User's Guide

AcuServer

Version 8.1

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

E-01-UG-080901-AcuServer 8.1

Contents

Chapter 1: Introduction

1.1 Overview 1-2
1.2 Implementation Details 1-4
1.2.1 Remote File Access Performance 1-4
1.2.2 Record Locking 1-6
1.2.3 System Security 1-7
1.2.4 System Load Balancing 1-8
1.2.5 Server Licensing 1-8
1.2.6 Vision Files 1-9
1.2.7 International Character Handling 1-10
1.3 Technical Services 1-11

Chapter 2: Preparing Your UNIX Network

2.1 Getting Started on UNIX 2-2
2.2 UNIX System Requirements 2-5
2.3 Installing a UNIX Server 2-6
2.4 AcuServer System Security 2-7
2.4.1 Ownerships and Permissions 2-7
2.4.2 File Access Security 2-9
2.5 Installing the Client 2-10
2.5.1 Passwords for Clients 2-10
2.5.2 Setting Up the Host Name 2-11
2.5.3 Setting Up the User Name 2-12
2.5.4 Confirming Network Services 2-12

Chapter 3: Preparing Your Windows Network

3.1 Getting Started in Windows 3-2
3.2 Windows Installation Requirements 3-4
3.3 Installing a Windows NT/2000/2003/2008 Server 3-6
3.4 Setting Up Accounts 3-8
3.5 AcuServer System Security 3-9
3.5.1 Setting Ownerships and Permissions 3-9
3.5.2 File Access Security 3-10
3.5.3 Setting Permissions on New Files 3-11
3.6 Installing the Client 3-12

3.6.1 Passwords for Clients....3-12
3.6.2 Setting Up the Host Name....3-12
3.6.3 Setting Up the User Name....3-13
3.6.4 Confirming Network Services....3-14
3.7 Using AcuServer's Graphical User Interface for Windows....3-14

Chapter 4: Configuration

4.1 Configuring AcuServer....4-2
4.2 Configuration Variables....4-2
4.2.1 Runtime Configuration Variables....4-3
4.2.1.1 AGS_PING_TIME....4-3
4.2.1.2 AGS_SOCKET_COMPRESS....4-5
4.2.1.3 AGS_SOCKET_ENCRYPT....4-5
4.2.1.4 CACHE_DIRECTORY....4-6
4.2.1.5 CACHE_DIRECTORY_SIZE....4-6
4.2.1.6 *filename*_MRC....4-7
4.2.1.7 USE_LOCAL_SERVER....4-7
4.2.2 Server Configuration Variables....4-8
4.2.2.1 ACCESS_FILE....4-11
4.2.2.2 ACUSERVER_MASTER_SERVER....4-11
4.2.2.3 AGS_SERVER_SOCKET_RESERVE....4-12
4.2.2.4 COUNT_STATISTICS....4-13
4.2.2.5 DEAD_CLIENT_TIMEOUT....4-13
4.2.2.6 DEFAULT_TIMEOUT....4-14
4.2.2.7 DEFAULT_UMASK....4-14
4.2.2.8 DEFAULT_USER....4-15
4.2.2.9 ENCRYPTION_SEED....4-15
4.2.2.10 FILE_TRACE....4-15
4.2.2.11 FILE_TRACE_FLUSH....4-15
4.2.2.12 FILE_TRACE_TIMESTAMP....4-16
4.2.2.13 *filename*_DATA_FMT....4-16
4.2.2.14 *filename*_INDEX_FMT....4-18
4.2.2.15 *filename*_VERSION....4-19
4.2.2.16 LOCK_ALL_FILES....4-20
4.2.2.17 LOCKS_PER_FILE, MAX_FILES, and MAX_LOCKS....4-20
4.2.2.18 MAX_ERROR_LINES....4-21
4.2.2.19 MULTIPLE_RECORD_COUNT....4-22
4.2.2.20 NO_LOCAL_CACHE....4-22
4.2.2.21 PASSWORD_ATTEMPTS....4-22
4.2.2.22 PROVIDE_PASSWORD_MESSAGES....4-23
4.2.2.23 SECURITY_METHOD....4-23
4.2.2.24 SERVER_IP and SERVER_NAME....4-25

