For more information about this variable, refer to section 7.4.5, "Accepting the Automatic Update."

TC_INSTALLER_UI_LEVEL

This configuration variable applies only to the thin client automatic update feature. The keywords or numeric values in the TC_INSTALLER_UI_LEVEL configuration variable control the Windows installer interface. Set TC_INSTALLER_UI_LEVEL to NONE or "0" if you do not want the Windows installer to display a user interface. Set this variable to UNATTENDED or "1" if you want the Windows installer to display informational and progress messages, but to execute unattended. Set the variable to INTERACTIVE, DEFAULT, or "2" if you want the Windows installer to prompt for and accept user input for the installation process. Set the variable to REDUCED or "3" if you want to use a reduced user interface.

For more information about this variable, refer to section 7.4.5, "Accepting the Automatic Update."

TC_MAP_FILE

This variable applies only in a thin client environment. Use this variable to define the name and path of the map file to be used (if any) to map special characters in your client character set to their decimal or hexadecimal equivalent on the server. This file lets you reconcile the character encoding between two machines that use different codes for the same characters and is particularly useful for international character translation. (See section 4.5, "International Character Handling," for more information.) Note that only single-byte alphanumeric characters are mapped.

TC_NESTED_AX_EVENTS

This variable applies only in a thin client environment. TC_NESTED_AX_EVENTS allows you to control whether the runtime processes subsequent ActiveX events while it is already processing an ActiveX event from the same control. Setting this variable to “1” enables this behavior. The default value is “0”.

TC_QUIT_MODE

This variable applies only in a thin client environment. TC_QUIT_MODE lets you control how your COBOL application shuts down when no client activity occurs during the interval defined by TC_CHECK_ALIVE_INTERVAL. Setting TC_QUIT_MODE to “-1” (the default value) shuts your program down according to the value chosen for the QUIT_MODE configuration variable. (Refer to *ACUCOBOL-GT Appendices*, Appendix H, for details.) If you set this variable to “0”, the runtime stops the program immediately. When this variable is set to a value greater than “0” (up to “32767”), your application has a program-controlled exit.

When the runtime determines that the thin client is no longer responding (no user interaction and no pings during TC_CHECK_ALIVE_INTERVAL), the MSG-MENU-INPUT event is sent to the program’s main window and EVENT-DATA-2 contains the value defined by TC_QUIT_MODE. Your program can detect this in the main window’s event procedure and you can perform whatever code you desire. At this point, there is no connection to the thin client, so user interface operations may not be performed. You must end your shutdown code with “STOP RUN” to terminate the runtime.

TC_REQUIRES_BUILD_NUMBER

This configuration variable applies only to the thin client automatic update feature. When the thin client executes, it compares its build number with the value of the TC_REQUIRES_BUILD_NUMBER configuration variable. If the value of this variable does not match the client’s build number, the automatic update process is initiated. Set this variable to the thin client build number required by the application. The default value of this variable is “0” (off, false, no).

For more information about this variable, refer to section 7.4.1, “Automatic Update Overview.”

TC_RESTRICT_AX_EVENTS

This variable applies only in a thin client environment. The thin client runtime suspends all ActiveX events when the application is not processing an ACCEPT statement. However, some ActiveX controls do not support the ability to suspend and resume events. As a result, in a thin client environment, an event procedure may run unexpectedly during a CREATE, DISPLAY, MODIFY, INQUIRE, or any other operation that waits for results from the thin client.

With the TC_RESTRICT_AX_EVENTS configuration variable, you can control whether your application ignores all ActiveX events between the termination of one ACCEPT statement and the beginning of another. Setting this variable to “1” (on, true, yes) enables this behavior. The default value for this variable is “0” (off, false, no).

To determine if a particular ActiveX control supports suspending and resuming events, check the control’s documentation or ask the control vendor. Note that to support suspending and resuming events, the control must implement the “IOleControl::FreezeEvents()” method.

TC_SERVER_LOG_FILE

This configuration variable applies only to the thin client automatic update feature. If the automatic update process fails for any reason, a log file may be created on the server. This file contains a log of the update operations and details about the failure. The TC_SERVER_LOG_FILE configuration variable can be used to configure the location and name of that log file. The name can optionally include the hostname of the client machine and the process ID of the server runtime that was managing the automatic update at the time of the failure.

By default, this file is named “autoupdate.%c.%p.log”, where *%c* is replaced by the client hostname and *%p* is replaced by the process ID of the server runtime. The default location is the working directory specified in the alias on the server. Note that the directory must exist at the time of the failure for the log file to be created.

For more information about this variable, refer to section 7.4.7, “Automatic Update Failure.”

TC_SERVER_TIMEOUT

This variable applies only in a thin client environment. TC_SERVER_TIMEOUT lets you determine how many seconds (from 0 to 32767) the client waits for a response from the server. If the client receives no response from the server in the specified time period, the following message box appears:

```
The remote host is not responding.
Press OK to close this program.
Press Cancel to wait another %s seconds.
```

where *%s* is the value of TC_SERVER_TIMEOUT. The default value is “20”.

TC_TV_SELCHANGING

This variable applies only in a thin client environment. The TC_TV_SELCHANGING variable lets you control how often Msg-Tv-Selchanging events are generated in the thin client environment. A setting of “0” means that the Msg-Tv-Selchanging event is never generated. Setting this variable to “1” (the default) results in generation of a Msg-Tv-Selchanging event only when a tree view selection is changed. A setting of “2” means that Msg-Tv-Selchanging is always generated. For more information on Msg-Tv-Selchanging, see the ACUCOBOL-GT *User’s Guide*, section 6.3, “Control Events.”

THIN_CLIENT_ENCRYPT

This variable applies only in a thin client environment. Use of this variable is superseded by the AGS_SOCKET_ENCRYPT configuration variable. Setting THIN_CLIENT_ENCRYPT to “1” means that AGS_SOCKET_ENCRYPT is automatically set to “1”. Refer to information about the AGS_SOCKET_ENCRYPT variable if you want to encrypt your data.

4 Preparing Your Application

Key Topics

4.1 Designing Your Application

AcuConnect® is a client/server technology that lets you distribute your computing processes in the way that best suits your business needs. How you design your application for use with AcuConnect can affect system performance and efficiency. Whether you choose the distributed processing model or use our thin client technology with AcuConnect, you will want to give some serious thought to your application design. Some issues for consideration are covered in section 4.2, "Distribution Considerations."

4.1.1 In Distributed Processing

To prepare your application for use in a distributed processing environment, you need to divide the application into two main components: a client component and a server component. If you are developing a new application, you can consider the division during the design stage of development. If you are modifying an existing application for use in a distributed processing environment, you need to find a logical division point.

With distributed processing, we recommend that your client application contain not only the user interface portion of the program, but also all of the interactive components. The server application, on the other hand, should be reserved for batch processing components (components that do not require user interaction).

How much processing is performed on the client and how much on the server is totally up to you. With AcuConnect's distributed processing capabilities, you can distribute the workload for maximum throughput. Be sure to consider such issues as network I/O, performance, and security when designing your application distribution.

If you change the location of your server application, all you need to do is change the CODE_PREFIX setting or the code alias definition in the client configuration file (for example, "client.cfg"). If you want to be able to access your application on multiple machines, you can install the server component in several locations, and then modify "client.cfg" on an as-needed basis. More information regarding the design of a distributed processing application can be found in section 4.3, "Distributed Processing Application Design."

4.1.2 With Thin Client Technology

Applications for use in a thin client environment require some preparation as well, even if you already have an ACUCOBOL-GT® program. Remember that with our thin client technology, only the user interface portion of your application resides on a Windows client or display host. Thin client supports all ACUCOBOL-GT control types and classical ACCEPT and DISPLAY. It also allows the use of ActiveX controls, provided the controls are installed on the client machine.

If your ACUCOBOL-GT program is already graphical, full benefits are immediately available in thin client. If your ACUCOBOL-GT program is character-based, you can run it as is and convert to graphical over time using AcuBench®, ACUCOBOL-GT, or the Character-to-GUI Wizard.

If you plan to redeploy Windows applications in a thin client environment, you should be aware that your application will be running on a back-end server rather than the desktop. The client machine provides screen and printing services. Other operating system services are provided by the server. If your program operates on the Windows registry, and the server is not a Windows machine, then the registry does not exist. Calls to those library routines return a “not supported” status. Information about thin client application design issues can be found in section 4.4, “Thin Client Application Design.”

4.2 Distribution Considerations

AcuConnect allows you to distribute your application components to best use your enterprise resources. You can perform most of your processing on the client (in a “thick” client arrangement), most on the server (in a “thin” client arrangement), or divide the processing equally. The ability to choose between these options or select anything in between makes the AcuConnect client a “smart” client.

Like any paradigm, there are advantages and disadvantages to the thin and thick client approaches. Deciding where to distribute your processing tasks requires extensive knowledge of the network. Your decision should be based on several issues, including:

- Security—weighing ease of use against system integrity.
- Performance—scaling client speed versus the speed of the server.
- Physical location of program and data files, factoring in two-tier versus three-tier applications.
- Number of users—planning for availability versus contention.
- Network bandwidth—intranet versus Internet, dial-up versus direct access.
- User requirements—occasional use versus constant connection, and batch processing versus real-time processing.

4.2.1 When to Use a “Thin” Client

For performance reasons, the “thin” client approach is ideal for data-intensive applications such as reporting. Data-intensive applications typically have a small number of screen I/Os and ACCEPTs, so they can easily be executed on the server to reduce network traffic. The client in this case is “thin”, because the server performs the bulk of the application processing.

Our Thin Client technology can be an efficient architecture, with only the user interface portion of your application residing on the client. This model can also be a good choice if you want to develop a graphical user interface for your non-Windows application. In this case, the user interface displays on a Windows machine, while the rest of your application resides on a UNIX, Linux, Windows, or VMS machine.

4.2.2 When to Use a “Thick” Client

The “thick” client approach is suitable for screen-intensive applications with interactive data entry. Such applications would suffer performance degradation if processing were performed on a server and passed across the network. By keeping interactive applications on the client, network traffic can be kept at a minimum.

You might also consider this arrangement if you want your client application to interact with other desktop software using the client’s processor.

4.2.3 When to Use a “Smart” Client

In reality, most applications have a combination of screen and data I/O. To maximize performance, you can divide your application into a client component and a server component, splitting the processing responsibilities between the client and the server where you see fit. The “smart” client approach is the foundation of the AcuConnect deployment. AcuConnect lets programmers or systems analysts design enterprise applications so that they use the least amount of network resources and at the same time reduce response time.

4.3 Distributed Processing Application Design

Preparing your application for use in a distributed processing environment involves the consideration of several issues. Use of the CALL verb, synchronous or asynchronous program execution, and various memory and environment issues should be explored. The following sections describe these areas of concern.

4.3.1 Embedding COBOL CALLs

AcuConnect in distributed processing achieves remote application access with standard COBOL CALLs. Just as you would CALL a local application component, you can CALL a remote application component using AcuConnect.

For example, if the name of the remote program that you want to invoke is “prog2.acu”, you embed the following statement into your client application:

```
CALL "prog2".
```

If you have not set the ACUCOBOL-GT configuration variable CODE_SUFFIX, the runtime system first attempts to locate an object file called “prog2.acu” and then an object file called “prog2” with no suffix.

Or, if you have added “CODE_SUFFIX obj” to the “client.cfg” file, the runtime looks only for an object file called “prog2.obj” on the remote machine.

AcuConnect supports ACUCOBOL-GT's standard usage of the CALL verb, so you could embed:

```
CALL "prog2" USING {parameter} . . .
CALL "prog2" ON {EXCEPTION} statement-1.
```

Note: You can use all forms of CALL documented in the *ACUCOBOL-GT Reference Manual* with AcuConnect, with the following exception: AcuConnect does not support the use of CALL THREAD, CALL PROGRAM, or CALL RUN to a remote server.

The configuration file on the client, "client.cfg", specifies the complete remote path of the CALLed program, so you don't have to specify the path in the application code. This way, you can redistribute the application in your network as often as necessary without changing a single line in your code.

4.3.1.1 Terminating the remote application

By default, AcuConnect leaves the server runtime in memory with an open connection until the client application exits. This allows you to CALL the application as often as necessary without having to restart the server runtime.

If you want to close the connection after a CALL is completed, you must embed a CANCEL or CANCEL ALL in your client program and set the ACUCONNECT_CLOSE_AFTER_CANCEL configuration variable in the "client.cfg" file. Set to "0", this variable leaves the remote application open until the client application exits. Set to "1", this variable closes the remote application whenever a CANCEL occurs in the program. Note that you cannot set ACUCONNECT_CLOSE_AFTER_CANCEL from within a program.

Note: If you make asynchronous CALLs using the C$ASYNCRUN routine described in section 4.3.2.1, "CALLing C$ASYNCRUN," you do not need to embed a CANCEL in your client program. The routine used to check application status, C$ASYNCPOLL, closes the remote application automatically when it receives a "completed" status indicator.

4.3.1.2 Exception handling

As is the case with any CALL, if your remotely CALLed program encounters an exception, the program terminates unless you use the “on exception” phrase. You could then use the C$CALLERR routine to find out what went wrong. If an ERR-CODE of “25” is returned, it means that the AcuConnect server is not running.

For more information on the “on exception” phrase or the C$CALLERR library routine, refer to the ACUCOBOL-GT documentation set.

4.3.1.3 CALLing multiple programs

With AcuConnect in distributed processing, you can start multiple applications on the server at the same time by invoking multiple instances of the runtime.

In addition, programs that have been CALLed using AcuConnect can CALL other programs on the same server as normal, or on a different server using AcuConnect. For example, program 1 on a client machine can CALL program 2 on Server A. In turn, program 2 can CALL program 3 on Server B, and program 3 could even CALL program 4 on Server C, and so on.

To start multiple programs in this fashion, you must design your application components to perform remote CALLs, then provide the necessary configuration files.

When CALLing multiple programs, be aware of the following:

- If a “second generation” AcuConnect program (such as program 3 on Server B in the example above) requires a password from the first generation program, the password must be supplied using the *acu_client_password* variable coded into the CALLing program. Otherwise, the authentication will fail, because the first generation AcuConnect program has no way to handle an interactive password screen. For more information on using passwords with AcuConnect, see section 2.7.1, “Passwords.”
- Each successive generation of AcuConnect programs can have its configuration file and runtime flags specified by its CALLing program using the ACUCONNECT_RUNTIME_FLAGS variable.

- Runtime configuration variables are not passed from one program to the next when you use AcuConnect. So, instead of creating code name aliases for all the programs in the client configuration file, you should provide only the variables relevant to the first generation programs you are CALLing. The first generation programs need their own configuration variables to be used in CALLing second generation programs, and so forth.

4.3.2 Synchronous or Asynchronous Execution

By default, AcuConnect performs synchronous CALLs to remote applications. That is, the client application CALLs the remote application, then waits for a response back from the server. Only when the client receives a response from the server does the client application continue processing.

To perform asynchronous CALLs, you can CALL the C$ASYNCRUN library routine along with the remote application name. This routine is installed during normal AcuConnect installation.

4.3.2.1 CALLing C$ASYNCRUN

If desired, you call the C$ASYNCRUN library routine along with the remote application name to achieve asynchronous processing. The syntax for this CALL is:

```
CALL "C$ASYNCRUN" using handle-of-call program_name parameter-1
parameter-2
```

where:

handle-of-call	is a handle of the CALL defined in Working-Storage
program_name	is the name of the CALLed program
parameter	is a parameter of the CALL

For example, you might have the following line in the Working-Storage section of your client program:

```
01 h-call-prog2     handle.
```

and the following in the Procedure Division section:

```
     CALL "C$ASYNCRUN" using h-call-prog2 "prog2.acu" customer-info.
```

C$ASYNCRUN tells AcuConnect to allow the client application to continue processing even while the server application is active.

4.3.2.2 CALLing C$ASYNCPOLL

To check the status of the server program while the client is running, you can call the C$ASYNCPOLL routine. The syntax for this CALL is:

```
   CALL "C$ASYNCPOLL" using handle-of-call state-of-call parameter-1
   parameter-2.
```

where:

handle-of-call	is the handle of the CALL previously run with C$ASYNCRUN
state-of-call	is a PIC s(9) with value “0” if the run is not yet completed, or “1” if the run is completed
parameter	is the parameter of the CALL returned when state-of-call is “1”

For example:

```
     CALL "C$ASYNCPOLL" using h-call-prog2 state-of-call
      customer-info.
```

Note that any parameters given in C$ASYNCPOLL must match the parameters given in C$ASYNCRUN. C$ASYNCPOLL tells AcuConnect to query the server about the status of the remote application. AcuConnect returns a status that you can DISPLAY on the client. If the status is “1” (CALL completed), C$ASYNCPOLL terminates the connection with the remote application.

4.3.3 Memory and Environment Issues

As you design your application for distributed processing, you should be aware of certain memory/environment issues with the ACUCOBOL-GT runtime.

Runtimes

Remotely CALLed programs cause the AcuConnect server to start a new runtime on the server machine. Spawned runtimes can start runtimes on other server machines, if desired, or they can call other programs on the same server.

AcuConnect can spawn Acu4GL® or AcuServer® linked runtimes, even if the client runtime is not so linked.

Memory

The remote program and remote runtime live in memory on the server machine.

Environment

The runtime started by AcuConnect on the server takes on whatever environment that the AcuConnect daemon itself was started in when it was started on the server. For example, if you log on as root and start AcuConnect, AcuConnect will inherit root's environment. If you log on as userX and then start AcuConnect by issuing the superuser account, AcuConnect will inherit userX's environment.

The important thing to note is that the CALLed program does not inherit the client environment of the CALLing program.

Open connections

The runtime started by AcuConnect on the server in response to a CALL is kept running until a "stop run" is encountered on the client. This has the effect of maintaining any items in memory on the server machine until the client program is shut down.

One exception to this is in the case of asynchronous program execution, when the C$ASYNCPOLL library routine is called. In this case, the runtime on the server is immediately shut down when a status of "1" (CALL completed) is returned by C$ASYNCPOLL.

Another exception is in response to a CANCEL command (see section 4.3.1.1, “Terminating the remote application,” for more information).

EXTERNAL data items

Data items declared as EXTERNAL are not shared between client and remote programs with AcuConnect.

CHAIN command

The CHAIN command cannot be used to start a remote program through AcuConnect.

4.4 Thin Client Application Design

Designing an application for our Thin Client environment presents some unique issues. In addition to user interface design concerns and certain issues with Windows and printing, various technical limitations apply in this model. The following sections cover these considerations.

Note that if your application is currently in a language other than ACUCOBOL-GT, you must migrate your code to ACUCOBOL-GT before you can implement our Thin Client solution. You can perform the migration yourself, or hire a Micro Focus ***extend*** consultant.

No special preparation is needed for your data in a thin client environment. Because your data resides on a server (either the same server as the application or on a different data server) access to that data is the same as it would be in a standard client/server environment using AcuServer or Acu4GL. The server runtime passes the results back to the ACUCOBOL-GT Thin Client for display. If the data resides on the same server as the application runtime, no extra communication software is required.

4.4.1 Limitations in Thin Client Environments

Note that there are some limitations to running a program with thin client. You may need to make some adjustments to your program if you rely on these features:

- The SCREEN EDITED-UPDATES=Formatted runtime configuration variable is not supported. Generally speaking, you can continue to run programs that use this feature without changing them. The user will experience a formatting difference while entering data, but the final result should be the same.
- Creating a graphical component does not move the cursor to the cell to the component's right. This restriction does not affect relative graphical component placement in the Screen Section (for example, "col + 2").
- The grid control does not generate the MSG-CELL-GOTO-DRAG event. This is done for performance reasons. Because of this, you cannot use the REGION-COLOR property to highlight an area when the mouse is dragged over it. For a description of an alternate technique for doing this, refer to section 8.8.2.7, "Grid control."
- The runtime assumes that graphical components and windows are successfully created in the thin client. If the client runs out of memory, the creation succeeds, but subsequent operation is undefined.
- The CREATE statement can only create COM and .NET objects on Windows-based servers. Refer to the CREATE statement documentation in the *ACUCOBOL-GT Reference* manual for details.
- The JUSTIFY_NUM_FIELDS configuration variable is not communicated to the thin client.
- The C$SETVARIANT library routine and OLE SAFEARRAY data type are supported in thin client environments, regardless of the kind of operating system on the server. The C$GETVARIANT library routine, however, is not supported and returns a negative RESULT-CODE value in thin client environments. To learn more about the C$SETVARIANT routine, refer to Appendix I in ACUCOBOL-GT Appendices.

Note: You should be careful when using C$SETVARIANT in thin client environments, because it generates network traffic and can affect performance. When using this library routine in a thin client environment, you should pass only small amounts of data.

- The WIN32_CTL_INPUT_STATUS configuration variable, which affects ACCEPT FROM INPUT STATUS behavior, is not supported in a thin client environment.
- The thin client environment provides limited support for the SHARED_LIBRARY_LIST configuration variable and the “-y” runtime option. Note that they do not load client-side dynamic link libraries (DLLs) for thin client applications that make calls using the CALL verb “@[DISPLAY]:” syntax. These applications must explicitly load the DLL by calling it with the CALL verb before calling a function within the DLL. For more information about SHARED_LIBRARY_LIST, refer to Appendix H in *ACUCOBOL-GT Appendices*. To learn more about the “-y” option, see Chapter 2 in *ACUCOBOL-GT User’s Guide*.

4.4.2 User Interface Work in Thin Client

Front-end work is necessary in the following situations:

- When you want a graphical display and haven’t built one yet
- When the limitations of the character display are unacceptable
- When your program is highly interactive and you want to deploy it in a wide-area network like the Internet

4.4.2.1 Building a new graphical display

If you want your application to display a full Windows graphical user interface (GUI) on the client, but it is currently character-based, you can develop a GUI using our graphical workbench, AcuBench. ACUCOBOL-GT includes a Character-to-GUI Wizard to assist in this process, and the workbench is designed to automatically use many of the graphical features of ACUCOBOL-GT.

4.4.2.2 Working with character display limitations

Because of certain limitations and restrictions, if you deploy a character-based application in our Thin Client configuration, certain display functions will not have the characteristics that you expect. Many will simply display in another fashion. But some unsupported display functions may be ignored. If this is unacceptable to you, you can alter your code to work around the limitations.

4.4.2.3 Deploying a highly interactive program in a wide-area network

Because highly interactive programs require frequent communication between the client and server, they are not practical for thin client configurations deployed over wide-area networks like the Internet. The more interactive the program, the slower the performance is over slow connections.

If your program was coded with event procedures (handlers), for instance, you should consider removing all but the events that your program really cares about, because event procedures result in frequent "calls" from the UI layer back to the COBOL layer. If your program tracks mouse movements or watches for interactions with controls, you may want to make further modifications. You can choose to redesign the screen or you can just simplify your program's interaction with the screen. The goal is to reduce the number of times that the client and server have to communicate.

4.4.3 Other Application Work in Thin Client

We have identified some other situations that require modifications to an application for use in a thin client environment. Issues involving certain Windows features and printing, among others, are described in the following sections.

4.4.3.1 Adjusting for certain Windows features

You may want to adjust your application for the individual Windows characteristics described in the following sections.

Making direct calls to the Windows API

Windows applications that contain direct calls to the Windows API must be modified to remove these calls, whether you deploy on a Windows or UNIX server.

Deploying a Windows application on UNIX

ACUCOBOL-GT is so portable that you can deploy your Windows application on a UNIX server, if you wish. As mentioned earlier, if your application calls Windows API functions directly, you will need to remove these calls. If it relies on DOS file naming conventions and directory structures, you will need to modify your program with UNIX file names and directories.

Testing the OS-IS-WINDOWS flag

Some programs that are intended to run under multiple operating systems may test the OS-IS-WINDOWS flag to determine whether a graphical display is available. If your server is not a Windows machine, these programs assume that graphics are not available in thin client. Programs like this should test the HAS-GRAPHICAL-INTERFACE flag rather than OS-IS-WINDOWS. More information can be found in the descriptions of ACCEPT FROM SYSTEM-INFO and ACCEPT FROM TERMINAL-INFO in section 6.6 of the *ACUCOBOL-GT Reference Manual*.

Accessing Windows help from a UNIX application

Your UNIX applications can access Windows help via the ACUCOBOL-GT $WINHELP library routine. Calling $WINHELP from your COBOL program sends the given arguments to the thin client to process. The thin client eventually calls the Windows help program with the arguments given to $WINHELP. Note that the help file is not copied to the client machine, but the client must be able to find this file. In a local office environment, help files can be installed on a local hard drive or a shared drive. Remote users should install the help files on the client machine. (More information about $WINHELP can be found in Appendix I in *ACUCOBOL-GT Appendices*.)

Using certain $WINHELP op-codes

Some $WINHELP op-codes are not supported in a thin client environment. If your application uses them, you may experience unexpected system terminations. These op-codes are HELP_MULTIKEY, HELP_SETWINPOS, HELP_CONTEXTMENU, HELP_SETPOPUP_POS, and HELP_WM_HELP.

4.4.3.2 Accessing the Windows registry on the client machine

A set of library routines lets you access and modify the Windows registry on the client machine. The names of the routines all begin with "DISPLAY" to distinguish them from similar routines that operate on the server's registry if the server is Windows. For example, you use DISPLAY_REG_CREATE_KEY or DISPLAY_REG_CREATE_KEY_EX to create a new registry key on the display host. Note that user authorization is required to change the registry with any of the following routines:

- DISPLAY_REG_CREATE_KEY
- DISPLAY_REG_DELETE_KEY
- DISPLAY_REG_SET_VALUE
- DISPLAY_REG_CREATE_KEY_EX
- DISPLAY_REG_DELETE_VALUE
- DISPLAY_REG_SET_VALUE_EX

You authorize or cancel an operation in an authorization dialog.

Refer to *ACUCOBOL-GT Appendices*, Appendix I, for a description of all the DISPLAY Windows registry library routines. Look for the section titled "Routines to Handle the Windows Registry."

4.4.3.3 Printing in thin client

Printing in a thin client environment presents the unique issues described in the following sections. You may need to modify your application accordingly.

Printing locally on Windows machines

If you'd like to allow end users to print locally on their Windows machines, you can add calls to ACUCOBOL-GT's WIN$PRINTER library routine to your program. The thin client environment supports WIN$PRINTER function calls, -P SPOOLER (DIRECT) and -Q printing. (Refer to the *Getting Started* manual for more information.)

Note that the following WIN$PRINTER functions are not supported by thin client because of variations in memory allocated by data types on the host and the client:

- WINPRINT_GET_SETTINGS_SIZE
- WINPRINT_GET_SETTINGS
- WINPRINT_SET_SETTINGS

Instead of using these functions, you can make calls to WINPRINT_GET_PRINTER_INFO_EX and WINPRINT_SET_PRINTER, which are very similar. (Refer to Appendix I in *ACUCOBOL-GT Appendices* for more information about these WIN$PRINTER functions.)

Note that if the UNIX server is not an active thin client host when a WIN$PRINTER call is made, WIN$PRINTER returns a "0" (not supported).

Any COBOL logic that disables the WIN$PRINTER capability in non-32-bit Windows environments should be removed to allow Windows client printing to work.

Printing to -P SPOOLER automatically routes the print job to the client printer.

Although it is possible to print directly to the server printer, font selection and formatting capabilities are limited. These restrictions can have an impact on form printing. We recommend that you send the print job to the client and let the client send the job to the desired printer. Note that the desired printer must be visible to the client.

4.4.3.4 Selecting a file from the client machine's drives

A call to the C$OPENSAVEBOX library routine allows you to browse the client machine's drives and select a directory or file. This routine is used in conjunction with other operations that have access to the client machine's drives, such as C$COPY, C$SYSTEM, DLL calls, and COM or ActiveX components. The server machine's file system does not appear in the box.

If your server machine is Windows, you can navigate to the server using Universal Naming Convention (UNC) notation. Set the C$OPENSAVBOX parameter OPNSAV-DEFAULT-DIR to the desired mapped drive or server directory using UNC notation.

4.4.3.5 Using W$BITMAP print screen features

Some elements of the W$BITMAP print screen feature deserve special mention in the context of a thin client deployment, particularly the WBITMAP-CAPTURE-IMAGE, WBITMAP-CAPTURE-DESKTOP, and WBITMAP-CAPTURE-CLIPBOARD functions. Full details regarding all W$BITMAP features can be found in Appendix I in *ACUCOBOL-GT Appendices*.

WBITMAP-CAPTURE-IMAGE usage is as follows:

```
CALL "W$BITMAP"
     USING WBITMAP-CAPTURE-IMAGE
     filename
     [window-handle]
     [client]
     [colordepth]
     GIVING [result].
```

If *filename* is specified, the image is stored in the named file on the server, not the client machine. For example, with the following code:

```
CALL "W$BITMAP" USING WBITMAP-CAPTURE-IMAGE
 "c:\myfile.bmp".
```

the bitmap image is stored as “myfile.bmp” in the c:\ directory on the server. In this example:

```
CALL "W$BITMAP" USING WBITMAP-CAPTURE-IMAGE
 "myfile.bmp".
```

“myfile.bmp” is stored in the working directory specified in the alias file. If you want the file stored on the client, use the C$COPY library routine to transfer it to the client. (Refer to section 7.2.1, “Copying Files Between the Client and Server,” for more information.)

If *filename* is not specified or is set to a space, the image is placed on the client machine’s clipboard.

The setting of *colordepth* is important to consider in a thin client deployment. Because color density is a memory-intensive setting and the image may be transferred over network connections, a combination of high resolution and low bandwidth can have a negative effect on thin client application performance. Consider how the image is being used and set the color density only as high as needed.

The descriptions of *filename* and *colordepth* in the previous paragraphs also apply to their usage in the WBITMAP-CAPTURE-DESKTOP function of the W$BITMAP library routine.

The WBITMAP-CAPTURE-CLIPBOARD function copies the current bitmap content of the client machine's clipboard. The description of *filename* in the previous paragraphs also applies to its usage in WBITMAP-CAPTURE-CLIPBOARD, except that for this function, *filename* is mandatory.

4.5 International Character Handling

Often in client/server environments, the client and server machines may use different character encoding, particularly if one machine is set up for foreign language characters that use values outside the ASCII character set (decimal values 128 and higher), or if the client is using a PC character set and the server is using a UNIX character set. This presents a problem when programs or data are passed between the environments, because the characters do not translate directly.

4.5.1 In Distributed Processing

If you anticipate passing any items with special characters (such as vowels with a grave accent, acute accent, circumflex, or tilde) during a remote CALL, then you should create a map file to reconcile the character encoding for you. You should also consider creating a map file if the client machine uses a different character set from the server machine. The map file should specify which client characters are to be converted to which values before passing the CALL’s arguments to the server process or returning information from the server process. The translation on returning data will affect items that were passed to the server process as “BY REFERENCE” (the default).

The map file should re-map only those values that vary between the two character sets. It should contain two values per line: the first indicating the decimal or hexadecimal value of the special character on the client machine, and the second indicating the decimal or hexadecimal value of the corresponding character on the server machine. You can check the values of specific characters by using the Windows Character Map accessory in the PC environment, or by referring to your UNIX man pages in the UNIX environment.

In your character map, hexadecimal values should use the standard “0x” notation. For instance:

```
0x90 0xC9
```

maps “E” (acute) in the IBM PC character set to “E” (acute) in the ISO8859-1 character set using hexadecimal notation.

```
144 201
```

gives the same mapping using decimal notation.

You can use the pound sign (#) to indicate a comment, if desired.

Note that the map will be used to translate only alphanumeric fields, but it will translate *all* alphanumeric fields, including group items and items subject to a REDEFINES clause. If this is not a desired behavior, you may need to restructure your program to avoid these clauses by passing the elementary items instead of the group item, or passing an item from the REDEFINES clause instead of the first reference.

For example, if you pass the group-name in the following COBOL program:

```
01 group-name
   03 field-1   pic99 comp-5.
   03 field-2   pic99 comp-5.
call "sun3:/usr/obj/prog2" using group-name.
```

translation will occur on the elementary numeric items. If these numeric items contain binary data that matches the value of mapped characters, the data will be corrupted. To correct this situation, you could change the CALL statement to:

```
call "*sun3:/usr/obj/prog2" using field-1, field-2.
```

Or you could change the definition of the numeric items to a type that will not conflict with potentially mapped characters, as in:

```
01 group-name
   03 field-1   pic99.
   03 field-2   pic99.
```

Defined this way, the numbers are stored as the ASCII representation of each digit, which should not conflict with any character mapping.

Once the map file is created, you place it on the client, or if the client is AcuServer enabled, you can place it on the client or server. Either way, you specify the location of the map file using the DEFAULT_MAP_FILE or server_MAP_FILE variable in the client configuration file (for example, “client.cfg”). Refer to Chapter 5 for more information on using these variables.

4.5.2 In Thin Client

International character handling in thin client is similar to that in distributed processing. You create a simple text file that contains the mapping of characters that may be different on the client than on the server, as described in the previous section. You need to include only the character codes that differ between the two machines.

After you create your map file, you need to point to it using a runtime configuration variable. For a thin client environment, this variable is called TC_MAP_FILE. Character mapping is triggered by the presence of the TC_MAP_FILE variable. Refer to TC_MAP_FILE for more information about this configuration variable.

You need to be sure that the configuration file that contains this variable is referenced in the program’s alias definition. Information about the thin client alias file can be found in section 2.4, “Creating a Server Alias File in Thin Client.”

5 Preparing the Client(s) in Distributed Processing

Key Topics

5.1 Installing the ACUCOBOL-GT Runtime

For the distributed processing deployment, there is nothing on the AcuConnect® distribution media to install on the client machine. However, you should ensure that every client system that will use AcuConnect has a licensed copy of the ACUCOBOL-GT® runtime that matches the version of AcuConnect installed on the server. For example, if you have installed AcuConnect Version 8.0 on the server, then the client should have the ACUCOBOL-GT runtime Version 8.0 installed.

To check the version number of the client runtime, use the "-v" option on the runtime command line. The runtime will display information similar to the following:

```
ACUCOBOL-GT runtime version 8.0
Serial number 1234
Licensed for 2 user(s), 2 processes per user
Copyright (c) 2008, Micro Focus (IP) Ltd.
```

A "-vv" (double-v) option also displays the runtime's version number, along with extended information. The "-vvv" (triple-v) option, valid on UNIX systems, results in the display of additional configuration information about the UNIX port that is primarily useful to the our Development team. The information displayed varies with individual UNIX systems and is subject to change without notice.

If you have an earlier version of the runtime installed, or no version at all, contact your Micro Focus ***extend***® Sales Professional for information on purchasing the latest version of the runtime. For instructions on installing an ACUCOBOL-GT runtime, refer to the *ACUCOBOL-GT User's Guide*.

5.1.1 Relinking the Client Runtime

No relinking is necessary to use AcuConnect. The regular runtime can call an AcuConnect server program. If you need special extensions for which you would normally relink the runtime, refer to Chapter 6 in *A Guide to Interoperating with ACUCOBOL-GT.*

5.1.2 Removing AcuConnect Client Support From the Runtime

The procedure for removing AcuConnect client support from your runtime varies with your operating system. For Windows, delete the “aclnt.dll” file from the bin subdirectory. For UNIX, use the following procedure:

1. Remove “sub.o”.
2. Set NO_ACUCONNECT=1 in “config85.c”.
3. Run the following make command:

```
make
```

5.1.3 Passwords for Clients

If you are requiring a client password to access the server machine, you can prompt the user for this password and store it in the *acu_client_password* variable, or you can let the runtime prompt for it.

5.1.4 Setting Up the Host Name

The runtime system on each client machine needs to know the name of the server host machine. This is accomplished in the client configuration file, “client.cfg.” You can either use the CODE_PREFIX variable to define the remote pathname, or you can define code name aliases. Code name aliases are faster, because the runtime does not have to search remote machines. If you use CODE_PREFIX, we recommend that you place the remote paths at the end of the variable. Refer to section 5.2.1, “Defining Remote Application Path,” for details.

5.1.5 Confirming Network Services

Whether your client is running UNIX or Windows, confirm that your TCP/IP software is loaded and running. On Windows clients, confirm that “WSOCK32.DLL” is loaded.

5.2 Creating a Client Configuration File

Once you have installed your client applications on the client(s), you must create a special configuration file that tells AcuConnect the location of the remote application component, the runtime flags to use when running the remote application (if any), and the runtime configuration file to use. This file can be named anything, but for clarity, it is called "client.cfg" in this manual. The client configuration file should be placed in the /etc or operating system drive:\etc directory on the client or specified using the "-c" command-line option at runtime.

When AcuConnect is shipped, it includes a sample "client.cfg" file. If you want to install this sample file on your client machine(s), you may do so. This may be a good starting point for creating and modifying your own file.

The "client.cfg" file can contain any standard configuration file variable defined in the *ACUCOBOL-GT User's Guide* (for instance, CODE_SUFFIX, ACUCOBOL, and TEXT). In addition, it can contain any of several AcuConnect-specific variables discussed in this section.

Note: Many of these variables are used to specify the location of remote programs and files. You can use either IP addresses or server names in the "client.cfg" file.

5.2.1 Defining Remote Application Path

You can define the pathname of the remote application in the client configuration file in two ways: by using the CODE_PREFIX variable or by defining code name aliases.

5.2.1.1 CODE_PREFIX

If desired, you can use the CODE_PREFIX variable to define the location of the object programs being called. In a distributed processing environment, the CODE_PREFIX variable can be defined as follows:

```
CODE_PREFIX . /usr/prog1 *servername:/usr/prog2
```

where *prog1* and *prog2* are the directories containing the ACUCOBOL-GT object code (for example, “prog1.acu” and “prog2.acu”). In this example, whenever the client application tries to access a program’s object code, it first looks for the code in the current directory (“.”), then in the local /usr/prog1 directory, and finally on the remote application server “servername” in the directory /usr/prog2.

Because AcuConnect supports Transmission Control Protocol/Internet Protocol (TCP/IP), it can also be used to launch application processes over the Internet. In an Internet environment, the “servername” portion of CODE_PREFIX would be the name of the application server on the Internet.

Notice that the remote server name is preceded by the character “*”. The asterisk indicates that the program is located on the server and must be run on the server as well.

When the client COBOL program executes a CALL, it verifies which directory contains the program (by looking in CODE_PREFIX), and executes the program either on the client or the server.

Note: When you use CODE_PREFIX to specify the location of your called program, the runtime traverses the specified paths to locate the resource. If you use this method, try to place remote notations toward the end of the file. Generally, you achieve better performance by defining a code name alias, because remote machines do not need to be searched using this method.

This architecture allows most existing COBOL programs to operate in a true distributed processing environment without any code modification. You can simply install AcuConnect and modify the “client.cfg” file according to your needs. Even the parameter transfer is performed following standard COBOL procedures (the USING clause after the CALL command).

For example, in a stand-alone environment, CODE_PREFIX could be defined as follows in a Windows system:

```
    CODE_PREFIX c:\prog1;c:\prog2
```

and as the following in a UNIX system:

```
    CODE_PREFIX /usr/prog1:/usr/prog2
```

5.2.1.2 Code name aliases

If you would rather specify the exact location of your called program (rather than having AcuConnect search through CODE PREFIX for the location), you can define *code name aliases*, also in the client configuration file. A code name alias is a substitute name for the program that you're calling.

For example, if you have this CALL statement in your application:

```
CALL "prog2" using customer-info.
```

You could create the alias "PROG2" in your client configuration file by adding two lines:

```
PROG2  *servername:/usr/prog2/prog2.acu
CODE_MAPPING 1
```

The first line creates the alias "PROG2" to represent "prog2.acu" on the application server in directory "/usr/prog2." The second line turns the code alias, and any other code aliases, "on."

With these two lines in the "client.cfg" file, AcuConnect knows to look on "**servername*" for the remote application when it encounters the CALL.

By using aliases in the "client.cfg" file, you enable your users to change their configuration file dynamically at runtime, and you enable them to improve performance.

Note: To disable code aliases, set CODE_MAPPING to "0". In this case, remote program location is derived from the CODE_PREFIX variable. For more information on the CODE_MAPPING variable, refer to Appendix H in *ACUCOBOL-GT Appendices*.

5.2.2 Other Variables

Although the "client.cfg" file can contain any standard configuration file variable defined in the *ACUCOBOL-GT User's Guide*, the variables described in the following sections have specific meaning to AcuConnect.

5.2.2.1 ACUCONNECT_DEBUG_METHOD

This configuration variable gives you some control over where remote program debugging occurs. ACUCONNECT_DEBUG_METHOD is undefined by default and may take one of the following values. These settings also determine what values the ACUCONNECT_DEBUG_METHOD_STRING configuration variable takes.

Value	**Description**
XTERM	The remote runtime creates an xterm and uses it for interaction with the runtime debugger. ACUCONNECT_DEBUG_METHOD_STRING should be set to the name of the X server to which the xterm is displayed.
TERMINAL	The terminal named in ACUCONNECT_DEBUG_METHOD_STRING is used by the runtime to interact with the user for debugging the remote COBOL program.
THIN or THINCLIENT	The client runtime attempts to start an instance of the thin client with the correct parameters using the client machine name and port number specified in ACUCONNECT_DEBUG_METHOD_STRING.

5.2.2.2 ACUCONNECT_DEBUG_METHOD_STRING

This configuration variable is used in conjunction with ACUCONNECT_DEBUG_METHOD to control remote program debugging. This variable is undefined by default.

When ACUCONNECT_DEBUG_METHOD is set to...	**ACUCONNECT_DEBUG_METHOD_STRING...**
XTERM	Should be set to the name of the X server to which the xterm is displayed. If the latter variable is not set, the xterm appears on the X server defined by the A_DISPLAY or DISPLAY environment variable.

When ACUCONNECT_DEBUG_METHOD is set to...	**ACUCONNECT_DEBUG_METHOD_STRING...**
TERMINAL	Must be set to the name of a TTY that is running a runtime started in a special mode (“**runcbl** --wait”). The runtime uses the specified terminal to interact with the user for debugging the remote COBOL program.
THIN or THINCLIENT	Must be set to <*client*:*port*>, where *client* is the name of a Windows machine, and *port* is the port to which the runtime attempts to connect in order to control a thin client. In this mode, the client runtime attempts to start an instance of the ACUCOBOL-GT Thin Client with the correct parameters. For example, if ACUCONNECT_DEBUG_METHOD_STRING is set to “rzack-xp:4444”, the client runtime starts “**acuthin** --wait --port 4444”. Note that if the client machine is not rzack-xp, you may have a thin client instance that is not being used.

5.2.2.3 ACUCONNECT_CLOSE_AFTER_CANCEL

This variable lets you specify whether the connection to a remote application should remain open or be closed after a CALL is complete.

Value	**Description**
“0” (off, false, no) (the default)	Leaves the remote application open until the client application exits.
“1” (on, true, yes)	Closes the remote application whenever a CANCEL occurs in the client program. For example: `ACUCONNECT_CLOSE_AFTER_CANCEL 1`

5.2.2.4 ACUCONNECT_RUNTIME_FLAGS

ACUCONNECT_RUNTIME_FLAGS are parameters that you would normally put on the runtime command line when starting the remote application. These are used when the runtime for the remote application executes.