4.2.2.25 SERVER_PORT 4-26
4.2.2.26 TEXT 4-27
4.2.2.27 USE_SYSTEM_RESTRICTIONS 4-28
4.2.2.28 V_BASENAME_TRANSLATION 4-28
4.2.2.29 V_BUFFERS 4-28
4.2.2.30 V_BUFFER_DATA 4-28
4.2.2.31 V_INDEX_BLOCK_PERCENT 4-29
4.2.2.32 V_READ_AHEAD 4-29
4.2.2.33 V_STRIP_DOT_EXTENSION 4-30
4.2.2.34 V_SEG_SIZE 4-30
4.2.2.35 V_VERSION 4-30
4.2.2.36 WINNT-EVENTLOG-DOMAIN 4-31
4.2.2.37 WINNT-LOGON-DOMAIN 4-31

Chapter 5: Administrator Utilities and Functions

5.1 Overview 5-2
5.2 acuserve Command-line Options and Formats 5-3
5.3 AcuServer Control Panel interface 5-4
5.4 Creating and Maintaining the Server Access File 5-5
5.5 Installing and Removing acuserve as a Windows Service 5-5
5.5.1 Installing from the Command Line 5-6
5.5.2 Installing from the Acuserver Control Panel 5-8
5.5.3 Removing the acuserve Service 5-9
5.5.4 Determining if acuserve is installed as a Service 5-10
5.6 Starting and Stopping acuserve 5-11
5.6.1 Starting from the command line 5-11
5.6.2 Starting on UNIX 5-15
5.6.3 Stopping from the Command Line 5-15
5.6.4 Starting and Stopping from the AcuServer Control Panel 5-16
5.7 Changing acuserve Properties 5-17
5.7.1 Changing Properties from the Command Line 5-17
5.7.2 Changing Properties from the AcuServer Control Panel 5-19
5.8 Checking System Status 5-19
5.8.1 Generating Status Reports from the Command Line 5-20
5.8.2 Checking Status from the AcuServer Control Panel 5-21
5.9 Tracking Server Statistics 5-22
5.10 Closing Stranded Files 5-26
5.10.1 Closing Files from the Command Line 5-26
5.10.2 Closing Files from the AcuServer Control Panel 5-31
5.11 Checking Version Information 5-31
5.12 Registering Servers 5-32

Chapter 6: System Security

6.1 Security Overview....6-2
6.2 The Server Access File....6-4
6.3 Access Records....6-6
6.4 Using the Access File Manager....6-9
6.4.1 Creating or Opening an Access File....6-10
6.4.1.1 Creating and Opening from the Command Line....6-10
6.4.1.2 Creating and Opening from the AcuServer Control Panel....6-11
6.4.2 Adding an Access Record....6-11
6.4.2.1 Adding a Record from the Command Line....6-13
6.4.2.2 Adding a Record from the AcuServer Control Panel....6-15
6.4.3 Removing an Access Record....6-15
6.4.4 Modifying an Access Record....6-15
6.4.4.1 Modifying a Record from the Command Line....6-16
6.4.4.2 Modifying a Record from the AcuServer Control Panel....6-16
6.4.5 Displaying an Access Record....6-17
6.4.6 Exiting the Access File Manager....6-18
6.5 AcuServer Connection Logic....6-18
6.5.1 Passwords....6-19
6.6 Encryption....6-21

Chapter 7: Programming for AcuServer

7.1 Programming Considerations....7-2
7.2 Accessing Remote Files....7-2
7.2.1 Remote Name Notation....7-3
7.2.2 Using FILE_PREFIX and CODE_PREFIX....7-3
7.2.3 Command-Line and Configuration File Remote Name Notation....7-6
7.2.4 Using Name Aliases....7-7
7.3 Server Name Management....7-8
7.4 Multiple-Record Mode....7-14
7.4.1 File Limitations....7-15
7.4.2 Other Considerations....7-16
7.5 Restrictions to Library Functions....7-18
7.5.1 C$COPY....7-18
7.5.2 C$FILEINFO....7-20
7.5.3 RENAME....7-20