For example:

```
ACUCONNECT_RUNTIME_FLAGS *servername: -le errfile
```

Any runtime flag documented in the *ACUCOBOL-GT User's Guide* can be used, except "-d". For information on how to run the remote application in debug mode, see section 5.2.2.1, "ACUCONNECT_DEBUG_METHOD," and section 5.2.2.2, "ACUCONNECT_DEBUG_METHOD_STRING."

Note: The server runtime runs in batch mode by default. It's not necessary to include "-b" in the runtime flags.

ACUCONNECT_RUNTIME_FLAGS also contains the name of the runtime configuration file for the remote application. By default, this file is named "cblconfig" and is located in the \etc directory in which you installed AcuConnect.

5.2.2.5 ACURCL_PORT

The ACURCL_PORT variable specifies a particular port number to be used for accessing AcuConnect on the server host machine. This is especially helpful when the server host machine has a security firewall, because firewalls generally allow access only to specific ports. With this variable, the site can ensure that firewall restrictions are satisfied.

The **acurcl** daemon can work with privileged port numbers (from 0 to 1023) and with non-privileged port numbers (1024 and higher, up to 32767).

The value of this variable is overridden by any port number assigned on the command line (via the "-n" option) when the daemon is started.

5.2.2.6 DEFAULT_MAP_FILE

Use this variable to define the name and path of the map file to be used (if any) to map special characters in your client character set to their decimal or hexadecimal equivalent on the server. This file lets you reconcile the character encoding between two machines that use different codes for the same characters. It is particularly useful for international character translation. (See section 4.5, "International Character Handling," for more information.) For example:

```
DEFAULT_MAP_FILE=c:\etc\pc_iso
```

If desired, you can keep your map file on a remote machine and access it using AcuServer®. Simply use remote name notation for your DEFAULT_MAP_FILE entry to point to a server that is running the AcuServer daemon. For example:

```
DEFAULT_MAP_FILE=@sun3:/user/data/pc_iso_map
```

Note that you are not required to keep your map files on the same remote machine that you are calling with AcuConnect.

5.2.2.7 *server*_MAP_FILE

Similar to DEFAULT_MAP_FILE, this configuration variable lets you specify a map file for use with international character translation. However, this variable specifies a map file for use with a specific server. The map file itself can be stored anywhere.

For example, if you include the following statement in your client application:

```
CALL "prog2" using my-data.
```

and your client configuration file includes:

```
CODE_PREFIX *sun3:/usr/obj *sun4:/usr/obj
```

then:

```
sun3_MAP_FILE=c:\etc\pc_iso_map
```

causes the map file "pc_iso_map" to be applied if prog2 is found and run on sun3 but not on sun4. As with DEFAULT_MAP_FILE, you may use AcuServer remote name notation to point to a map file residing on a remote server running AcuServer.

Note: You can have only one DEFAULT_MAP_FILE variable specified in the client configuration file, but you can have as many *server*_MAP_FILE variables as you want.

5.2.2.8 SECURITY_METHOD

This variable lets you determine the security method AcuConnect employs for user logon. This variable must also be set in the server configuration file. AcuConnect can use the operating system's native logon facility or the security provided by the AcuAccess file. It is recommend that you use native system security rather than AcuConnect system security. On Windows 2008 it is essentially required that you use system security. To use native security, you set the SECURITY_METHOD variable. You still create a server access file containing access records that define your user base, but the server access file is used only to check if the user connecting to the server is allowed to connect, and to check to which local account the connection should be mapped.

5.2.2.9 TEXT_*nnn*

A set of TEXT configuration variables is used to control the text of selected AcuConnect messages. Three runtime messages support AcuConnect password handling. The default text for each message is as follows:

```
TEXT_015  "A password is required to connect to host %s."
TEXT_016  "Please enter a password: "
TEXT_017  "Invalid password."
```

Note that in message 15, the name of the AcuConnect host will be substituted for the "%s" characters.

To change the text associated with a given message, follow these steps:

1. Place the word "TEXT" at the beginning of a line in the configuration file.

2. Follow “TEXT” with the following, in this order:

    a. an underscore

    b. the message number

    c. an “=” character (or a blank space or tab)

    d. the text to be used when that message number is displayed.

For example:

```
TEXT_016=Enter your password now:
```

Note that to enable the display of the password messages, you set the PROVIDE_PASSWORD_MESSAGES server configuration variable to “true” (1, on, yes). This variable tells the remote listener to send these messages to the client.

5.2.3 Sample “client.cfg” File

The following file is included in the sample directory on your AcuConnect installation CD-ROM. You can use this file as a starting point and modify it to suit your needs.

```
"client.cfg"
# This is a sample configuration file used when starting your
  program
# on the client.
# You should edit it to match your needs. (See the AcuConnect User's
# Guide).
# Use the "*" notation to point to the cobol object on the server.
code_prefix . *server1:c:\acuconnect\sample.
# Allow code name aliases.  A name alias is a substitute string for
  the
# literal name that appears in the CALL statement.
#code_mapping 1
#prog2 *server1:c:\acuconnect\sample\prog2.acu
# runtime flags on the server
acuconnect_runtime_flags *server1: -c c:\acuconnect\etc\cblconfig
  -le
errfile
# action to be taken with the connection to the remote runtime after
  a
# CANCEL
acuconnect_close_after_cancel 0
```

```
# map special characters in client character set to their
  equivalent
# on server.
#default_map_file c:\etc\pc_iso
```

5.3 Installing Client Programs

The last step in preparing your client for use with AcuConnect in distributed processing is copying the client components of your COBOL application onto the client into the desired directory. If multiple clients will be used to access the server application, then you must install the client application components on each client machine.

5.4 Executing Programs on the Client

After AcuConnect is running on the server, all you have to do to access the remote server application from the client is start the client program. If you have designed your application properly (embedding synchronous or asynchronous CALLs into the client application component, as described in section 4.3.2, "Synchronous or Asynchronous Execution"), the client program will automatically and seamlessly launch the server application for you. The client configuration file, "client.cfg" or equivalent, defines the location of the server, along with the runtime flags and configuration file to be used at the server program run time.

When starting your client program, you can use any of the runtime options available to a standard ACUCOBOL-GT runtime. If you want to provide AcuConnect trace information for the client, specify "-le" plus the name of the error file in which to store the data.

5.4.1 Executing Non-COBOL Programs on the Client

You don't have to have a COBOL object running on the client. AcuConnect can also be used to execute remote COBOL objects from client applications developed in C, C++, Java, .NET, Delphi, or Visual Basic. This relies on the C, Java, and .NET APIs contained in the ACUCOBOL-GT runtime.

To call a remote COBOL object from a non-COBOL program on the client:

1. Install the runtime on the client, as described in section 5.1 of this chapter.
2. In the runtime configuration file on the client, set CODE_PREFIX to point to the AcuConnect server.
3. Use the C, Java, or .NET API contained in the runtime to call COBOL functions from the non-COBOL program. Refer to *A Guide to Interoperating with ACUCOBOL-GT* for information on each of these APIs.
4. Set the u.pass_type member of the argument structure of the runtime's "sub85" interface to control how data is passed back and forth between the non-COBOL program and the remote COBOL object: by reference, by content, or by value. Refer to section 6.3.3.2 of *A Guide to Interoperating with ACUCOBOL-GT* for more details.
5. Run the application developed in C, C++, Java, .NET, Delphi, or Visual Basic on the client.

AcuConnect will execute a COBOL object remotely and share data with the x client.

5.4.2 Debugging in a Distributed Processing Environment

You can interact with the runtime debugger and debug remote programs in two ways.

The first way is to view a debugger window on the server itself: This is accomplished by starting acuconnect with the "–d" flag, or example:

```
acuconnect -start -d.
```

When started with this flag, AcuConnect will run in the foreground. This means that all logging messages will be sent to the console or command line window. On UNIX, only one connection is allowed at a time. On Windows, multiple users can connect simultaneously. When a server runtime is started by AcuConnect, a debugger window will be displayed to the console. On Windows, multiple debugger windows can be opened for multiple connections. The server debugger is a fully functional.

The second way to debug a remote program is to direct the debugger window to be opened on a remote machine with a TTY, an xterm, or the thin client.

If you want to use an xterm, the remote runtime will create an xterm and use it for interaction with the runtime debugger. Alternatively, you can specify the name of a TYY that will be used by the runtime to interact with the debugger. You can also specify a Windows client machine and port number. The client runtime will then start an acuthin instance for interacting with the debugger. For specific instructions in implementing these options refer to the following AcuConnect configuration variables:
ACUCONNECT_DEBUG_METHOD and
ACUCONNECT_DEBUG_METHOD_STRING

5.5 Sample Programs

This section contains sample client and server programs: "prog1.cbl", "prog2.cbl", and "asynch.cbl". By default, these files were placed in your acuconnect/sample directory during installation on the server. These programs are included in the sample directory on your AcuConnect installation media.

"prog1.cbl" is a client program that performs a synchronous CALL. Notice the CALL statement in the Procedure Division section of this program. This CALL invokes "prog2" on the server and displays the result on the client workstation. Remember that the client configuration file ("client.cfg", for example) defines the location of the server. In this case, it also defines the CODE_SUFFIX, ".acu".

"asynch.cbl" is a client program that performs an asynchronous CALL. Notice that in the Procedure Division section of this program, a CALL is made to the C$ASYNCRUN library routine using "prog2" and some parameters. This tells AcuConnect to allow processing on the client to continue. Notice also that this program later calls the C$ASYNCPOLL routine to check the status of "prog2".

Both "prog1.cbl" and "asynch.cbl" start "prog2" on the server using the same "client.cfg" file.

After you have prepared your server and client, you can try running “prog1.acu” and/or “asynch.acu” on the client to test your AcuConnect connection. For instructions, refer to section 5.6, “Running the Sample Programs.”

Note: The sample file “prog1.acu” was compiled with “ccbl32 -o @.acu prog1.cbl”. “prog2.acu” was compiled with “ccbl32 -o @.acu prog2.cbl”.

“prog1.cbl”

```
identification division.
program-id. prog1.
working-storage section.
01  err-code                    pic x(2).
01  err-message                 pic x(60).
01  customer-info.
    05  requested-age.
        10  low-age             pic x(2).
        10  high-age            pic x(2).
    05  age-group-count         pic 9(3).
procedure division.
main-logic.
    move "35" to low-age.
    move "39" to high-age.
    call "prog2" using customer-info
        on exception
            perform exception-handling
        end-call.
  display "Number of customers between " low-age " and "
        high-age ": " age-group-count.
    accept omitted.
    stop run.
exception-handling.
    call "C$CALLERR" using err-code, err-message.
    display "Error#: " err-code ", " err-message.
    accept omitted.
    stop run.
```

“prog2.cbl”

```
identification division.
program-id. prog2.
environment division.
```

```
input-output section.
file-control.
select customer-file
    assign to "custfile"
    organization is line sequential
    file status is customer-status.
data division.
file section.
fd customer-file.
01 customer-record.
    05 customer-id           pic 9(3).
    05 customer-name         pic x(20).
    05 customer-age          pic xx.
working-storage section.
01  customer-status pic xx.
    88  eof-customer-file    value "10".
linkage section.
01  customer-info.
    05  requested-age.
        10  low-age          pic x(2).
        10  high-age         pic x(2).
    05  age-group-count      pic 9(3).
procedure division using customer-info.
main-logic.
    perform count-customers.
    goback.
count-customers.
    move zero to age-group-count.
    open input customer-file.
    perform until eof-customer-file
        read customer-file next record
        at end
            continue
        not at end
            if customer-age >= low-age and
                customer-age <= high-age
                add 1 to age-group-count
            end-if
        end-read
    end-perform.
    close customer-file.
```

“asynch.cbl”

```
identification division.
program-id. asynch.
author. MJE.
remarks.
  Used to test asynchronous calls.  Need prog2 on server.

working-storage section.
01  h-call-prog2           usage handle.
01  state-of-call          pic s9.
01  call-status            pic xx.
    88  call-complete      value "ok".
01  customer-info.
    05  requested-age.
        10  low-age        pic x(2).
        10  high-age       pic x(2).
    05  age-group-count    pic 9(3).

procedure division.
main-logic.
    move "35" to low-age.
    move "39" to high-age.
    call "C$ASYNCRUN" using h-call-prog2 "prog2"
  customer-info.
    if return-code not equal zero
         display "CALL ERROR#: " return-code
         accept omitted
         stop run
    end-if.
   display "age-group-count immediately after async call: "
         age-group-count.
    display "Begin sleep for 5".
    call "C$SLEEP" using 5.
    display "End sleep, call asyncpoll".
    perform until call-complete
     call "C$ASYNCPOLL" using
         h-call-prog2 state-of-call customer-info
     if return-code not equal zero
         display "CALL ERROR#: " return-code
         accept omitted
         stop run
     end-if
     if state-of-call = 1
```

```
            display "Number of customers between " low-age "
and "
                      high-age ": " age-group-count
            set call-complete to true
            accept omitted
      end-if
    end-perform.
    stop run.
```

5.6 Running the Sample Programs

To test your AcuConnect connection between the client and server, you can copy the sample client programs onto the client and try to run them. If the connection succeeds, the server returns a response to the client, and that response displays on the client workstation. The following sections describe how to run the sample programs in UNIX and in Windows.

5.6.1 Running the Sample Programs in UNIX

1. Start AcuConnect on the server by going to the server and typing:

   ```
   acurcl -start -c acurcl.cfg -e server.err
   ```

2. Copy the programs “prog1.acu” and “asynch.acu” from the sample directory on the server to any directory on the client.

   If you kept the installation defaults, these files can be found in the /acuconnect/sample directory on the server. You may want to create a new directory on the client to hold the samples.

3. Start the sample client program by changing to the directory in which you installed the programs on the client and typing:

   ```
   runcbl -c client.cfg prog1.acu
   ```

   to test synchronous CALLs, or

   ```
   runcbl -c client.cfg asynch.acu
   ```

   to test asynchronous CALLs.

5.6.2 Running the Sample Programs in Windows

1. Start AcuConnect on the server by using the "**acurcl** -start" command or by clicking **Start** on the graphical control panel's Services tab.
2. Copy the programs "prog1.acu" and "asynch.acu" from the sample directory on the server to any directory on the client.

   If you kept the installation defaults, these files can be found in the c:\Program Files\acucorp\acucbl8xx\AcuGT\sample directory on the server. You may want to create a new directory on the client to hold the samples.
3. Start the sample client program by opening a Windows Command Prompt (DOS window) and changing to the directory in which you installed the programs on the client. From the prompt, type:

   ```
   wrun32 -c client.cfg prog1.acu
   ```

   to test synchronous CALLs, or

   ```
   wrun32 -c client.cfg asynch.acu
   ```

   to test asynchronous CALLs.

5.6.3 Results

Whether you are working in Windows or UNIX, you can expect the same results from the sample programs. If the connection is unsuccessful, you receive an error message. Refer to section 8.7, "AcuConnect Distributed Processing: Troubleshooting," for advice.

Synchronous sample

When "prog1" CALLs "prog2", AcuConnect automatically starts "prog2" on the server with the runtime flags and configuration file specified in the sample "client.cfg" file. This is equivalent to typing, "wrun32 -b -le prog2.err -c cblconfig prog2.acu" on a Windows server or "runcbl -b -le prog2.err -c cblconfig prog2.acu" on a UNIX server.

If the connection is successful, the following displays on the client:

```
    Number of customers between 35 and 39:  003
```

Asynchronous sample

When “asynch.acu” CALLs “prog2”, AcuConnect automatically starts prog2 on the server with the runtime flags and configuration file specified in the sample “client.cfg” file. Since C$ASYNCRUN was called as well, however, AcuConnect allows the client program, “asynch.acu”, to continue running this time.

If the connection is successful, the following displays on the client:

```
    age-group-count immediately after async call:
    Begin sleep for 5
    End sleep, call asyncpoll
    Number of customers between 35 and 39:  003
```

6 Preparing the Client(s) in Thin Client

Key Topics

6.1 Preparing the Client

In the AcuConnect® Thin Client environment, the client is also known as the display host. This is where screen, mouse, and keyboard handling is performed, but no actual application processing is performed on the client. Unlike servers (or application hosts), which are considered to be "thick," display host clients are "thin" because of their lack of application components and substantive processing work. Thin clients communicate with their application hosts through a split runtime, the thin portion of which is installed on the client.

The following two steps are all you need to do to prepare the client for AcuConnect Thin Client operation:

1. Install client software.
2. Run the ACUCOBOL-GT® Thin Client.

6.2 Installing the ACUCOBOL-GT Thin Client

For in-house installations, the ACUCOBOL-GT Thin Client installation files are included on the product media. A self-extracting installation program ("atcinst.exe") can be found in the REDIST directory on either the Windows or the UNIX CD-ROM. In accordance with the applicable license agreement, this file can be distributed within your organization or placed on a shared network drive. Users double-click this file and follow the on-screen instructions to install the thin client. The thin client can also be selected as an installed product from the Windows CD-ROM.

Once it is installed, you can invoke the ACUCOBOL-GT Thin Client from a command line or you can set up an icon for it. We recommend that you create an icon for the thin client, specifying command parameters as desired. Command parameters for the thin client are described in section 6.5.1, "The acuthin Command." For information on establishing program icons, refer to your Windows documentation.

You can also download and install the ACUCOBOL-GT Thin Client from Micro Focus's Web site. Click the link for Micro Focus's download page, then click the link for the ACUCOBOL-GT Thin Client. Follow the displayed instructions to download the file. When the download is complete, follow the instructions on the screen to install the thin client.

When you install the thin client from "atcinst.exe" or from Micro Focus's Web site, the system displays the terms and conditions for using the ACUCOBOL-GT Thin Client. You must agree to the terms of the agreement to install or use the thin client.

Note: Vista's UAC mechanism blocks the installation of ActiveX controls, such as the Thin Client, onto systems that are not running Internet Explorer 7 in administrator mode. Vista includes an ActiveX installer service that allows IT administrators to use Group Policy to specify Web sites from which standard users will be allowed to install ActiveX controls.

6.3 Installing ActiveX Files

If the server application uses ActiveX resource files or ActiveX controls, these resources and controls must be installed on the thin client machine before the application can be executed. This is because there is no way to send such resources along with screen commands to the UI layer in a single network message, and if they are sent separately, the UI layer doesn't "know" anything about how to group them into a Screen Section item.

End users can usually install and register ActiveX controls and resources by downloading the controls from the control vendor's Web site and running the accompanying setup program.

For simple controls, end users can usually accomplish installation and registration by downloading or copying the ActiveX control files (at least an ".ocx" or ".dll" file) to their hard disk and executing the following command:

```
    regsvr32 <ocx or dll name>
```

"regsvr32.exe" is normally located in the \windows\system directory and may not be in the end user's search path. Even if the control is already installed on the machine (for instance, if it came with other software that's been installed), the user may still need to register the control with "regsvr32.exe."

Note that the ability to load and destroy ImageList data types used by ActiveX controls is supported in thin client via the W$BITMAP library routine. Information about this function can be found in Appendix I in *ACUCOBOL-GT Appendices*.

6.4 Thin Client Splash Screen

The ACUCOBOL-GT Thin Client displays a splash screen on startup. This screen remains in place until the application's initial window appears. You can disable the splash screen by using the "--nosplash" option in the **acuthin** command line. For example:

```
acuthin --nosplash myserver:5764 myprog
```

Another method for suppressing the splash screen is to include the following in your ".atc" or ".acutc" command file:

```
atc_splash_screen   off
```

where *off* can also be "0", "no", or "false".

You can also use a splash screen of your own design. Simply name it "acuthinsplash.bmp" and place it in the same directory as "acuthin.exe".

6.5 Launching Remote Programs From the Client

Assuming that AcuConnect has been started on the server, you can launch remote server programs from a Windows client by issuing the **acuthin** command, either on the command line or from an icon. To point the client machine to the remote application host, specify a server name and port number along with an alias name and arguments.

6.5.1 The acuthin Command

The **acuthin** command parameters can be specified as follows:

```
acuthin server[:port] [rarg1 [rarg2 . . .]] alias [carg1 [carg2 .
  . .]]
```

or

```
acuthin acurcep://server[:port]/alias[?carg1[&carg2 . . .]]
```

where:

[ ]	optional parameters
server	server name
port	port number
rarg1	an argument passed on the runtime command line before the program name
carg1	an argument passed on the runtime command line after the program name
acurcep	ACUCOBOL-GT Remote COBOL Execution Protocol
alias	an alias that identifies a COBOL application on the server

For example:

```
acuthin myserver:5633 –d myprog1 10 20 30
```

or

```
acuthin acurcep://myserver:5633/myprog1?10&20&30
```

If you execute the **acuthin** command with an invalid argument, AcuConnect displays a standard “Usage:” error message. For example:

```
Usage: acuthin [--ping | -p] server:port
acuthin --accept
acuthin --register
acuthin --unregister
acuthin [-t [n]] server[:port] [runtime options] ALIAS
[COBOL arguments]
```

Other **acuthin** command-line options allow you to perform specific functions. The "-d" debugging option is described in the following section. For a description of debugging a COBOL program from within a transaction processing environment, refer to section 6.5.1.2, "Debugging in a transaction processing environment." Information about command-line options to test your connection ("-p" or "--ping") can be found in section 6.5.1.3, "Testing your AcuConnect connection." Options for designating username and password ("--user" and "--password") are described in section 6.5.1.4, "Setting username and password."

6.5.1.1 Debugging option

If you want to test and debug your application in a thin client configuration, you can run the ACUCOBOL-GT Thin Client with the "-d" argument. This causes the server to be started in debug mode. For example, the following command line:

```
acuthin server[:port] -c <path>\configfile –dlxe trace <progname>
```

would create a file named "trace" on the server in the directory specified by the alias. You can specify a full path, but it must be for the server, not for the client. For example:

```
acuthin server[:port] -c <path>\configfile –dlxe <path>\trace
<progname>
```

If the problem is slow screen display when the debugger window appears on the client screen, use the "ts" command to trace screen control creation.

You may also use the "Ctrl-Break" command to enter the debugger program (this works even if the program is not running an ACCEPT statement, as long as the program periodically makes requests to the user interface or is doing file I/O). If the thin client program is busy (such as looping around calls to external software or performing lengthy computations), insert some ACCEPT var FROM INPUT STATUS statements if you want to be able to use <Ctrl-Break> to enter the debugger in that part of your program. For further information on using Ctrl-Break to enter the debugger, see ACUCOBOL-GT book 2, *User Interface Programming*, section 11.5 "Regarding Debugging."

You should not attempt to execute a shell command (!) while your program is in debug mode. If you do, you receive the error message “Unable to start shell in thin-client mode.”

6.5.1.2 Debugging in a transaction processing environment

You can debug your program from within a transaction processing environment, as long as this capability is supported by the vendor of that environment. Please see your vendor’s documentation for details.

Use the following **acuthin** command-line syntax from your Windows machine:

```
acuthin --wait [--port nnnn]
```

where *nnnn* is the desired port number. When this option is used, the thin client waits for the transaction server to start the runtime, after which you can debug your application in the runtime debugger screen.

In this mode, the **acuthin** command has an additional option, “--restart”. With this option, the thin client restarts itself after each debug session, so if a transaction requires multiple COBOL programs to run, you can debug all of them. To terminate the thin client after your debugging session, use the Windows Task Manager.

On UNIX servers, an environment variable setting is required before the transaction server can start the runtime. A_DEBUG_USING_THIN must be set to a nonzero numeric value for debugging on thin client in transaction processing to occur. No additional environment settings are required for Windows servers.

Information about transaction processing environments can be found in Chapter 9 of *A Guide to Interoperating with ACUCOBOL-GT.*

6.5.1.3 Testing your AcuConnect connection

To test your initial AcuConnect setup, you can add a command-line option to the **acuthin** command. Add the “-p” or “--ping” option to test whether AcuConnect is running on the server. The command-line syntax is as follows:

```
acuthin -p [servername]
```

If a connection is established, the program shows results for message ID, time at client, time at server, and round trip time. If the client doesn't have the proper runtime installed, a message displays to this effect.

6.5.1.4 Setting username and password

Two **acuthin** command-line arguments let you send a username and a password for AcuConnect to use when it reads the AcuAccess file. The "--user <*username*>" option allows you to log into the server as "username" rather than as the current logged-on user. The "--password <*password*>" option lets you send the given password to the server without being asked for it. If the password is not valid, the thin client does not allow the connection to occur. These options should appear before the server:port designation in the **acuthin** command line.

These options can also be set in a thin client command-line file. Use the "atc-user" and "atc-password" options described in section 7.3.1, "Thin Client Command-line Files."

You are required to set a password on the server if you use the "--user" command-line option. In this situation, a prompt for the password appears after *username* is passed. If a password is not required, the "--user" option does not work.

6.6 Launching Programs on the Internet

The ACUCOBOL-GT Thin Client works in an Internet environment as well as a local-area or wide-area network environment. When end users invoke the thin client in an Internet environment, they simply enter the server name and, optionally, the port number of the server they are accessing over the Internet.

For example:

```
    acuthin myserver.acucorp.com:5633 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 **acuthin** command line, as in the following example:

```
acuthin 128.110.121.42:5633 myprog1 10 20 30
```

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

If you include your application on a Web site as described in section 7.3, "Launching Thin Client Applications From a Web Page," end users can launch your program from their browsers rather than typing command-line syntax. When they visit your Web site and click a link, the ACUCOBOL-GT Thin Client is automatically invoked on their machine, initiating thin client communication with the server.

6.7 Using the Client Cache Directory

When the thin client needs to use a bitmap (".bmp"), wave (".wav"), or ActiveX resource file, it downloads the file from the server to a local cache directory. This directory is the one identified by the value in the TEMP environment variable. If no directory is designated in the TEMP variable, the file is downloaded to the client's current working directory.

The client downloads a particular bitmap, wave, or resource file on subsequent requests only if the file on the server is a different size or has a later timestamp than the corresponding file in the client cache directory, or the file in the cache is deleted for any reason.

The thin client assigns a unique name to each file that it downloads to the cache. This name includes characters from the server name, port number, alias name, COBOL program archive or library name, server path, and server file name. For example, two bitmap files with the same name might be uniquely identified in the cache directory as shown:

```
myserver5764myalias,library1.acu,image.bmp
myserver5764myalias,library2.acu,image.bmp
```

If your application calls the WIN$PLAYSOUND library routine, and the named sound refers to a disk file on the server or a resource in the program or a library, the client downloads the file to its cache directory, even from a UNIX server. Another use for the cache directory includes storing keystroke files from the W$KEYBUF library routine. (Refer to section 7.2.3, "Using Files Containing Keystrokes," for information about the use of W$KEYBUF.)

If your application calls the W$BITMAP routine, there may be a limit to the number of characters you may use in a file name. You are limited to 127 characters if the name passed to W$BITMAP is a file name and extension only AND you want W$BITMAP to search for a matching file in the following directories and the order shown below:

1. The directory in which an application is loaded.
2. The current directory.
3. The Windows system directory.
4. The 16-bit Windows system directory.
5. The Windows directory.
6. The directories that are listed in the PATH environment variable.

7 Thin Client Special Topics

Key Topics

7.1 Introduction

This chapter covers several topics of interest for thin client users:

- Calls to some ACUCOBOL-GT library routines and dynamic link libraries (DLLs) in thin client involve special procedures
- Instructions for starting a thin client application from a Web page
- A highly configurable automatic update feature for the ACUCOBOL-GT Thin Client piece is described in detail

7.2 Using Library Routines and DLLs in Thin Client

Calls to some library routines and dynamic link libraries (DLLs) may have some unique characteristics in thin client environments, as opposed to distributed processing deployments. Implementations of the C$COPY, C$SYSTEM, and W$KEYBUF library routines, among others, are included in this category. The method for calling DLLs on the display host as well as on a Windows application host (server) is also somewhat different for thin client deployments. The following sections describe these functions for the thin client.

7.2.1 Copying Files Between the Client and Server

The C$COPY library routine allows the transfer of files between the application host and the display host in thin client and between directories on the display host. A special name notation of “@[DISPLAY]:” indicates that a particular file is located on the display host, or client. Use this designation for the source file or the destination file in the call to C$COPY. If the copy operation is performed entirely on the client side, the “@[DISPLAY]:” notation appears in both the source and destination file parameters. As an example, for a file transfer from the server to the client, “@[DISPLAY]:” precedes the destination file name, as shown:

```
C$COPY "/usr/data/file1.ext" "@[DISPLAY]:C:\My Documents\file1.ext"
```

For a file transfer from the client to the server, “@[DISPLAY]:” precedes the source file name. For example:

```
C$COPY "@[DISPLAY]:C:\My Documents\file1.ext" "/usr/data/file1.ext"
```

Path specifications on the client are relative to the directory identified in the TEMP user environment variable. On the server, they are relative to the working directory specified in the application's alias definition. For example, if TEMP is C:\Documents and Settings\joe\Local Settings\Temp and the application's working directory on the server is /usr/apps/app1, then the following statement:

```
    C$COPY "@[DISPLAY]:file1.ext" "file2.ext"
```

uploads "C:\Documents and Settings\joe\Local Settings\Temp\file1.ext" from the client to "/usr/apps/app1/file2.ext" on the server.

Access to files on the display host is also accomplished via a set of directory specifiers. If the file name on the client starts with a special directory specifier, the thin client attempts to locate those files in special Windows directories. The special directory names are as follows:

Identifier	**Directory**
<APPDATA>	C:\Documents and Settings\<user>\Application Data
<COMMON_APPDATA>	C:\Documents and Settings\All Users\Application Data
<COMMON_DOCUMENTS>	C:\Documents and Settings\All Users\Documents
<DESKTOP>	C:\Documents and Settings\<user>\Desktop
<LOCAL_APPDATA>	C:\Documents and Settings\<user>\Local Settings\Application Data
<MYDOCUMENTS>	C:\Documents and Settings\<user>\My Documents

Note that these directories are not necessarily the same for all versions of Windows, and may in fact be on network drives.

If further directory information is specified, the thin client attempts to create the directories added. For example, when attempting to access <APPDATA>\ACUCOBOL-GT\settings.dat, the thin client ensures that the ACUCOBOL-GT directory actually exists as a subdirectory of the Application Data directory.

If the syntax for files on the display host is

```
C$COPY "@[DISPLAY]:"<APPDATA>/filename
```

the thin client would attempt to locate the specified file in

```
C:\Documents and Settings\<user>\Application Data
```

For security purposes, user authorization is required to allow a file transfer to the server using C$COPY. Copying a file to a location other than the directory named in the TEMP user environment variable also needs user authorization. The thin client displays a dialog box requesting the authorization.

Note that when the “@[DISPLAY]:” notation appears in the first or second parameter in the call to C$COPY, the optional parameter FILE-TYPE is not used. Only the single file named as the source file is copied. When an application is not running in the thin client, the “@[DISPLAY]:” notation is ignored.

Using C$COPY with AcuServer

A technical limitation in AcuConnect affects the ability to copy a file from a remote AcuServer server directly to the client in a thin client environment. The following code, for example, does not accomplish the desired copy function:

```
CALL "C$COPY" USING "@SERVER:c:\serverpath\test.dat"
  "@[DISPLAY]:c:\clientpath\test.dat"
```

A recommended workaround involves the use of three CALL statements in your code. First, have the AcuConnect server runtime copy the file from AcuServer to a temporary file. Use the “@[DISPLAY]:” notation to copy that file to the thin client. Finally, delete the temporary file on the server. For example:

```
CALL "C$COPY" USING "@SERVER:e:\serverpath\test.dat" "tempfile"
CALL "C$COPY" USING "tempfile" "@[DISPLAY]:e:\clientpath\test.dat"
CALL "C$DELETE" USING "tempfile"
```

7.2.2 Executing Desktop Programs

Your server application can execute desktop programs via the ACUCOBOL-GT C$SYSTEM library routine. To achieve this functionality, add the CSYS-DESKTOP flag to the set of flags passed to C$SYSTEM. This flag indicates that the application wants to run the program on the client machine rather than the application server.

Note that the thin client appears “frozen” while the command executes in order to return the termination status to the application correctly. To avoid this behavior, run the command asynchronously by specifying the CSYS-ASYNC flag.

For example, the following code fragment starts Notepad on the client machine:

```
77 flags      pic 9(4) comp

add csys-desktop, csys-async giving flags
call "c$system" using "notepad", flags
```

Because C$SYSTEM can perform harmful actions on the desktop (like removing files), the thin client asks the user for permission to run the command. The user can choose not to see the question again for a specific application host.

For detailed information about C$SYSTEM, refer to Appendix I in *ACUCOBOL-GT Appendices*.

7.2.3 Using Files Containing Keystrokes

W$KEYBUF library routine behavior is slightly different in thin client environments than in the distributed processing environment. This routine lets you create files of keystrokes that can be played back to simulate user input. Specifically, W$KEYBUF op-codes 7, 8, and 9 have a different implementation in thin client. In a thin client environment, these files of keystrokes are created in a temporary cache file on the thin client machine, then uploaded to the server when recording stops (op-codes 7 and 8). When a file is to be played back, the thin client requests that the file be downloaded from the server to the client’s temporary cache directory, as specified in the

TEMP environment variable. If TEMP is not set, the files are stored in the thin client's current working directory. The file is not downloaded again unless it is changed on the server or deleted from the client.

More information about the W$KEYBUF library routine can be found in Appendix I in *ACUCOBOL-GT Appendices*.

7.2.4 Selecting Files on the Client

A call to the C$OPENSAVEBOX library routine allows you to browse the client machine's drives and select a directory or file. This routine is used in conjunction with other operations that have access to the client machine's drives such as C$COPY, C$SYSTEM, DLL calls, and COM or ActiveX components.

Note that the server machine's files do not appear in the dialog. However, if your server machine is Windows, you can navigate to the server using Universal Naming Convention (UNC) notation. Set the C$OPENSAVBOX parameter OPNSAV-DEFAULT-DIR to the desired mapped drive or server directory using UNC notation.

7.2.5 Accessing Local Resource Files

You can access a local resource file on the display host in thin client by adding the "@[DISPLAY]:" notation to the file name in calls to the W$BITMAP, WIN$PLAYSOUND, W$KEYBUF, or C$RESOURCE library routine. When a file name that you pass to one of these routines begins with "@[DISPLAY]:", the routine attempts to access the file in the display host's file system. It does not download the file from the server. This function is helpful when your application uses many or large resource files and you want to avoid the download delay that occurs when you first run your application.

You can distribute any number of resource files with the thin client executable. You can specify absolute paths or paths relative to the directory location of "acuthin.exe". For example, if you distribute the wave file "MyApp\sounds\sound1.wav", you can access it on the display host with the following statement:

```
CALL "WIN$PLAYSOUND" USING
     "@[DISPLAY]:../../MyApp/sounds/sound1.wav",
     SND-SYNC.
```

Note that if you change a distributed resource file, you need to distribute the updated file to all your users.

7.2.6 Calling Dynamic Link Libraries (DLLs)

In a thin client environment, you can call subroutines in DLLs on the display host (client), as well as on the application host, when the application host is a Windows server. DLL calls to the display host involve some special considerations, which are described in this section. Calling DLLs on the Windows application host uses the COBOL syntax and configuration variables described in Chapter 3 in *A Guide to Interoperating with ACUCOBOL-GT*. However, DLLs that are called on the server side and that interact with the user interface may be of limited use to thin client applications.

7.2.6.1 Calling client-side DLLs

Calling a DLL on the display host requires special notation. The name of the DLL called on the display host must begin with "@[DISPLAY]:". For example, to call a DLL named "MYLIB" on the client, you would use the following code:

```
CALL "@[DISPLAY]:MYLIB.DLL"
```

The file path for a called DLL is relative to the directory that contains "acuthin.exe". For example, if "acuthin.exe" is located in C:\Program Files\Acucorp\Acucbl8xx\AcuGT\bin, then the following call:

```
CALL "@[DISPLAY]:MYAPP\MYLIB.DLL"
```

refers to C:\Program Files\Acucorp\Acucbl8xx\AcuGT\bin\MYAPP\MYLIB.DLL. Note that the CALL statement ignores CODE_PREFIX when it searches for DLLs on the client. CODE_PREFIX is applied only to actions on the application host.

7.2.6.2 Related configuration variables

You can use the CODE_MAPPING configuration variable (see *ACUCOBOL-GT Appendices*, Appendix H) to map CALLs to DLLs on the display host. For example, the following configuration file entry:

```
CODE_MAPPING  1
MYLIB.DLL @[DISPLAY]:MYLIB.DLL
```

causes a call to MYLIB.DLL to execute on the client. In addition to mapping the DLL name, you should map the name of each function contained in the DLL in the configuration file, as shown below for FuncA and FuncB in MYLIB:

```
FuncA @[DISPLAY]:FuncA
FuncB @[DISPLAY]:FuncB
```

It is important that you use the DLL_CONVENTION configuration variable to specify the appropriate calling convention. Refer to *ACUCOBOL-GT Appendices*, Appendix H, for more details about DLL_CONVENTION.

7.2.6.3 Passing pointers in DLL calls

Passing a pointer as a parameter of a DLL call that executes on the display host requires some special handling in code, because the pointer value refers to a memory address on the application host. In certain cases, you may safely pass a pointer BY VALUE. You may never pass a pointer BY REFERENCE. This section includes some guidelines for passing pointers BY VALUE.

Using M$ALLOC

The runtime knows the size of allocated memory and can pass the contents of that memory to the thin client if the pointer is returned by a call to the M$ALLOC library routine. Then, if the DLL call on the client modifies the memory, the display host returns the modified memory back to the application host. As an example, assume you have DLL routines named SetData and GetData that take size and pointer arguments. You may use the following code:

```
77  PTR     USAGE POINTER.
77  ITEM-1  PIC X(100).
...
 CALL "M$ALLOC" USING 100, PTR.
 CALL "M$PUT" USING PTR, "Hello World".
```

```
    CALL "@[DISPLAY]:SetData" USING 100, BY VALUE PTR.
    ...
    CALL "@[DISPLAY]:GetData" USING 100, BY VALUE PTR.
    CALL "M$GET" USING PTR, ITEM-1.
    CALL "M$FREE" USING PTR.
```

The application host sends the 100-byte contents of memory to the client on the CALL to SetData. After the client fills the 100 bytes of memory with GetData, it returns the contents to the server.

Using the "-Zm" compiler option

You can use a new compiler option with the Format 7 SET statement to pass pointers to client-side DLL calls. Use this option if you don't want to add the WITH MEMORY SIZE phrase for each pointer parameter in a CALL statement. (WITH MEMORY SIZE is described in the next section.) If your Format 7 SET statement returns a pointer value and you compile with the "-Zm" option, the compiler generates additional code that provides the runtime with the size of the item specified in the SET statement. Once the runtime has this information, it can pass the memory contents to the thin client and then retrieve the contents after the thin client modifies them. Note that the "-Zm" option adds runtime overhead for each Format 7 SET statement. For information about the SET statement, refer to *ACUCOBOL-GT Reference Manual*, section 6.6, "Procedure Division Statements."

Using the CALL statement

The USING phrase in the Format 1 CALL statement includes a *size-phrase* item with which you can pass the size of a memory item along with the pointer value. Use *size-phrase* when the runtime and the thin client cannot automatically determine the size of the memory being pointed to. This can occur if, for example, a pointer is passed through the Linkage section or is an external item. The syntax is

```
[ USING { [ BY {REFERENCE} ] {parameter} size-phrase ...} ...]
               {CONTENT  }   {OMITTED  }
               {VALUE    }   {NULL     }
```

where *size-phrase* is:

```
[WITH MEMORY SIZE { =  } memory-size]
                  { IS }
```

Memory-size is a numeric literal or data item.

As an example, assume you have pointer PTR that points to 1024 bytes of memory. You can pass this pointer to a DLL called SetData on the display host with the following code:

```
CALL "@[DISPLAY]:SetData" USING BY VALUE PTR
          WITH MEMORY SIZE 1024.
```

Detailed information about the CALL statement can be found in *ACUCOBOL-GT Reference Manual*, section 6.6, “Procedure Division Statements.”

7.2.7 Calling the DLL Version of the ACUCOBOL-GT Thin Client

A DLL interface to the ACUCOBOL-GT Thin Client is designed to be called from languages that don’t support ActiveX or when a DLL interface is preferable to ActiveX. If you decide that a DLL interface is the best method of accessing the Thin Client from your program, you call the thin client DLL, “acuthin.dll”.

In order to call the ACUCOBOL-GT Thin Client DLL from another programming language, you must add a single declaration to the source program. This example shows what you would use in C/C++:

```
int PASCAL AcuThinMain( const char *pCmdLine );
```

To start the thin client, you must pass AcuThinMain a single parameter that specifies a command-line string using the same syntax as required by the “acuthin.exe” command line. For example, in C/C++ you would use the following syntax to start an application on “myserver” using port 5764 with an **acurcl** alias named “myappalias”:

```
int iRetVal = AcuThinMain( "myserver:5764 -d myappalias" );
```

AcuThinMain returns “1” on success, “0” on failure.

The thin client DLL depends on “acme.dll”, “aclnt.dll”, and “atermmgr.dll”. If your program uses “.jpg” files, you also need “ajpg32.dll”.

7.3 Launching Thin Client Applications From a Web Page

If you want thin client users to launch your application from a Web page, you can do the following:

1. Install your COBOL application on your application server host. This machine must also contain the ACUCOBOL-GT runtime, AcuConnect, and the other required software components. 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. This file should have an “.atc” or “.acutc” extension. The format of this file is described in the following section.
3. Place the thin client command-line file on your Web server and create a link to it from an appropriate Web page. Your users click on this link to launch your thin client application.
4. To ensure that your thin client is recognized across a wide variety of browsers, configure your Web server to generate an HTTP “Content-type” response header field, which is included with the thin client command-line file. This field should contain the MIME type “application/vnd.acucorp.thincommandline”. Refer to your Web server documentation for instructions on adding a MIME type.

After you set up your hyperlink, end users who visit your Web page can run the program by clicking on the program’s link. The display occurs in an application window outside of the user’s browser. Note that users who start the thin client from an “.atc” or “.acutc” file that was downloaded from the Internet are presented with a security dialog asking them to authorize the action.

Note: A Windows XP Service Pack 2 user may receive a File Download dialog before the thin client application can execute. If the user clears the “Always ask before opening this type of file” check box the first time it appears, then subsequent requests for this thin client application proceed immediately.

Another way to launch thin client applications from a Web page is the ACUCOBOL-GT Web Thin Client. The Web thin client is an ActiveX control that encapsulates the functionality of the ACUCOBOL-GT Thin Client. It is designed for browsers that support ActiveX controls, particularly Microsoft Internet Explorer. More information about the Web thin client can be found in *A Programmer's Guide to the Internet*.

Users can also launch the ACUCOBOL-GT Thin Client by entering the host name of the application server or an Internet IP address on the "acuthin.exe" command line. Refer to section 6.6, "Launching Programs on the Internet," for more information.

7.3.1 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. The thin client uses the contents of this file to establish its command-line parameters. The thin client command-line file has a format that resembles other configuration files. It may contain the following variables:

Variable name	**Description**
atc-auto-update	Lets you disable the thin client automatic update feature with a setting of "off" (0, false, no)
atc-user	The logon user name (no default value)
atc-password	The password given to the server (a required variable when atc-user is used; no default value)
atc-splash-screen	The option that lets you disable the thin client splash screen with a setting of "off" (0, false, no)
atc-trace	The trace level (default value is "0")
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 (default value is 5632)
atc-runtime-options	Runtime options passed to the runtime via AcuConnect (no default value)

Variable name	**Description**
atc-alias	The alias to execute (a required variable with no default value)
atc-cobol-args	COBOL arguments passed to your COBOL program via AcuConnect (no default value)

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

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