Chapter 8: System Management

8.1 Machine Failures....8-2

8.2 Error and Status Codes 8-4
8.2.1 Error Logging 8-4
8.2.2 Event Logging 8-5
8.2.3 Start Status Codes 8-5
8.2.4 File Access Failures, “9D” Errors 8-6
8.3 Diagnosing Errors with C$PING 8-8
8.4 Troubleshooting 8-10
8.4.1 Ambiguous File Error 8-10
8.4.2 Connection Times Out 8-11
8.4.3 File Access Denied 8-12
8.4.4 Unexpected User Name 8-19
8.4.5 Connection Refused 8-22
8.4.6 Invalid Password 8-23
8.4.7 Client and Server Both Hang When Connecting 8-24
8.5 Frequently Asked Questions 8-25

Glossary of Terms

Index

1 Introduction

Key Topics

Overview 1-2
Implementation Details 1-4
Technical Services 1-11

1.1 Overview

Welcome to the AcuServer® file server, a client/server technology that provides remote file access services to ACUCOBOL-GT® applications running on UNIX and Windows TCP/IP-based networks. AcuServer is part of the ***extend***® family of solutions.

With AcuServer, your applications gain:

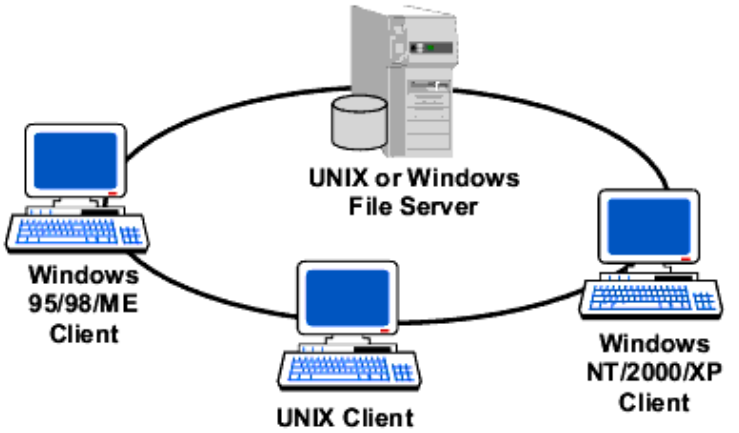
- the ability to create and store data files on any UNIX, Windows NT, Windows 2000 - 2008 server equipped with AcuServer.
- full function remote access from UNIX and Windows clients to all Vision, relative, sequential, and object files stored on an AcuServer server.
- full record locking support of all Vision and relative files.
- transparent access of remote and local files.

AcuServer does not require any changes to your existing application code. (Some applications that contain hard-coded paths to files must be modified to use the FILE_PREFIX or CODE_PREFIX environment variable, or to include the name of the file server in the path. A discussion of name strategies is included in **section 7.2, "Accessing Remote Files."**)

AcuServer does not require that you recompile your existing programs. Programs compiled with any version of ACUCOBOL-GT can be executed by any later version of ACUCOBOL-GT. ACUCOBOL-GT runtimes are available for most UNIX and Windows platforms. Contact our customer service representative for a current list of supported platforms.

*AcuServer can be combined with other **extend** technologies to provide access to data in networked, client/server, and distributed computing environments.* It can be used with the ACUCOBOL-GT Web Runtime to provide access to data over the World Wide Web (with permission). It can be used with ACUCOBOL-GT's CGI technologies to provide Internet access to data through an HTML front end. And it can be combined with AcuConnect® to facilitate file distribution and optimize network performance in a distributed computing setting.

Pictured below is a simple AcuServer network. ACUCOBOL-GT applications running on UNIX and Windows client systems access and store data files on a common UNIX, Windows NT, Windows 2000 - 2008 file server running AcuServer. More complex networks might incorporate multiple AcuServer file servers supporting dozens of client machines running multiple applications.



Simple AcuServer Network

Note: Your AcuServer version number and your ACUCOBOL-GT runtime version number should match. Programs compiled with earlier versions of ACUCOBOL-GT can be executed, provided that the version of ACUCOBOL-GT runtime and AcuServer are the same.