```
acuthin [atc-user atc-password atc-splash-screen atc-trace]
  <atc-server>[:<atc-port>] [atc-runtime-options] <atc-alias>
[<atc-cobol-args>]
```

As an example, assume the following command-line file components:

atc-server	myserver
atc-port	5633
atc-runtime-options	-lxe runtime.err
atc-alias	myprog
atc-cobol-args	10 20 30

These command-line file contents are interpreted as the following command line:

```
acuthin myserver:5633 -lxe runtime.err myprog 10 20 30
```

7.3.2 Using Anchor Tags

Use the HTML anchor tags <a> and </a> to create a link to your COBOL application. Anchor tags are closed elements that 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. Because the link in this case is a thin client command-line file, when users click on the highlighted item, the thin client is automatically invoked.

To turn text into a hypertext anchor, enclose the clickable text in the anchor tags. For example:

```
<a href="myprog.atc">Click here to run the application</a>
```

7.4 Thin Client Automatic Update

The ACUCOBOL-GT Thin Client has an automatic update feature that determines whether the **acuthin** executable on the client is compatible with the server components (the AcuConnect executable **acurcl** and the ACUCOBOL-GT runtime executable **wrun32** or **runcbl**) and triggers an automatic update of the client piece if it is not. This feature is especially helpful if you need to upgrade a large number of clients after you update your server components. With the automatic update, users do not see an error message regarding a version mismatch, and they can use the application immediately after the update.

With this feature, the thin client can automatically download and install a new version of the client piece that matches the server version. It can then restart the user's application using the newly installed thin client.

Note that if your server is running AcuConnect Version 7.2 or later, it cannot automatically update a thin client that is earlier than Version 7.2. However, Version 7.2 can be upgraded from one product build to another via the TC_CHECK_INSTALLER_TIMESTAMP or TC_REQUIRES_BUILD_NUMBER configuration variable. (This function may be useful when installing a patched product.) You can also test the automatic update process by using the "--testautoupdate" **acuthin** command-line option. More information about the configuration variables and command-line options that control the update process appears in the following sections.

7.4.1 Automatic Update Overview

The thin client first connects to the AcuConnect executable, **acurcl**, which is running on the server. This executable looks up the specified alias and starts the ACUCOBOL-GT runtime on the server, which takes over the communication with the thin client. The runtime sends the thin client information about itself, including its version number and thin client protocol number. It also sends several configuration variables that control the thin client automatic update behavior.

You specify these configuration variables in the runtime configuration file for your application rather than in the **acurcl** configuration file. As a result, you can have a different automatic update configuration for each application alias managed by a single AcuConnect server. More information about the automatic upgrade configuration variables can be found in the following sections.

Each of the following five events can cause an automatic update of the thin client:

1. The client's major and minor version numbers do not match the server's.
2. The client's protocol number does not match the server's.
3. The TC_REQUIRES_BUILD_NUMBER configuration variable is a nonzero value, and the client's build number does not match this value.
4. The TC_CHECK_INSTALLER_TIMESTAMP configuration variable is set to "on", and the modification time of the installer file on the server is later than the modification time of the installer file on the client. If the installer file does not exist on the client, the modification time of the running "acuthin.exe" is used.
5. The **acuthin** command has been executed with the "--testautoupdate" command-line option.

The TC_REQUIRES_BUILD_NUMBER configuration variable facilitates the automatic update of the thin client via a comparison of the client's build number with the variable value. You set this variable to the thin client build number required by the application. The default value of this variable is "0" (off, false, no).

When thin client executes, it compares its build number with the value of TC_REQUIRES_BUILD_NUMBER. If the value of this variable does not match the client's build number, the automatic update process is initiated. For example, suppose you have a patched version of the thin client containing some enhancements needed by your application, and you want to use the automatic update feature to deliver the patched thin client to users. Run "**acuthin** -v" to get the build number of the patched thin client, then set

TC_REQUIRES_BUILD_NUMBER to that build number in the runtime configuration file set up for your application. Even though the version and protocol numbers have not changed, the different build number triggers the automatic update. Note that if the value of TC_REQUIRES_BUILD_NUMBER does not match either the old or the new thin client build number, the automatic update process is initiated each time the user launches the **acuthin** executable.

The value of the TC_CHECK_INSTALLER_TIMESTAMP configuration variable determines whether the thin client compares the modification times of the installer files on the client and on the server. If this variable is set to "1" (on, true, yes) and the modification time of the client file is older than the time of the server file, the automatic update process is initiated. If the installer file does not exist on the client, then the comparison is made with the modification time of the thin client executable (**acuthin**) currently running. The default value for this variable is "0" (off, false, no). To use this automatic update method, you should ensure that your client and server clocks are synchronized to avoid unnecessary automatic updates. Times are converted to UTC (Coordinated Universal Time)—formerly known as Greenwich Mean Time (GMT)—so the comparison works across world time zones.

The "--testautoupdate" **acuthin** command-line option lets you test the automatic update feature with your configuration. You need to specify a server, port number, and alias when you use this option.

7.4.2 Automatic Update Process

When the server runtime determines that a thin client automatic update is needed, it triggers the following process (optional steps are enabled by default):

1. The thin client checks to see if the automatic update feature is enabled.
2. If the feature is enabled, the thin client informs the user that an update is required and asks the user to proceed with or cancel the update (optional). If the user accepts the update, the thin client downloads a single "installer" file while displaying a download progress dialog.
3. If the installer file has an ".msi" extension, then the thin client launches the Microsoft Windows Installer ("msiexec.exe") to complete the thin client installation using the contents of the ".msi" file.

4. Thin client waits for the installer process to exit and then starts the newly installed thin client with the same command-line parameters or ".atc" (or ".acutc") file passed to the original thin client process (optional).
5. If the automatic update fails for any reason, the thin client informs the user of that fact (optional).

7.4.3 Enabling or Disabling the Automatic Update Feature

The thin client automatic update feature is enabled by default. The feature can be disabled in one of three ways. You can

- set the atc-auto-update thin client command-line variable in the ".atc" (or ".acutc") file to "0" (off, false, no). The default value is "1" (on, true, yes).
- set the TC_DISABLE_AUTO_UPDATE runtime configuration variable to "1" (on, true, yes). Default value is "0" (off, false, no).
- use the "--noautoupdate" **acuthin** command-line option.

7.4.4 Informing the User When an Update Is Needed

When a thin client event triggers the update process, the thin client displays a message box informing the user that an update is required. The appearance of the box may be triggered by a version or protocol number mismatch or a build number mismatch. At this point, the user may choose to proceed with or cancel the update. Three runtime configuration variables control the message box's appearance.

The default setting of "1" (on, true, yes) for the TC_AUTO_UPDATE_QUERY configuration variable enables the display of the query message box. Setting this variable to "0" (off, false, no) prevents the message box appearance.

You use the TC_AUTO_UPDATE_QUERY_TITLE configuration variable to specify the title bar text in the automatic update query message box. The default value of this variable is

```
ACUCOBOL-GT Thin Client
```

The value of the TC_AUTO_UPDATE_QUERY_MESSAGE configuration variable determines the message displayed in the query message box. The default value depends on the circumstances that triggered the automatic update. For example, if the automatic update is triggered by a version or protocol number mismatch, the message displayed is:

```
Incompatible server version
Server version: <srvvers>, client <clntvers>
Server protocol: <srvproto>, client <clntproto>
Press OK to automatically correct this problem
```

where <*srvvers*>, <*clntvers*>, <*srvproto*>, and <*clntproto*> are replaced by the server version, client version, server protocol number, and client protocol number, respectively.

If the automatic update is triggered by a build number mismatch, the message displayed is:

```
This application requires a newer version of ACUCOBOL-GT Thin
 Client
Build required: <reqbld>, current <curbld>
Press OK to automatically update this software
```

where <*reqbld*> is the value of the TC_REQUIRES_BUILD_NUMBER configuration variable, and <*curbld*> is the build number of the currently running thin client.

If the automatic update is triggered by any other condition, the following message appears:

```
This copy of ACUCOBOL-GT Thin Client is out of date
Press OK to automatically update this software
```

7.4.5 Accepting the Automatic Update

If the user proceeds with the thin client automatic update, the thin client downloads an installer file. we provide a default ".msi" installer file called "acuthin.msi" in the REDIST directory on our product CD. This file contains all of the information needed to update the thin client, including installation instructions and properties. To use this file as your installer file, you should copy it to the directory that contains the **wrun32** or **runcbl** runtime executable. If you rename this file or install it in any other directory, you must set the TC_INSTALLER_SERVER_FILE configuration variable to the path and file name of the installer file.

You may use Microsoft® Visual Studio®, InstallShield®, Wise installer, or a similar tool to create your own ".msi" file if you want to customize the thin client automatic update installation.

You can also create an ".exe", ".html", ".doc", ".pdf", ".bat", or ".cmd" file to run instead of a ".msi" installer. If the installer file ends with ".msi" or ".exe", the thin client executes the file directly. If the installer file does not have either extension, the thin client uses the Windows ShellExecute() API to run it as if the user had double-clicked on the file or opened the file using Windows Explorer. Other issues apply if you use an installer file that does not have the ".msi" extension. (Refer to the description of the TC_INSTALLER_RUN_ASYNC configuration variable in section 7.4.6 for details.)

7.4.5.1 ".msi" installer

When the installer file has an ".msi" extension, the location specified in the TC_INSTALLER_TARGET_DIR configuration variable is used for setting the installation directory that appears in the installer dialog. The "acuthin.msi" file distributed with AcuConnect defines a default location of

```
C:\Program Files\Acucorp\Acucbl8xx\AcuGT\bin
```

If the installation directory for the new version of thin client is the same as the location of the currently running thin client, the installer cannot overwrite the executables and DLLs because they are in use. To avoid this problem, set TC_INSTALLER_TARGET_DIR to a unique name that includes the version number and build number, if necessary.

Note that if you set TC_INSTALLER_RUN_ASYNC (described in section 7.4.6), you may install the new thin client in the same directory as the currently running thin client. The system allows the executable and DLLs to be overwritten as soon as the current thin client exits, which with TC_INSTALLER_RUN_ASYNC, happens immediately after the thin client starts the installer. However, in this instance, the thin client does not automatically restart the application.

7.4.5.2 Other installers

If your installer file does not have a “.msi” extension, it needs to choose or allow the user to choose the installation directory. After the new directory is chosen, your installer needs to update the “.atc” file type association in the registry so that the thin client can locate the newly installed executable after your installer finishes. If you want to install the thin client in a directory relative to the location of the existing thin client, you can determine the location by querying the registry entries for the “.atc” (or “.acutc”) file type association.

The thin client examines the “.atc” file type association in the registry by first reading the value of HKEY_CLASSES_ROOT\.atc to identify the version-specific name of the current thin client. It then reads the value of HKEY_CLASSES_ROOT\<current_version>\shell\open\command to get the absolute directory path to the “acuthin.exe” executable.

For example, after a Version 7.2 thin client installation, HKEY_CLASSES_ROOT\.atc has the value

```
Acucorp.ThinClient.72
```

HKEY_CLASSES_ROOT\Acucorp.ThinClient.72\shell\open\command may have the value

```
"C:\Program Files\AcuThin\acuthin.exe" "%1"
```

This value starts with the absolute directory path for the thin client. Your installer should change this value if you install the new thin client in a different directory.

If the value of TC_INSTALLER_CLIENT_FILE is an executable (“.exe” file), the thin client uses the value of the TC_INSTALLER_ARGS configuration variable as the command-line options passed to that executable. For example, if you want “msiexec.exe” to log all of its operations to a file named “msi.log”, then you could set TC_INSTALLER_ARGS to “/log msi.log”. (See the Microsoft Windows Installer documentation for more details.) TC_INSTALLER_ARGS has no default value.

Note that if the TC_INSTALLER_ARGS configuration variable is set, the TC_INSTALLER_UI_LEVEL configuration variable (see section 7.4.5.5) is ignored.

7.4.5.3 Installer file locations

Two configuration variables specify the locations of the client and server installer files. You set the TC_INSTALLER_SERVER_FILE configuration variable to the path and file name of the server installer file. Its default value is

```
<runtime_path>/acuthin.msi
```

where <*runtime_path*> is the directory that contains the **wrun32** or **runcbl** runtime executable. As mentioned earlier, you can substitute another type of file for the ".msi" file (for example, an ".exe", ".html", ".doc", ".pdf", or ".cmd").

Note that on Windows servers, you should not use drive letters mapped to network shares in TC_INSTALLER_SERVER_FILE. Problems can occur if you try to use a drive letter that is mapped in a different security context. Instead, use Universal Naming Convention (UNC) names to access resources. The format looks like the following:

```
TC_INSTALLER_SERVER_FILE
 \\MyServer\filesharename\directoryname\filename
```

Note: To access network shares, you must either set the AcuConnect SECURITY_METHOD configuration variable to "LOGON" or use the control panel to set the AcuConnect Service "Log On As" property to an account that can access the network share.

Use the TC_INSTALLER_CLIENT_FILE configuration variable to specify the path and file name of the installer file that you want to create on the client. You may use the following special directory names for Windows to specify the value of TC_INSTALLER_CLIENT_FILE:

Identifier	**Directory**
<APPDATA>	C:\Documents and Settings\<user>\Application Data
<COMMON_APPDATA>	C:\Documents and Settings\All Users\Application Data
<COMMON_DOCUMENTS>	C:\Documents and Settings\All Users\Documents

Identifier	Directory
<DESKTOP>	C:\Documents and Settings\<user>\Desktop
<LOCAL_APPDATA>	C:\Documents and Settings\<user>\Local Settings\Application Data
<MYDOCUMENTS>	C:\Documents and Settings\<user>\My Documents

More information about these special directory names can be found in section 7.2.1, “Copying Files Between the Client and Server.”

The default value of TC_INSTALLER_CLIENT_FILE is

```
<APPDATA>\ACUCOBOL-GT\<installer_server_filename>
```

where <*installer_server_filename*> is the file name specified in the TC_INSTALLER_SERVER_FILE configuration variable. By default, the specified file name is “acuthin.msi”. As an example, if user Bob has TC_INSTALLER_SERVER_FILE set to the default, the default value of TC_INSTALLER_CLIENT_FILE is

```
C:\Documents and Settings\Bob\Application
  Data\ACUCOBOL-GT\acuthin.msi
```

7.4.5.4 Download progress dialog

During the installer file download process, a progress dialog appears. You can cancel the download at any time from this dialog. Four configuration variables control the behavior of the download progress dialog.

The default value of “1” (on, true, yes) for the TC_DOWNLOAD_DIALOG configuration variable enables the appearance of the progress dialog. If you set this variable to “0” (off, false, no), the progress dialog does not appear.

The TC_DOWNLOAD_DIALOG_TITLE configuration variable is used to specify the title bar text in the download progress dialog. The default value of this variable is

```
ACUCOBOL-GT Thin Client Automatic Update
```

You use the TC_DOWNLOAD_DESCRIPTION configuration variable to specify the text that appears in the middle of the download progress dialog. Its default value is

```
Downloading installation file. . .
```

Use the TC_DOWNLOAD_CANCEL_MESSAGE configuration variable to specify the message that appears if the download process is cancelled. The Cancel push button is disabled immediately after the user clicks it, the description line in the middle of the dialog disappears, and the message specified in this configuration variable appears. The default value for TC_DOWNLOAD_CANCEL_MESSAGE is

```
Please wait while the download is being cancelled . . .
```

7.4.5.5 Microsoft Windows Installer

After the installer file downloads successfully, the thin client executes the Microsoft Windows Installer ("msiexec.exe"), which installs the new ACUCOBOL-GT Thin Client. Note that "msiexec.exe" is available on, and the ".msi" file extension is automatically recognized by, later generations of Microsoft operating systems. However, Windows NT may require the installation of a patch from Microsoft to execute the ".msi" file. These patch files are "InstMsiW.exe" for Windows NT Service Pack 6 and later.

Distributed with AcuConnect, these two files are located in the REDIST directory on the product CD. For the automatic download to work, you need to copy the appropriate patch file to the same directory as the installer file on the server, that is, the directory containing the runtime executable **wrun32** or **runcbl** or the directory specified in TC_INSTALLER_SERVER_FILE.

The "acuthin.msi" file contains a default location for the install. If you want to install to a different location, use the TC_INSTALLER_TARGET_DIR configuration variable.

Note that on Windows servers, you should not use drive letters mapped to network shares in TC_INSTALLER_TARGET_DIR. Problems can occur if you try to use a drive letter that is mapped in a different security context. Instead, use UNC names to access resources. The format looks like the following:

```
TC_INSTALLER_TARGET_DIR \\MyServer\sharename\directoryname
```

The keywords or numeric values in the TC_INSTALLER_UI_LEVEL configuration variable control the Windows installer interface. Set TC_INSTALLER_UI_LEVEL to NONE or “0” if you do not want “msiexec.exe” to display a user interface. This setting adds the **msiexec** “/qn” command-line option.

Set this variable to UNATTENDED or “1” if you want “msiexec.exe” to display informational and progress messages but to execute unattended. This setting adds the **msiexec** “/passive” command-line option.

Set TC_INSTALLER_UI_LEVEL to INTERACTIVE, DEFAULT, or “2” if you want “msiexec.exe” to prompt for and accept user input for the installation process. For example, the user could choose the directory location for thin client installation. This setting adds the **msiexec** “/qf” command-line option, which is the default **msiexec** option.

Set the variable to REDUCED or “3” if you want “msiexec.exe” to use a reduced user interface. This setting adds the **msiexec** “/qr” command-line option.

7.4.6 Restarting the Application with the New Thin Client

After the installer process is complete, the newly installed thin client is started with the same command line parameters or “.atc” (or “.acutc”) file passed to the original process. The thin client examines the “.atc” file type association in the registry to find the new thin client version. If the registry entries are properly set, when you open the “.atc” file from Windows Explorer, a desktop shortcut, or the start menu, the updated version of thin client is automatically found.

The thin client also adds the “--noautoupdate” **acuthin** command-line option, which prevents an infinite loop in the event an error causes the new thin client to detect that it needs another update. This could happen if, for instance, the automatic update was configured with an incorrect version of the installer that installs an older thin client than required.

You use the TC_INSTALLER_RUN_ASYNC configuration variable when you want to prevent the thin client from restarting after an automatic update or when your installer file handles the automatic update process to completion. When you set this variable to “1” (on, true, yes), the thin client starts the installer process asynchronously and then exits immediately. It does not wait for the automatic update process to complete and does not restart using the same command line parameters or ".atc" file. The default value is "0" (off, false, no).

You may also use TC_INTALLER_RUN_ASYNC to install the new thin client in the same directory as the currently running thin client. When this variable is set to "1", the system allows the executable and DLLs to be overwritten as soon as the current thin client exits, which happens immediately after the thin client starts the installer.

This variable is useful if you want to run an ".exe", ".html", ".doc", ".pdf", ".cmd", or other file instead of a ".msi" installer. For example, you could set TC_INSTALLER_SERVER_FILE to "AutoUpdate.html" and TC_INSTALLER_RUN_ASYNC to "1". The thin client downloads "AutoUpdate.html" and runs it. The original thin client process would exit and leave Internet Explorer (or the default Web browser or application associated with ".html") displaying the contents of "AutoUpdate.html". "AutoUpdate.html" could include scripting code, redirect the Web browser to a Web site, or simply display an informational message to the user.

7.4.7 Automatic Update Failure

If the thin client automatic update process fails for any reason, a message box appears and a log file is created on the server by default. This file contains a log of the update operations and details about the failure. The configuration variables that affect the failure notification process are described in the following paragraphs.

The TC_AUTO_UPDATE_FAILED_TITLE configuration variable lets you set the title bar text for the message box that appears on update failure. Its default value is

```
ACUCOBOL-GT Thin Client
```

The TC_AUTO_UPDATE_FAILED_MESSAGE configuration variable lets you specify the text in this message box. Its default value is

```
ACUCOBOL-GT Thin Client: Automatic update was unsuccessful
```

In some cases, you may not want the thin client to inform the user that the automatic update failed. Set the TC_AUTO_UPDATE_NOTIFY_FAIL configuration variable to “false” (0, off, no) to configure the thin client so it does not notify the user if the automatic update fails. The default value of TC_AUTO_UPDATE_NOTIFY_FAIL is “true” (1, on, yes).

Note that the message box does not appear if the user cancels the Windows installer download process.

You can use the TC_SERVER_LOG_FILE configuration variable to configure the location and name of the log file created on the server. The name can optionally include the hostname of the client machine and the process ID of the server runtime that was managing the automatic update at the time of the failure.

By default, this file is named “autoupdate.%c.%p.log”, where *%c* is replaced by the client hostname and *%p* is replaced by the process ID of the server runtime. The default location is the working directory specified in the alias on the server.

The directory must exist at the time of the failure. Otherwise, this file is not created. The location of the log file is relative to the working directory specified in the alias. For example, the following setting:

```
TC_SERVER_LOG_FILE  updatefailed/%c.%p.log
```

configures the log files to be created in a directory named “updatefailed” relative to the working directory. The files are named with the hostname of the client followed by a dot and the process ID of the server runtime, ending with the “.log” file extension.

The client and server may be in different time zones. You can look at the file’s creation date to see what time the update failed according to the clock on the server, and view the timestamps at the beginning of the lines in the log file to see what time it failed according to the clock on the client.

Set the TC_DISABLE_SERVER_LOG configuration variable to “true” (1, on, yes) to prevent the creation of a log file for a thin client automatic update failure. The default value of TC_DISABLE_SERVER_LOG is “false” (0, off, no).

8 Managing the System

Key Topics

8.1 Introduction

This chapter discusses some system management issues to consider when you oversee AcuConnect® operations. Most UNIX system management functions are performed using the **acurcl** command described in section 8.2, “Managing the System: UNIX.” Windows users have two methods for performing management functions: the **acurcl** command or a graphical control panel, described in section 8.3, “Managing the System: Windows.”

This chapter also discusses machine failures, event logging, and troubleshooting for both the distributed processing and thin client environments.

8.2 Managing the System: UNIX

The **acurcl** command is your control and maintenance interface to AcuConnect. The **acurcl** command is used on the server at the command line (or with the graphical control panel under Windows) and initiates the following actions:

- starts and stops the AcuConnect program and installs and removes AcuConnect as a Windows service
- controls the configuration of running instances of AcuConnect
- reports AcuConnect system status
- creates and maintains the server access and server alias files

When the **acurcl** command is entered with no options, a list of options (or a graphical control panel in Windows) appears.

The **acurcl** command formats on UNIX servers are:

```
acurcl
acurcl –access
acurcl -alias
acurcl -config [server] [-n  port]
acurcl -info [server] [-n port]
acurcl -kill [server] [-f] [-n port] [-p PID]
acurcl -start [-c config_file] [-d] [-e error_file]
 [-l] [-t #] [-n port]
acurcl -version
```

In addition to these command formats, the following formats are available on Windows servers:

```
acurcl -install [server] [-depends ServiceDependency].
                [any valid start options]
acurcl -query [server] [-n port]
acurcl -remove [server] [-n port]
```

8.2.1 The acurcl Command

The following sections describe the **acurcl** command options in detail.

8.2.1.1 acurcl -access

The "**acurcl** -access" command starts the server access file manager utility. The access file manager is used to create and maintain the database of authorized AcuConnect clients and users. To use "-access" on a UNIX server, you must be logged in as root or superuser. On a Windows server, you must use "-access" from the Administrator account or from an account that belongs to the Administrators group. For a description of how to use this tool, see section 2.3.3, "The Server Access File."

8.2.1.2 acurcl -alias

The "**acurcl** -alias" command starts the server alias file manager utility. This utility is used to create and maintain the alias file required on the server only for thin client operations. This file holds all the information that is needed to invoke the appropriate runtime on the server. To use "-alias" on a UNIX server, you must be logged in as root or superuser. On a Windows server, you must use "-alias" from the Administrator account or from an account that belongs to the Administrators group. For a description of how to use this tool, see section 2.4, "Creating a Server Alias File in Thin Client."

8.2.1.3 acurcl -config

You can configure server configuration variables on the fly using the "**acurcl** -config" command. The "**acurcl** -config" command allows you to change the configuration of a running AcuConnect server, for instance, to turn tracing on or off or to change configuration variables such as

PASSWORD_ATTEMPTS. When you enter this option, you start communicating with the server running on the named server and named port. If no server name is given, the local server is assumed. If no port number is given, the default port is assumed.

Optional arguments to "-config" include:

Argument	**Description**
server	Specifies the name of the server machine to be configured. If no server is specified, the server is assumed to be the local host.
-n	Identifies a particular instance of the AcuConnect program by port number. The "-n" must be followed by a space and then an integer, for example, "6524". If no port number is specified, the default port is configured.

After you have entered the "-config" option, the prompt "rclcfg>" appears. At this prompt, enter any of the following commands:

GET *variable-name*	Gets the value of a single variable. For example: `rclcfg> get password_attempts` `PASSWORD_ATTEMPTS: 3`
SET *variable-name new-value*	Sets a variable in the server. For example: `rclcfg> set password_attempts 5` Note that if given an invalid variable name, the remote AcuConnect sets a variable of that name with the given value, even though that variable will not have any effect on the remote AcuConnect. For a list of valid server variables, refer to section 3.3, "Creating a Server Configuration File." Setting certain variables has no effect on the remote AcuConnect, since the variable is used at initialization time and never checked again. For example, trying to set ACURCL_PORT, SERVER_IP, or SERVER_NAME results in an error message such as: `rclcfg> set SERVER_IP 192.215.170.34` `Setting the SERVER_IP has no affect` `rclcfg> get SERVER_IP` `SERVER_IP: 192.215.170.34`

	Note that a GET command shows the new value, but it is not used.
LIST [{>|>>} *file*]	Lists the names and current values of all variables of which AcuConnect is aware, including some that do not directly affect AcuConnect. Use the optional "{>|>>} *file*" syntax to send the output to a named file instead of the screen. ">" creates a new file of the specified name, and ">>" appends the output to an existing file.
! cmd	Causes the command "cmd" to be executed.
HELP	Prints a quick synopsis of available commands. It looks very similar to the above list.
QUIT EXIT	These commands exit the configure mode.

8.2.1.4 acurcl -info

The "**acurcl** -info" command returns a report of AcuConnect system status for the current host (unless another machine is specified). This report includes all child runtimes that have not yet terminated, including the working directory the runtime was started from, the list of runtime arguments used to start the runtime, and the date and time that each child runtime was started (in HH:MM:SS.FF format, where *FF* is hundredths of a second).

The status report contains the following fields:

Field	**Description**
Client	Identifies the client machine associated with the child runtime.
User	Identifies the client user name associated with the child runtime.
PID	The process ID of the child runtime. Because these PIDs are local to the server machine, they are unique.
Directory	The working directory changed to before executing the child runtime.

Field	Description
Command	The full command line with which the child runtime was invoked. This will include all the options given to the runtime. This command line is not truncated and may wrap around on some screens, depending on how many options are used.

Optional arguments to “-info” include:

Argument	Description
server	Specifies the name of the server machine whose status is to be reported.
-n	Identifies a particular instance of the AcuConnect program by port number. The “-n” must be followed by a space and then an integer, for example, “6524”. If no port number is specified, then status for the default port is reported.

8.2.1.5 acurcl -install

The “**acurcl** -install” command is for Windows servers only. It installs AcuConnect as a Windows service. Optional arguments to “-install” include:

Argument	Description
server	Specifies the name of the server machine. If no server is specified, the server is assumed to be the local host.
depends	You can add Windows service dependencies to the installation of AcuConnect as a service. You may include multiple dependencies; simply use the “-depends” option for each one. The name of the service should be used as the argument following “depends” on the command line. For information on service dependencies, consult your Microsoft Windows documentation.

Argument	Description
any valid start options	Any valid "-start" option can be used for "-install". (Refer to "acurcl -start" below for a list of options.) Note that these options are stored for service start-up on this particular port. For example, to run the service with the arguments "-c c:\etc\server1.cfg -le c:\tmp\server1.log" on start-up, you would use the following "-install" command: `acurcl -install -c c:\etc\server1.cfg -le c:\tmp\server1.log` Installing a service on a particular port resets all start-up options for the service on that port. If no options are stored for a service, starting the server on a particular port automatically installs a service on that port and stores the options used. In other words, you can install and run a service with one set of arguments, and then occasionally run the service with different arguments by using "**acurcl** -start".

If you plan to have multiple instances of the AcuConnect service running at the same time, each instance will have a unique name. The name of an installed service will be "AcuConnect X", where *X* is the port number. If you do not use the "-n" option, the name of the service will be "AcuConnect".

In a thin client configuration, clients can be assigned to a particular instance of the AcuConnect service via the <server:port> notation of the "**acuthin**" command.

8.2.1.6 acurcl -kill

The "**acurcl** -kill" command causes the AcuConnect process to halt. If no server is specified, the AcuConnect process is halted on the current host; otherwise, the process is halted on the named host.

To use "-kill" on a UNIX server, you must be logged in as root or superuser. On a Windows server, you must use "-kill" from the Administrator account or from an account that belongs to the Administrators group.

Note that although files are not broken when you use this command, they may not be consistent. If a child runtime is terminated between two file updates that are both required for consistency, the runtime shuts down between them.

Unless the “-f” option is specified, AcuConnect prompts for confirmation before the halt action is executed.

```
Shutting down acurcl on: condor
Do you really want to shut down acurcl [N]?
```

Respond by entering “Y” or “N”.

Optional arguments to “-kill” include:

Argument	Description
server	Specifies the name of the server machine to kill. If no server is specified, the server is assumed to be the local host.
-f	Causes the AcuConnect process to terminate immediately, without prompting for confirmation. “-f” should be used when “**acurcl** -kill” is included in a program or script.
-n	Identifies a particular instance of the AcuConnect program by port number. The “-n” must be followed by a space and then an integer, for example, “6524”. If no port number is specified, the default port is terminated.
-p	Causes AcuConnect to terminate an individual child process. The “-p” must be followed by a space and the process ID of the child process to be terminated.

Caution: The AcuConnect process can also be terminated from the command line. For example, on UNIX you could use the command “kill -9” (signal #9). However, a signal #9 prevents AcuConnect from performing an orderly shutdown and should never be used when clients are actively using AcuConnect.

8.2.1.7 acurcl -query

The “**acurcl** -query” command is for Windows servers only. It allows you to query whether or not the specified Windows service is installed.

Optional arguments to “-query” include:

Argument	**Description**
server	Specifies the name of the server machine. If no server is specified, the server is assumed to be the local host.
-n	Identifies a particular instance of the AcuConnect program by port number. The “-n” must be followed by a space and then an integer, for example, “6524”. If no port number is specified, the default port (which has the service name “AcuConnect”) is removed.

When you enter this option, you receive a message stating whether or not the service is installed, and if it is, whether or not the service is currently running. The “-query” command also displays the start-up parameters that have been stored for the service, if any. For example:

```
C:> acurcl -query
Service AcuConnect (AcuConnect 8.0 on the default port
 (5632)) exists
Start Up: Automatic
Program: C:\AcucblGT\bin\acurcl.exe
Startup arguments: '-c c:\etc\server1.cfg -le
c:\tmp\server1.log'
Status: running
C:>
```

Note that the start-up arguments listed are the arguments used to install the service, and not the arguments given to the currently running server.

8.2.1.8 acurcl -remove

The “**acurcl** -remove” command is for Windows servers only. It shuts down AcuConnect (if it is not already shut down), removes the Windows service for AcuConnect, and removes the service’s stored start-up parameters.

Optional arguments to "-remove" include:

Argument	**Description**
server	Specifies the name of the server machine. If no server is specified, the server is assumed to be the local host.
-n	Identifies a particular instance of the AcuConnect program by port number. The "-n" must be followed by a space and then an integer, for example, "6524". If no port number is specified, the default port (which has the service name "AcuConnect") is removed. See the "acurcl -install" option above for more information about port numbers.

8.2.1.9 acurcl -start

The "**acurcl** -start" command starts AcuConnect. On Windows servers, this option also installs AcuConnect as a Windows service, if it is not already installed. In addition, if no start-up options are stored, "**acurcl** -start" stores the start-up options to be used as the default options.

To use "-start" on a UNIX server, you must be logged in as root or superuser. On a Windows server, you must use "-start" from the Administrator account or from an account that belongs to the Administrators group.

Unless the "-f" option is specified, AcuConnect starts in background. If AcuConnect is already running, it outputs the following message:

```
acurcl is already running on hostname
```

and a new AcuConnect process is not started. If you want to start AcuConnect with new options, you must stop and then restart it.

Note: On Windows NT, 2000 to 2008 systems, it is best to specify "**acurcl** -start" with no options. This causes it to use the start-up parameters that have been stored for the service. Typically, if you specify options with "**acurcl** -start", these options are in effect only until the AcuConnect service is started again. The exception to this is if no start-up parameters have been stored for the service. In this case, the parameters that you start AcuConnect with are stored for future start-up use.

A detailed description of valid arguments to the "-start" command can be found in section 2.6, "Starting AcuConnect."

8.2.1.10 acurcl -version

The "**acurcl** -version" command causes the version number of AcuConnect to be output. "-version" must be the only argument on the command line.

8.3 Managing the System: Windows

Windows administrators can manage AcuConnect through a graphical control panel. To access the control panel, enter the **acurcl** command on the server command line with no options. The control panel has five tabs, which correspond to the various actions available from the command line. You can access help from the graphical control panel by pressing the **F1** key, as well as by clicking the **Help** button.

Note: To use the Control Panel on Windows 2008 where User Access Control security is turned on (as it is by default), any user must choose "Run as Administrator" in order to use the various Acuserver utilities. UAC can be turned off, in which case the user must merely be a member of the administrators group in order to fully operate the control panel.

Please note that some of the tabs on the control panel are available only if you have Administrator privileges.

8.3.1 AcuConnect Control Panel

The AcuConnect control panel comprises five tabs:

- Access Tab
- Config Tab
- Alias Tab
- Info Tab
- Services Tab

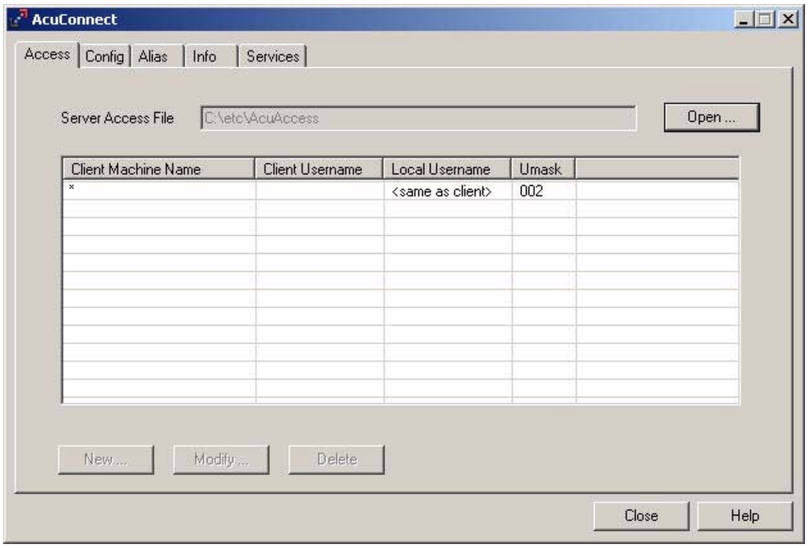
These tabs are described in detail in this section.

8.3.1.1 Access Tab

The Access tab, shown below, allows you to maintain the AcuAccess file through the access file manager. To use the access file manager from the control panel, you must be logged onto a Windows server from the Administrator account or from an account that belongs to the Administrators group. For detailed information about the AcuAccess file, refer to section 2.3.3, "The Server Access File."

You can use the Access tab to:

- create an access file
- add an access record
- remove an access record
- modify an access record
- display an access record



Server Access File: C:\etc\AcuAccess

Client Machine Name	Client Username	Local Username	Umask
*		<same as client>	002

Access Tab

The following table describes the contents of each of the fields on this tab.

Field	**Description**
Client Machine Name	The official host name by which the client machine is identified on the TCP/IP network. For distributed processing on Windows, this name is found in the Host field under Control Panel/TCP/IP Properties/DNS Configuration. An asterisk (*) in this field means "match all client machines."
Client Username	The user's login name on the client system. An empty field means "match all client users."
Local Username	The local user name that AcuConnect will use when fulfilling requests for the client user. An entry of "same as client" means "use the client username." When the string "same as client" is specified, certain conditions apply. If Client Username is not a valid name on the server, DEFAULT_USER is used. In addition, if the Local Username field is blank, DEFAULT_USER is used. If DEFAULT_USER is used to connect to AcuConnect on an NTFS partition under Windows NT, 2000, or 2003, 2008 be sure that DEFAULT_USER has both READ (RX) permissions on "wrun32.exe" and the appropriate permissions to access any file.
Umask	A three-digit file creation mask.

For an example of common access records, see section "Common access records example" in Chapter 2.

To create a new access file

1. On the Access tab, click **Open**.
2. Browse to the directory in which the new access file will reside (the default is c:\etc).
3. In the **File Name** text box, enter the name of your new access file and click **Open**.
4. Because this is a new file name, you will get a message stating:

   'C:\etc\filename' does not exist. Do you want to create it [N]?

Enter **Yes** to create a new access file.

The **New** button is now available for you to use to add records to the new access file.

To display a record in an access file

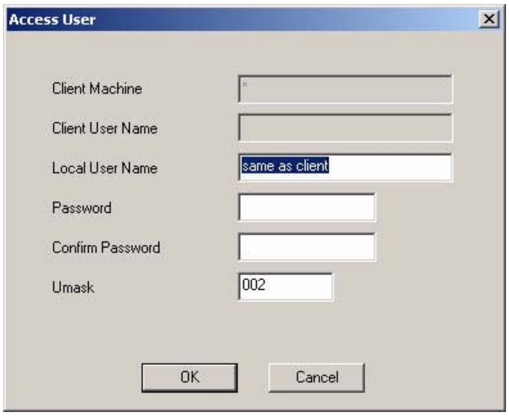
1. On the Access tab, click **Open**.
2. Browse to the access file that contains the record you wish to display.

   The records of the selected access file appear in the list box.

To add an access record to an access file

1. On the Access tab, click **New**.

   The Access User dialog appears, as shown below.



Client Machine: *

Local User Name: same as client

Umask: 002

Access User Dialog

2. In the **Client Machine** text box, enter the client machine identification name.
3. In the **Client User Name** text box, enter the user’s login name on the client system.
4. In the **Local User Name** text box, enter the local user name that AcuConnect will use when fulfilling requests for the client user.

5. In the **Password** text box, specify a password that must be supplied by requesters who match this record., then enter it again in the **Confirm Password** text box.
6. The **Umask** text box contains the default umask of the runtime process started for the user. You may change this to another valid value, if you wish.
7. Click **OK** to add the record to the access file.

To modify a record in an access file

1. On the Access tab, select from the list box the record you wish to modify.

   The Access User dialog appears, with the selected record's information in the **Client Machine** and **Client User Name** text boxes.
2. Select a text box to modify and enter the desired information.
3. When you have finished modifying the information in the text box(es), click **OK** to apply your changes.

To remove an access record from an access file

1. On the Access tab, select from the list box the record you wish to remove.

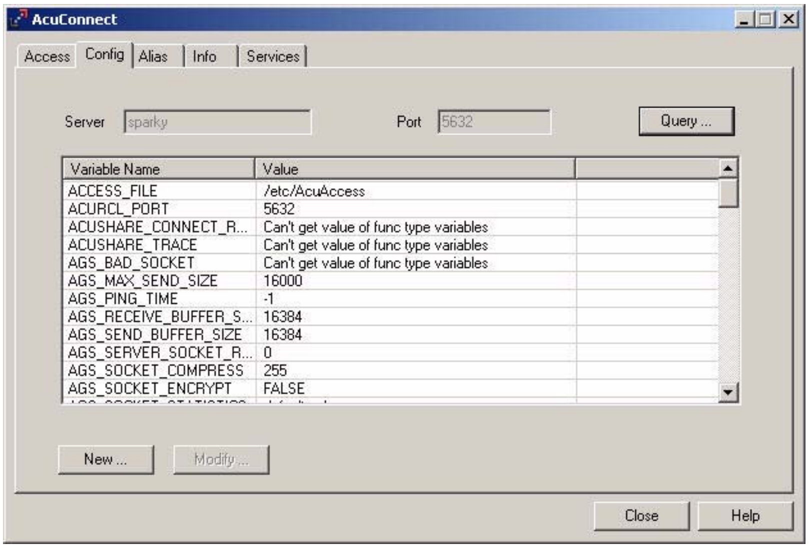
   Caution: Be certain that you want to delete the record you select—you will not be asked if you are sure, and the action cannot be undone.
2. Click **Delete** to remove the selected record.

8.3.1.2 Config Tab

The Config tab, shown below, allows you to view and set the status of the current configuration settings of a running server.

You can use the Config tab to:

- change the server and port
- add variables to the running server
- modify the values of variables



Server sparky

Port 5632

Variable Name	Value
ACCESS_FILE	/etc/AcuAccess
ACURCL_PORT	5632
ACUSHARE_CONNECT_R...	Can't get value of func type variables
ACUSHARE_TRACE	Can't get value of func type variables
AGS_BAD_SOCKET	Can't get value of func type variables
AGS_MAX_SEND_SIZE	16000
AGS_PING_TIME	-1
AGS_RECEIVE_BUFFER_S...	16384
AGS_SEND_BUFFER_SIZE	16384
AGS_SERVER_SOCKET_R...	0
AGS_SOCKET_COMPRESS	255
AGS_SOCKET_ENCRYPT	FALSE

Config Tab

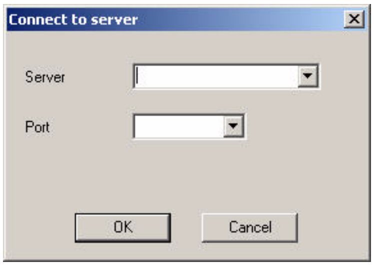
The list box on the Config tab displays the variables and their values associated with the server and port shown in the fields at the top of the tab. The following table describes the contents of each of the fields on this tab.

Field	**Description**
Variable Name	The name of a configuration variable that has been added to the server configuration file.
Value	The value that has been assigned the variable.

To change the server and port you are viewing

1. On the Config tab, click **Query**.

The Connect to server dialog appears, as shown below.



Connect to server Dialog

2. Enter or select from the **Server** drop-down list the name of the server you wish to view.
3. Enter or select from the **Port** drop-down list the name of the port you wish to view on the server.
4. Click **OK** to apply your selections.

   The information for the new server and port appears in the list box.

To add a variable to the running server

1. On the Config tab, click **New**.

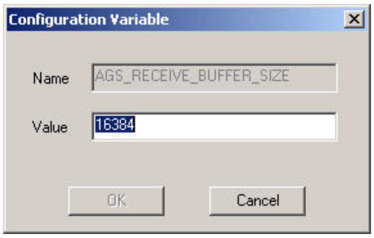
   The Configuration Variable dialog appears with blank fields.

2. In the **Name** field, enter the name of the variable you wish to add.
3. In the **Value** field, enter the value for the variable you added.
4. Click **OK** to add the variable to the server.

To modify the value of a variable

1. In the list box on the Config tab, select the variable you want to modify.
2. Click **Modify**.

The Configuration Variable dialog appears with the selected variable in the **Name** field and its value in the **Value** field, as shown below.



Name: AGS_RECEIVE_BUFFER_SIZE

Value: 16384

Configuration Variable Dialog

3. In the **Value** field, enter the value you want for the variable.
4. Click **OK** to apply the new value.

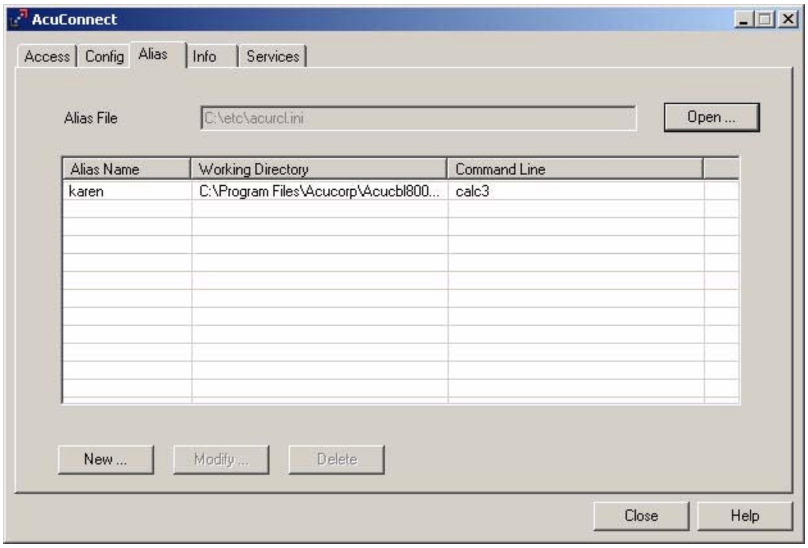
8.3.1.3 Alias Tab

Before using AcuConnect for thin client operations, you must create a single association file—an alias file—to hold all the information that will be needed to invoke the appropriate application on the server. To create thin client aliases from the control panel, you must be logged onto a Windows server from the Administrator account or from an account that belongs to the Administrators group.

The Alias tab, shown below, allows you to maintain AcuConnect thin client aliases.

You can use the Alias tab to:

- create an alias file
- modify an alias record
- delete an alias record



Alias File: C:\etc\acurcl.ini

Alias Name	Working Directory	Command Line
karen	C:\Program Files\Acucorp\Acucbl800...	calc3

Alias Tab

The following table describes the contents of each of the fields on this tab.

Field	**Description**
Alias Name	The name of an alias.
Working Directory	The default directory in which the runtime looks for all files specified on the alias command line.
Command Line	The full command line used to invoke the alias, including all the options given to the runtime.

Click **Open** to select an alias file to maintain. The information for the selected file appears in the list box.

To create an alias file

1. On the Alias tab, click **Open** and browse to the folder in which you want to save the file.
2. Enter a file name.

3. You will be told the file does not exist and asked if you want to create it. Click **Yes**.
4. Click **New** to add a new alias entry to the alias file.

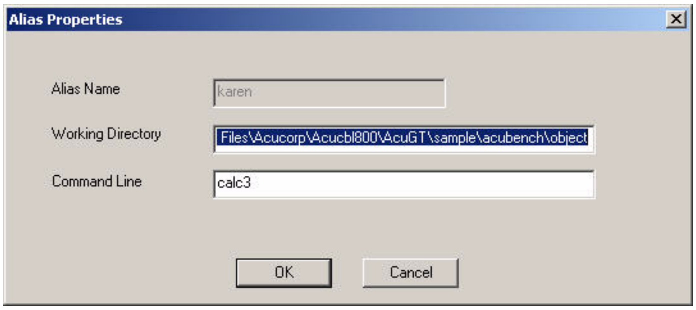
   The Alias Properties dialog appears with blank fields.
5. In the **Alias Name** text box, enter the name of the alias you want to add.

   For example, you might enter “reserve” for an airline reservation service.
6. In the **Working Directory** text box, enter the directory in which the COBOL object files for this alias can be found. (This must be an existing directory.)
7. In the **Command Line** text box, enter the full command line you want the runtime to use, including all options, to invoke the alias file.
8. Click **OK** to add this entry to the alias file.
9. Repeat steps 2–8 for each record that you want to add to the alias file.
10. Click **OK** to apply these options.

To modify an alias record

1. In the list box on the Alias tab, select the alias you want to modify. If you want to modify a record in a different alias file, open the alias file first.
2. Click **Modify**.

The Alias Properties dialog appears, displaying the properties of the selected alias, as shown below.



Alias Name: karen
Working Directory: Files\Acucorp\Acucbl800\AcuGT\sample\acubench\object
Command Line: calc3

Alias Properties Dialog

3. To change the properties of the alias’ working directory, enter the information you want to appear in the **Working Directory** text box.
4. To change the properties of the command line for the alias, enter the information you want to appear in the **Command Line** text box.
5. Click **OK** to apply these changes.

To delete an alias record

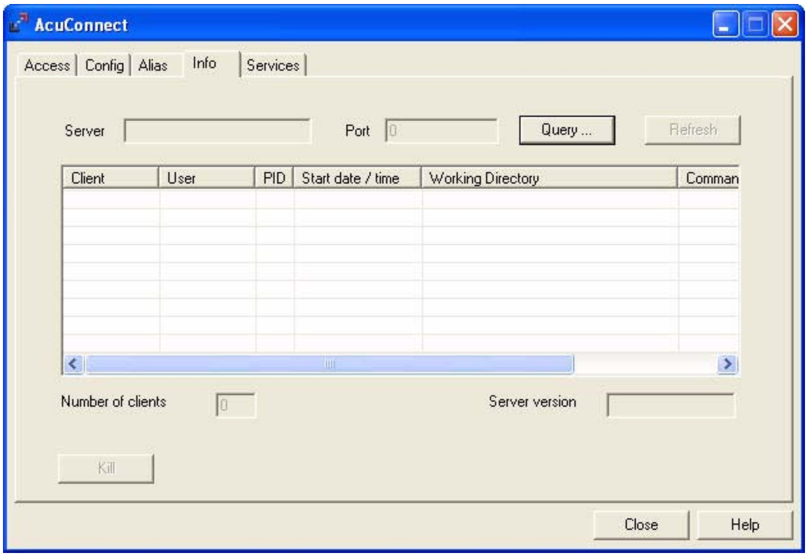
1. In the list box on the Alias tab, select the alias you want to delete.
2. Click **Delete**.

8.3.1.4 Info Tab

The Info tab, shown below, allows you to list and terminate child processes.

Use the Info tab to:

- list child processes
- update the list box display
- automatically update the list box display
- terminate child processes



Info Tab

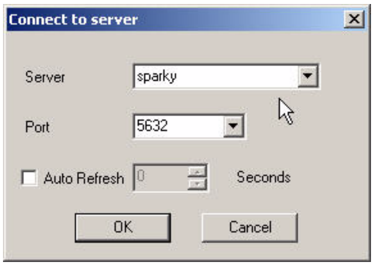
The following table describes the contents of each of the fields on this tab.

Field	**Description**
Client	The name of the client machine associated with the child runtime.
User	The client user name associated with the child runtime.
PID	The process ID (PID) of the child runtime.
Startdate/time	The date and time the runtime started.
Working Directory	The default directory in which the runtime looks for all files specified on the alias command line.
Command Line	The full command line used to invoke the child runtime, including all the options given to the runtime.

To list child processes

1. On the Info tab, click **Query**.

The Connect to server dialog appears, as shown below.



Server: sparky
Port: 5632
Auto Refresh: 0 Seconds

Connect to server Dialog

2. From the **Server** drop-down list, select a server for the child process.
3. From the **Port** drop-down list, select a port for the child process.
4. Click **OK** to apply your selections.

   The server you selected appears in the **Server** field, the port you selected appears in the **Port** field, and the child processes for the selected server/port combination are now listed in the Info tab list box.

To update the list box display

On the Info tab, click **Refresh** to update the information displayed.

To update the list box display automatically

1. On the Info tab, click **Query**.

   The Connect to server dialog appears, as shown above.

2. Check the **Auto Refresh** checkbox.
3. In the list box, specify the number of seconds between updates.
4. Click **OK** to apply your selections.

To terminate a child process

1. In the list box, select a child process to terminate (Kill).

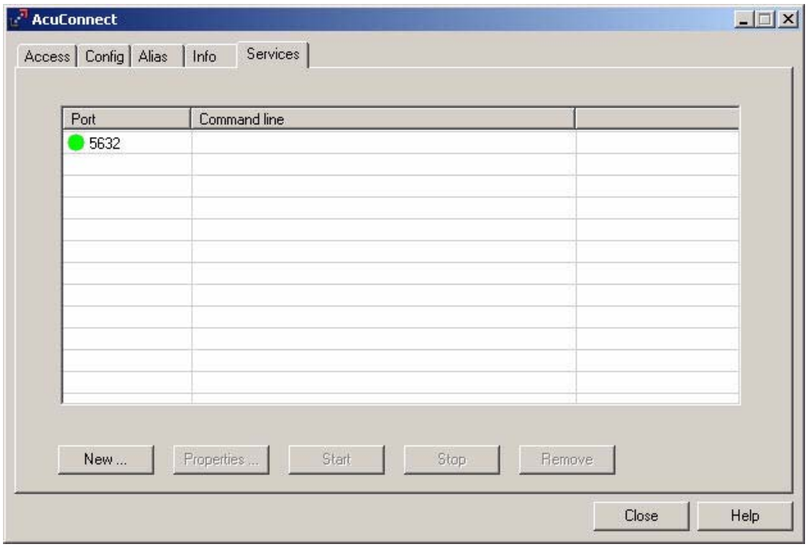
2. Click **Kill** to terminate the process.

8.3.1.5 Services Tab

The Services tab, shown below, allows you to add, start, stop, and remove services, as well as view or modify the properties of an existing service. When you choose this tab, services that are currently running are marked with a green bullet and services that are currently stopped are marked with a red bullet.

Use the Services tab to:

- add services
- start, stop, and remove services
- add dependencies



Port	Command line	
● 5632		

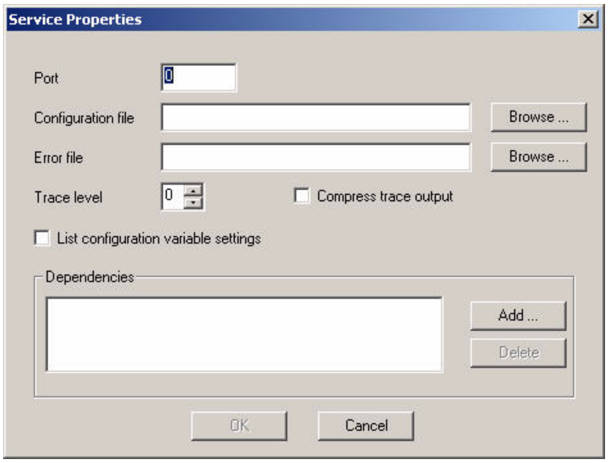
Services Tab

The following table describes the contents of each of the fields on this tab.

Field	**Description**
Port	The port number on which AcuConnect is running.
Command line	The full command line used to invoke the runtime, including all the options.

The Service Properties dialog

For most operations on the Services tab, the Service Properties dialog opens, as shown below.


Service Properties

Port: 0

Configuration file:

Error file:

Trace level: 0

☐ Compress trace output

☐ List configuration variable settings

Service Properties Dialog

You can perform the following actions in the Service Properties dialog:

Field	**Description**
Configuration file	Use **Browse** to specify the name and path of the configuration file for the server.
Error file	Use **Browse** to specify an error file for the server.

Field	Description
Trace level	Selecting a trace level from this drop-down list box turns on the tracing function. If you have specified an error file in the Error file field, trace information is placed in the named error file. Trace level can be “1” through “7”: “1” provides information about access file match attempts. The trace information buffer is flushed to the error file when the buffer is filled or the service terminates. “2” provides information about runtime requests. The buffer is flushed to the error file when the buffer is filled or the service terminates. “3” provides the information described for “1” and “2”. “5” is equivalent to “1”, but the tracing buffer is flushed to the error file each time an access file match is requested. (File trace flushing can also be controlled with the FILE_TRACE_FLUSH server configuration variable. See section 3.3, “Creating a Server Configuration File.”) “6” is equivalent to “2”, but the tracing buffer is flushed to the error file each time a runtime is requested. “7” provides the information described for “5” and “6”.
Compress trace output	Causes the file specified in Error file to be compressed using the gzip compression method.

Field	Description
List configuration variable settings	Selecting this checkbox causes a listing of the server configuration file to be printed to standard error output. This can be helpful when you are debugging problems that may be related to configuration variables. If you have specified an error file, the listing is captured in this file.
Dependencies	You can add the name of a Windows service or service group on which AcuConnect depends. (You may include multiple dependencies.) Click **Add** to display a list of possible services and service groups. Note that if you enter dependencies in this field, AcuConnect will not start unless each service named is currently installed on the server as a service. For information on service dependencies, consult your Microsoft Windows documentation.

To add a service

1. On the Services tab, click **New**.
2. In the Service Properties dialog, select the options you wish to add to your new service (see the options and their descriptions in the table above.)

To start, stop, or remove a service

1. On the Services tab, in the list box, select the service you wish to start, stop, or remove.
2. Perform one of the following actions:
   - To start a selected service, click **Start**.
   - To stop a selected service, click **Stop**.

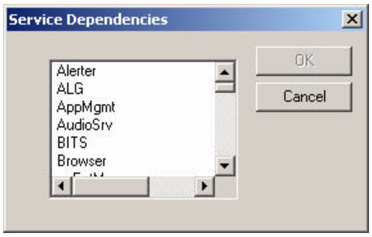
- To remove a selected service, click **Remove**.

Caution: Be certain that you in fact want to remove the record you select—you will not be asked if you are sure, and the action cannot be undone.

To add dependencies to a service

1. In the Service Properties dialog, click **Add** to the right of the **Dependencies** list.

   The Service Dependencies dialog appears, as shown below.



Service Dependencies Dialog

2. From the list, select a dependency you want to add to this service.
3. Click **OK** to add this selection.
4. Repeat steps 2 and 3 until you have added all of the dependencies you want for this service.
5. In the Service Properties dialog, click **OK** to apply the selected dependencies to the service.

8.4 Machine Failures

Two major concerns regarding client and server machine failures are:

- What happens to open files/programs when a server crashes?
- How are other clients and servers affected by a crash?

When a client application or connection is terminated with **Ctrl+C** or a **kill** command (other than a “kill -9”), AcuConnect detects the termination and closes all files/programs held open for that client process. However, other terminal software and hardware failures are not detected. Note that although files are not broken when you use this command, they may not be consistent. If a child runtime is terminated between two file updates that are both required for consistency, the runtime shuts down between them.

Should the server go down, all clients actively using AcuConnect get access errors when attempting to communicate with the server. Client systems must terminate their current applications and wait for the return of the server. All files that were open on the server at the time of the termination are left in an unknown state and may be corrupt. **If AcuConnect is automatically started when the server boots, AcuConnect should be immediately halted.** Before you start AcuConnect, you should check all files that might have been affected by the termination and, if necessary, rebuild them. After you have verified all files, you can start the **acurcl** daemon.

To view an AcuConnect system status report, issue the “**acurcl** -info” command. In Windows, you can use the graphical control panel’s Info tab to check the status of active child runtimes. Click **Query** to select a server/port combination and list the child processes running on the selected server.

8.5 Event Logging

AcuConnect can write to the Windows event notification system or the UNIX system log when information or error messages need to be reported to the System Administrator. Such information might be generated when an unexpected termination occurs or a broken file is detected. If your operating system supports it, the UNIX version of AcuConnect writes to the syslog facility. If your operating system does not support syslog, it writes this information to the console.

The Windows version can write to the event log of other machines in the network. Set the WINNT_EVENTLOG_DOMAIN configuration variable to the UNC name of the computer to which all event log messages should be sent. This variable must be set in the configuration file and is not changeable. Note that if this variable is not set, system logging information is sent to the local machine on which the server is executing.

You should be aware that some Windows administration expertise may be required to implement this feature on a complex Windows network. For example, you can start AcuConnect on one server and send event log information to a different server. If you are Administrator on one machine but not on the other machine, however, the log file is not created and no error message informs you of that fact.

8.6 AcuConnect Error Messages

In the course of operations, AcuConnect may produce error messages that inform you of system problems. Error messages that appear when AcuConnect cannot be started are seen on the server by the system administrator who is trying to start AcuConnect, or on a client by a user who is trying to establish a connection to AcuConnect and start a client runtime.

Error messages that appear when a remote program cannot be started via AcuConnect may be different depending on whether AcuConnect is deployed in a distributed processing or thin client environment. Refer to section 8.7.1 for distributed processing and section 8.8.1 for thin client environments for the various messages that are generated when a remote application cannot be started.

acurcl start-up errors

Messages generated as a result of the inability to start AcuConnect are different depending on the operating system. Two conditions can be responsible for the set of messages that appear when AcuConnect does not start: a missing or expired AcuConnect license file or a missing AcuAccess file.

In a UNIX/Linux environment, the following message may be produced when the license file is missing:

```
License file './acurcl.alc' inaccessible.
```

The following message indicates that the user license is expired:

```
This license for the AcuConnect product has expired.
 Please contact your software vendor for assistance.
```

For a missing server access file in a UNIX/Linux environment, the AcuConnect log file displays the following:

```
Access file '/etc/AcuAccess' not found.
```

From the graphical control panel in Windows, the following message may indicate that the AcuConnect license file is missing or expired, or that the server access file is missing:

```
The AcuConnect service failed to start.
 Invalid or missing license.
```

8.7 AcuConnect Distributed Processing: Troubleshooting

This section is a collection of error messages and step-by-step diagnostic procedures for finding and resolving common system problems in AcuConnect's distributed processing configuration. Many of these procedures require access to root or administrator privileges and are intended for use by your site's system administrator.

8.7.1 Error Messages

In distributed processing, AcuConnect generates the following set of error messages after an unsuccessful attempt to start a program via the AcuConnect remote listener.

```
Program missing or inaccessible.
```

This error message indicates that the program would not start, possibly because

- The called program cannot be found or permissions to execute the program do not exist.
- The ACUCONNECT_RUNTIME_FLAGS variable contains incorrect parameters, such as an invalid error file path.
- Permission to write to the server runtime error file does not exist.
- AcuConnect cannot find the server runtime.

- The server runtime license is expired or missing (Windows servers only).
- The called program's return code indicates a failed call.

```
Connection refused - perhaps acuconnect is not running.
 Unable to start the server runtime.
```

Possible explanations for this message include:

- AcuConnect is running on a port other than the one specified in the connection attempt.
- The server runtime license is expired or missing (UNIX/Linux servers only).
- The AcuAccess file does not allow access or it is missing.

The following message means the number of users allowed by the runtime or distributed processing license has been exceeded:

```
Connection refused - User count exceeded on remote
 server.
```

When the "acurcl.clc" license file is missing, the following message appears:

```
Invalid license or license not found for this
 connection type.
```

Various file status codes may also be returned to the client as error messages. Refer to Book 4, Appendix E, in the ACUCOBOL-GT® documentation set for information about file status messages.

Refer to the following sections for tips on resolving some of the system problems that can cause these error messages to appear.

8.7.2 Unexpected User Name

AcuConnect establishes a connection with the client, but uses an unexpected user name (Local Username). There are two common reasons for getting an unexpected Local Username on the server:

1. The client machine/client user combination doesn't match the expected access record.
2. The client machine/client user combination matches an access record that specifies an unexpected Local Username (perhaps the name of a group account, or an account with restricted privileges).

To investigate and correct this situation, you must be familiar with AcuConnect server access configuration and have access to root or Administrator privileges. We recommend that you work with your AcuConnect system administrator.

UNIX diagnostics

1. Confirm your client user name.

   Log on to the client system using the same user name and UNIX environment that resulted in the unexpected user name. At the UNIX prompt, enter:

   ```
   who am i
   ```

   Is the user name returned the name you expected?

2. Confirm the name of the client system. Enter:

   ```
   hostname
   ```

   The system will return its official network host name.

3. On the server, examine the server access file for the record that matches the client machine name/client user name combination. This requires root privileges on a UNIX server and Administrator group privileges on a Windows server. It should be done by the AcuConnect system administrator.

   Run the server access file manager utility:

   ```
   acurcl -access
   ```

   a. Be sure to enter the name of the working server access file when prompted.

   b. Select menu item [4], "Display one/all security records."

   c. Respond "N" to the "Display all records?" prompt.

d. To the next two prompts, provide the client machine name and client user name, respectively. The matching record will be displayed.

The AcuConnect system administrator should be able to determine whether this is the appropriate and expected access record for the client machine name/client user name combination and take any necessary steps to modify the record, or add a new one.

Windows diagnostics

1. Confirm the name of the client system.

   Under Windows XP, the runtime uses the host name that is set in the Control Panel.

   a. In the Control Panel/Network/Configuration menu, select TCP/IP.

   b. Then choose Properties/DNS Configuration/Host. The name you specify for the "Host" entry is the one that the runtime uses.

   c. Under Windows NT/2000/2003/2008, navigate to Start/Settings/Control Panel/Network/Protocols.

   d. Select TCP/IP, and then choose Properties/DNS/Host Name. The name you specify for the "Host Name" entry is the one that the runtime uses.

2. On the server, examine the server access file for the record that matches the client machine name/client user name combination. This requires root privileges on a UNIX server, and Administrator group privileges on a Windows server. It should be done by the AcuConnect system administrator.

   Run the server access file manager utility:

   ```
   acurcl -access
   ```

   a. Be sure to enter the name of the working server access file when prompted.

   b. Select menu item [4], "Display one/all security records."

   c. Respond "N" to the "Display all records?" prompt.

d. To the next two prompts, provide the client machine name and client user name, respectively. The matching record will be displayed.

The AcuConnect system administrator should be able to determine whether this is the appropriate and expected access record for the client machine name/client user name combination and take any necessary steps to modify the record, or add a new one.

8.7.3 Connection Refused

Attempts to connect to the server fail, possibly returning a 9D, 103, connection refused error. Connection refused errors occur for a variety of reasons:

1. The matching access record specifies an invalid Local Username, or DEFAULT_USER holds an invalid user name. Note that the use of “same as client” in the Local Username field of the access record can lead to the attempted use of an invalid user name (“same as client” directs AcuConnect to use the Client Username as the Local Username).
2. There is no matching access record for the client machine name/client user name combination.
3. If connecting to a UNIX server, the AcuConnect license on the server may require that the **acushare** license manager be running, and AcuConnect was not able to start **acushare** automatically.
4. If connecting to a Windows server, various limitations in Windows can prevent a child runtime from starting.

UNIX diagnostics

1. Confirm your client user name.

   Log on to the client system using the same user name and UNIX environment that resulted in the unexpected user name. At the UNIX prompt, enter:

   ```
   who am i
   ```

   Is the user name returned the name you expected?

2. Confirm the name of the client system. Enter:

   ```
   hostname
   ```

   The system will return its official network host name.

3. Examine the server access file for the record that matches the client machine name/client user name combination (this should be performed by the AcuConnect system administrator).

   To do this, become superuser or log on as root and run the server access file manager utility:

   ```
   acurcl -access
   ```

   a. Be sure to enter the name of the working server access file in response to the utility's first prompt.
   b. Select menu item [4], "Display one/all security records."
   c. Respond "N" to the "Display all records?" prompt.
   d. Respond to the next two prompts with the client machine name and client user name, respectively. The matching record will be displayed.

   If there is no matching entry, you need to add one.

   - If the Local Username field contains the name of a user (a string), check the UNIX password file (/etc/passwd) for the presence of a valid entry for that name. If no entry exists, the name is not valid.
   - If the Local Username field is "same as client", Local Username is set to the value of Client Username. Check the UNIX password file (/etc/passwd) for the presence of a valid entry for that name.
   - If the Local Username field is blank, Local Username is set to the value of the server configuration variable DEFAULT_USER. The value of DEFAULT_USER is defined in the server configuration file. Check the UNIX password file ("/etc/passwd") for the presence of a valid entry for that name.

4. Confirm that **acushare** is running on the UNIX server by issuing the "**acushare**" command with no arguments. If you receive a usage message, **acushare** is not running, and may be manually started using the "**acushare** -start" command.

5. Determine whether you have reached the Windows limit on the number of simultaneous child runtimes. The CHILD_WAIT configuration variable contains a value for the number of milliseconds that AcuConnect waits for a child runtime to start successfully. If the child runtime subsequently stops, check the trace file for the entry "Child runtime stopped!! Exit code is *nnn*." If *nnn* is "128", the Windows limit has been reached and no additional child runtimes are allowed until another runtime process is terminated. Please refer to Microsoft Knowledge Base article 184802, "User32.dll or Kernel32.dll Fails to Initialize," to learn how to work around this limitation. Refer to CHILD_WAIT in this manual for information about how to set the configuration variable.

Windows diagnostics

1. Confirm the name of the client system.

   Under Windows XP, the runtime uses the host name that is set in the Control Panel.

   a. In the Control Panel/Network/Configuration menu, select TCP/IP.

   b. Then choose Properties/DNS Configuration/Host. The name you specify for the "Host" entry is the one that the runtime uses.

   Under Windows NT/2000/2003/ 2008, navigate to Start/Settings/Control Panel/Network/Protocols.

   a. Select TCP/IP.

   b. Then choose Properties/DNS/Host Name. The name you specify for the "Host Name" entry is the one that the runtime uses.

2. Examine the server access file for the record that matches the client machine name/client user name combination (this should be performed by the AcuConnect system administrator).

   To do this, log in as Administrator or from an account that belongs to the Administrators group, then run the server access file manager utility ("**acurcl** -access").

   a. Be sure to enter the name of the *working* server access file in response to the utility's first prompt.

   b. Select menu item [4], "Display one/all security records."

c. Respond "no" ("N") to the "Display all records?" prompt.

d. Respond to the next two prompts with the client machine name and client user name, respectively. The matching record will be displayed.

If there is no matching entry, you need to add one.

- If the Local Username field contains the name of a user (a string), check the Windows server User Manager for the presence of a valid entry for that name. If no entry exists, the name is not valid.
- If the Local Username field is "same as client", Local Username is set to the value of Client Username. Check the Windows server User Manager for the presence of a valid entry for that name.
- If the Local Username field is blank, Local Username is set to the value of the server configuration variable DEFAULT_USER. The value of DEFAULT_USER is defined in the server configuration file. Check the Windows server User Manager for the presence of a valid entry for that name.

3. Confirm that **acushare** is running on the UNIX server by issuing the "**acushare**" command with no arguments. If you receive a usage message, **acushare** is not running, and may be manually started using the "**acushare** -start" command.

4. Determine whether you have reached the Windows limit on the number of simultaneous child runtimes. The CHILD_WAIT configuration variable contains a value for the number of milliseconds that AcuConnect waits for a child runtime to start successfully. If the child runtime subsequently stops, check the trace file for the entry "Child runtime stopped!! Exit code is *nnn*." If *nnn* is 128, the Windows limit has been reached and no additional child runtimes are allowed until another runtime process is terminated. Please refer to Microsoft Knowledge Base article 184802, "User32.dll or Kernel32.dll Fails to Initialize," to learn how to work around this limitation. Refer to CHILD_WAIT in this manual for information about how to set the configuration variable.

8.7.4 Invalid Password

Attempts to connect to the server fail with an invalid password error:

```
Invalid connection password specified by client
```

Invalid password errors occur for two reasons:

1. The password supplied via interactive input does not match the password in the access record.
2. The password supplied by the application via the *acu_client_password* variable does not match the password in the access record.

Because password transactions involving *acu_client_password* are invisible to the user, the cause of these failures is not immediately obvious. If you are getting the error message given above, but you're not being prompted for a password, the application is using *acu_client_password*. The value of *acu_client_password* is set when the application is installed. (Note that the value of *acu_client_password* should terminate with LOW-VALUES.) Refer to section "Option 1: Program Variable" for more information about *acu_client_password*.

Diagnostics

1. If the error occurred after a password prompt, work with your AcuConnect system administrator to establish a new password or to have the password requirement removed.
2. If no password prompt occurred, ask the application's technical support group to change or remove the *acu_client_password*. Contact your application vendor.

8.7.5 AcuConnect Fails to Start

AcuConnect fails to start.

1. Verify that Windows and UNIX permissions are set correctly on the files. See section 2.3, "Establishing System Security."
2. In Windows, verify the status of the service. To do this, enter:

   ```
   sc query acurcl
   ```

   The system should display the current state of AcuConnect and should say "RUNNING". If the service is *not* running, the system will return an error message similar to the following:

```
[sc]EnumQueryServiceStatus:OpenService failed 1060
```

Another method for verifying the status of the service is to select:

Start/Settings/Control Panel/Services

and then check the AcuConnect service status. You should see "Started" under "Status". You may also check the Event Viewer for more information. Select these options:

Start/Programs/Administrative Tools/Event Viewer/Log/Application

3. Examine AcuConnect service events to determine if there were any errors at startup.

8.7.6 Problems Starting and Stopping Services

You have problems starting or stopping the services or receive a message that the service has stopped when it has not.

- A service started using the Server Manager interface should be stopped using the same interface.
- A service started on the command line should be stopped using the command line.

Diagnostics

Be sure to stop services the same way you start them.

8.8 AcuConnect Thin Client: Troubleshooting

Many things can affect the way the server application and the ACUCOBOL-GT Thin Client interact. For example, computers can crash, networks can get interrupted, or other errors can occur. This section outlines some areas you should be aware of as you use our Thin Client technology. (Limitations and restrictions involved in running your application in the thin client are described in section 4.4.1, “Limitations in Thin Client Environments.”) The following paragraphs also include some frequently asked questions.

8.8.1 Error Messages

Error messages are produced in a thin client environment as a result of the following basic scenarios: AcuConnect cannot be started, a server connection cannot be made, or a remote application cannot be started via AcuConnect. Refer to section 8.6 for the messages generated when AcuConnect cannot be started. The following paragraphs describe the messages in the other two categories.

If you try to connect to a Windows server, limitations in Windows can prevent a child runtime from starting, resulting in a “Connection failed” error message. Section 8.7.3, “Connection Refused,” contains information for determining whether you have reached this Windows limit and how to work around it.

When a server connection cannot be established, possibly because the socket on the server is busy, the following message appears:

```
Connection failed. General socket error.
```

The following message also appears as a result of a failed server connection:

```
Connection failed. Connection to server refused.
```

Specific reasons for this error message include a missing or invalid license file (“acurcl.alc”), an AcuAccess file that is missing or not allowing access for this particular user, or the use of a port number that is not available for some reason.

A non-existent alias file or an alias file that does not contain the requested alias causes the following message to appear:

```
Connection failed. Alias does not exist.
```

If you are running on VMS and you specify an invalid working directory in your alias file, you receive the following error message:

```
Connection failed. Unknown error.
```

If you receive the following message, check to make sure you have a valid error file path and name in the alias definition:

```
Connection failed. Server failed to start remote
  process. Check messages on the server.
```

Inability to start the specified remote application results in the following message:

```
prog-name: Program missing or inaccessible.
```

where *prog-name* is the name of the called program. This message may be the result of

- an alias record that contains an invalid working directory.
- a called program that may not exist in the working directory.

The following message indicates that the maximum number of runtime users in thin client has been exceeded:

```
You have exceeded the licensed number of users for
  ACUCOBOL-GT. If you would like to add users, please
  contact your customer service representative.
```

8.8.2 Tuning System Performance

The nature of our Thin Client technology may make some functions expensive in terms of system performance. Certain adjustments have been made to minimize the amount of message traffic between the client and the server. For example, if a program resizes a window to be larger than the available screen space, the operation succeeds, because checking for failure is an expensive operation.

Frequent message traffic as a result of communications between screen controls on the display host and an application on the server can negatively affect thin client performance. To identify operations that may be causing performance issues for the thin client, you can establish a screen trace to monitor network message traffic on a slow screen. You can start with a trace of a frequently used screen if it is not clear which screen is causing your performance issue.

One way to improve system performance is to recompile your programs to the Version 5.2 or later object format. In older object files that use MODIFY to set control properties frequently, the thin client must return status to the runtime after each MODIFY statement that sets a property value. When you recompile, thin client sends status back to your program only if the program uses that status value.

The following sections describe other specific issues in thin client system performance and tips for handling these situations.

8.8.2.1 Buffer size considerations

Configuration variables that let you adjust internal network buffer sizes can help to improve thin client performance by increasing the amount of data that travels across the network at any one time.

The AGS_RECEIVE_BUFFER_SIZE and AGS_SEND_BUFFER_SIZE variables control the internal network buffers in the low-level socket routines. The ACUCOBOL-GT thin client and runtime call these routines to send data across the network. Every time data is sent across the network, it must be acknowledged from the other side. So increasing the size of these buffers may enhance performance, because sending more data at a time means fewer acknowledgement messages. Making these buffers larger has the most impact when a large amount of data is sent back and forth. Note that the buffer size won't change the performance characteristics in situations where data must be sent immediately.

The AGS_MAX_SEND_SIZE variable can work in tandem with the two BUFFER_SIZE variables mentioned above to help with performance. This variable works best if its value matches the BUFFER_SIZE values. For example, if AGS_MAX_SEND_SIZE is 8000 bytes and AGS_SEND_BUFFER_SIZE is 16000 bytes, the send buffer does not fill completely before the socket layer decides that data should be sent.

8.8.2.2 File compression

File compression is another factor to consider in system performance, and the AGS_SOCKET_COMPRESS variable can help address this issue. If the speed of your computer is much higher than the speed of the network, then the "ZLIB" setting for this variable may work well for you. However, if the latency caused by compression time is higher than the latency in the network, you may not gain much. RUNLENGTH compression is faster, but much less is compressed.

As an example, suppose you need to send 10000 bytes over a network that can transfer 1000 bytes per second. With no compression, the data can be sent in 10 seconds. If ZLIB compression can compress that 10000 bytes to 1000 bytes in 20 seconds, then total time for this operation is 21 seconds (20 seconds to compress plus 1 to send). In this case, no compression is the better option. However, if ZLIB can compress that 10000 bytes to 2000 bytes in 2 seconds, then the total time with ZLIB is 4 seconds (2 seconds to compress plus 2 to send). In this situation, the compression option is the better choice.

Note that Windows supports ZLIB compression, but not all UNIX machines do. If ZLIB compression is not supported on a particular machine, a variable value of “ZLIB” will be ignored.

8.8.2.3 TC_CONTROL_SYNC_LEVEL runtime configuration variable

Normal runtime behavior ensures that VALUE data items in a Screen section are updated whenever a BEFORE, AFTER, or EXCEPTION procedure executes. This can result in decreased performance under the thin client. You can determine which VALUE data items are updated by the setting of the TC_CONTROL_SYNC_LEVEL configuration variable. Limiting updates can mean improved performance. For more information about this configuration variable, refer to TC_CONTROL_SYNC_LEVEL.

8.8.2.4 TC_TV_SELCHANGING configuration variable

The tree view control normally generates a Msg-Tv-Selchanging event in some unexpected circumstances. For example, this event is generated when a tree view becomes active, even if the selection in the tree view is not changing. The frequency with which these events are passed between the client and the server can cause performance issues.

The TC_TV_SELCHANGING configuration variable lets you control how often Msg-Tv-Selchanging events are generated in a thin client environment. Different values for this variable enable you to suppress the generation of this message event to varying degrees. For more information about this configuration variable, refer to TC_TV_SELCHANGING.

8.8.2.5 Graphical control event handling

Three runtime configuration variables give you some control over which events your application receives. Letting the thin client filter out some events can reduce network traffic and enhance performance. TC_EVENT_LIST and TC_AX_EVENT_LIST can contain the numeric value of an event type or a list of event types separated by non-numeric characters. TC_AX_EVENT_LIST is for .NET or ActiveX event types. If the value of TC_EXCLUDE_EVENT_LIST is "1", the specified events are not sent to your program. For more information about these variables, refer to section 3.4, "Creating a Runtime Configuration File for the Remote Server Component."

Three common control properties perform the same functions for individual graphical controls. EVENT-LIST and AX-EVENT-LIST properties allow you to specify a list of event types to either send or block from the program, depending on the value of the EXCLUDE-EVENT-LIST property. By default, listed events are sent to the program. For detailed information about these properties, refer to section 6.4.9, "Common Screen Options," in the *ACUCOBOL-GT Reference Manual*.

One advantage to using the configuration variables for these functions is that you don't have to change your application code. For example, if program performance is affected by a single ActiveX control type or events that have unique type numbers, these are easily specified in a configuration file. Using the configuration variables is also an easy way to test whether this feature can improve system performance.

8.8.2.6 Screen Section table handling

The compiler produces some internal data items to hold intermediate values when a Screen Section USING, TO, or FROM data item is indexed or reference modified. In the thin client, this behavior is extended so that it uses the same intermediate item when both the FROM and TO items refer to the same data item, consuming slightly less memory. As a result, a program with a Screen Section that stores its values in tables may be able to use the performance benefit of a lower TC_CONTROL_SYNC_LEVEL configuration setting. For more information about this configuration variable, refer to TC_CONTROL_SYNC_LEVEL.

8.8.2.7 Grid control

Because the performance of some controls may be affected in a thin client environment, some control properties may need to be changed. The grid control experiences performance delays when the user clicks and drags the mouse to highlight several cells in the grid. To avoid this, the thin client does not generate the MSG-GOTO-CELL-DRAG event. This prevents you from highlighting a region of the grid during drag operations via the REGION-COLOR property.

The grid control supports a DRAG-COLOR property, which allows you to do highlighting during a drag operation. You can use this with both the thin client and with the stand-alone runtime. You replace REGION-COLOR references with DRAG-COLOR and remove any MSG-GOTO-CELL-DRAG handling. Note that REGION-COLOR is often located in the MSG-GOTO-CELL-DRAG handling code, and it will need to be moved when you change it to DRAG-COLOR. Most commonly, you can place the DRAG-COLOR reference in the code (or Screen section) where the grid is created.

For more information about the DRAG-COLOR property, refer to the grid control description in Book 2, *ACUCOBOL-GT User Interface Programming*, of the ACUCOBOL-GT documentation set.

8.8.2.8 Bitmaps

A program that uses bitmaps can be slower than one that does not. The first time a bitmap is displayed, it must be transferred from the server to the client. The bitmap is then placed in a memory cache on the client so that it doesn't have to be downloaded again. However, network messages travel between client and server regardless of whether the bitmap has already been downloaded to the client.

Bitmap download performance can be enhanced via the W$BITMAP library routine WBITMAP-LOAD process. Setting the WBITMAP-LOAD parameter FLAGS to WBITMAP-NO-DOWNLOAD optimizes bitmap handling. When WBITMAP-NO-DOWNLOAD is used, the server assumes that the bitmap is already in the client's cache directory and that the client can successfully load the bitmap. The server need not wait for a result code from the client that the bitmap is successfully loaded. For more information about W$BITMAP and the WBITMAP-LOAD function, refer to *ACUCOBOL-GT Appendices*, Appendix I.

Using ".jpg" files instead of ".bmp" files can also improve system performance, because ".jpg" files are compressed and therefore are smaller and transfer more quickly over the network.

8.8.2.9 Multiline entry fields

A multiline entry field with a compressed size larger than the value of AGS_MAX_SEND_SIZE (see AGS_BAD_SOCKET) may experience degraded performance in thin client. Although the default value of AGS_MAX_SEND_SIZE should be adequate in most situations, large multiline entry field performance may improve if the value is increased from its default of 16000 characters. Note that if compression is enabled, it is performed before the calculation of maximum packet size.

8.8.3 Connecting to AcuServer

It is possible that the ACUCOBOL-GT runtime may appear to hang if the application host needs to open files on a remote (third) machine using AcuServer® and a password is needed. The problem may appear in one of two ways.

1. If the runtime is on a UNIX machine and the remote file is an error file or a configuration file, the runtime and the ACUCOBOL-GT Thin Client both hang. Note that in this instance, if the remote file is requested after the COBOL program has started, the password dialog appears as expected.
2. If the runtime is running on a Windows machine, the runtime and the ACUCOBOL-GT Thin Client both hang.

You can avoid this behavior by not requiring passwords on machines running AcuServer if the files on the AcuServer machine are needed by the ACUCOBOL-GT runtime serving the thin client.

8.8.4 Frequently Asked Questions

Question: **Can AcuConnect load object files on mapped server drives?**

Answer: AcuConnect starts a new instance of the ACUCOBOL-GT runtime for each client request for an application to be executed. This runtime can locate object files on a Windows server mapped drive only if the drive mapping has occurred at an administrator level at system startup.

To load object files on a Windows server mapped drive in this way, you must ensure that the SECURITY_METHOD configuration variable defines the type of Windows security to be used for the connection. For information about the SECURITY_METHOD variable, refer to SECURITY_METHOD.

Question: **Is the Windows print spooler supported in the thin client?**

Answer: Generally speaking, all functionality of the Windows print spooler is supported except the following WIN$PRINTER functions:

WINPRINT_GET_SETTINGS_SIZE
WINPRINT_GET_SETTINGS
WINPRINT_SET_SETTINGS

These functions are not supported because of variations in memory allocated by data types on the client and the server (for example, a 64-bit UNIX server versus a 32-bit Windows client). Instead, you can make calls to WINPRINT_GET_PRINTER_INFO_EX and WINPRINT_SET_PRINTER, which are similar.

For information about these functions, refer to section 4.4.3.3.

Question: **I am using ACCEPT user-id FROM SYSTEM-INFO to return details about the user that is running a specific copy of the application, but the returned user ID is "root". How can I ensure that the correct user information is returned?**

Answer: To return the actual user ID of the user who is executing the application on a UNIX server, AcuConnect must be started automatically at server startup or by a user that logs into the server as root and then logs out of the system. When a subsequent client user starts an application that executes ACCEPT user-id FROM SYSTEM-INFO, the correct user ID is returned.

To return the correct user ID when a program is running on a Windows server, AcuConnect must be started as a Windows service and SECURITY_METHOD must be set to LOGON.

Information about starting AcuConnect automatically at system startup or as a Windows service can be found in section 2.6.5.

Question: **I have an application that contains branching code, which is used when the application is executed on either UNIX or Windows servers. The code relies on information that is returned about the operating system by ACCEPT FROM SYSTEM-INFO.**

However, with the thin client, the information returned is from the server side of the configuration. How can I obtain information about the client side, so that the application is executed using the correct client-side information?

Answer: ACCEPT FROM SYSTEM-INFO returns server-side information for the thin client environment. To obtain client-side information, you should modify your existing code to perform an ACCEPT FROM TERMINAL-INFO.

Question: **If a thin client application is launched from a Web browser, does that application have Web browser access? We may want to include a hyperlink on the screen.**

Answer: The thin client does not run in a Web browser. However, you could use the Web browser control inside your application to allow Web browser access, or include your links on the same Web page that launches the application.

Question: **Are there any restrictions regarding the use of ActiveX or COM objects?**

Answer: You can use ActiveX controls. However, they must be installed on the client machine. (Refer to section 6.3, “Installing ActiveX Files.”) Note that some controls may generate too many messages to be usable over the Internet.

COM and Distributed COM objects may be used with Thin Client.

Index