Unless otherwise indicated, the references to “Windows” in this manual denote the following 32-bit versions of the Windows operating systems: Windows XP, Windows NT 4.0 or later, Windows 2000, Windows 2003; and the following 64-bit version of the Windows operating system: Windows Server 2003 and 2008 x64. 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 (“Windows 2000,” “Windows NT 4.0,” and so on).

1.2 Implementation Details

AcuServer provides remote file access services through the use of a memory resident program (daemon or service) named **acuserve**, running on the file server. The client application executes using an ACUCOBOL-GT runtime. The runtime must be the same version as AcuServer.

The runtime recognizes access requests to remote files and uses socket calls to **acuserve** to fulfill the requests.

On the server, **acuserve** waits for file access requests, manages their execution, and returns the result to the client requester.

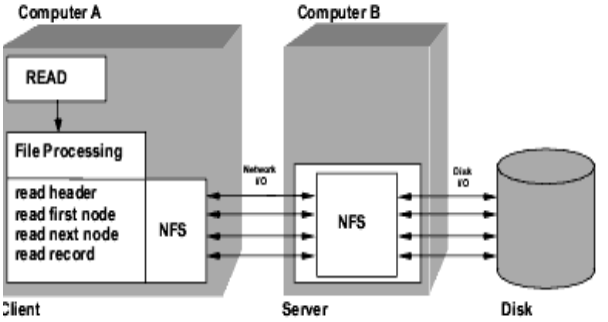
A typical AcuServer interaction might be:

1. An application running on a network machine attempts to READ a file.
2. The ACUCOBOL-GT runtime recognizes that the file to be read is located on a remote system and packages the request to **acuserve** on the file server.
3. **acuserve** receives the request, executes the READ, and returns the result to the client, completing the interaction.

1.2.1 Remote File Access Performance

AcuServer provides excellent network file access performance by using sockets as the principal communication protocol. Here's why.

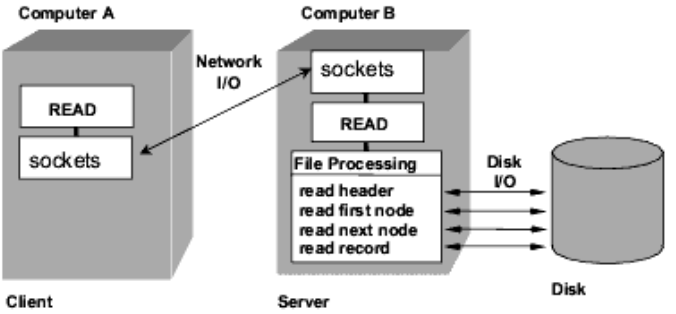
With the exception of a few unusual systems, remote file access always takes significantly more time than the same access made to a local file. It is not unusual for *network overhead* to add five fold or more to the time required to complete a common file action.



Processing a READ on a local file.

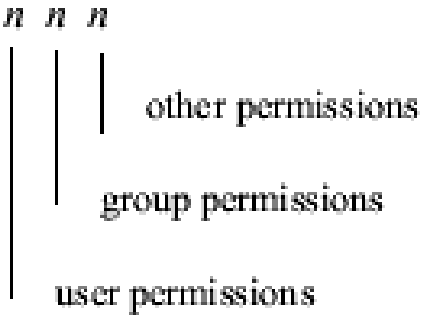
As the preceding drawing illustrates, a file access request (in this case a READ) can require multiple disk I/O actions. When these multiple disk actions are executed across the network, overhead is added to every I/O action.

Some network protocols, notably NFS (Network File System), require that each disk action be handled as a separate network transaction.



Processing a READ using NFS.

Using sockets, AcuServer executes file I/O requests with a single network transaction. In this way, AcuServer minimizes network overhead, resulting in superior performance.



Processing a READ using AcuServer.

Ordinarily, AcuServer returns a single record in each packet sent over the network. If you are reading a large number of sequential records without user interaction (for instance, if you are generating a report), you can optimize performance even further by using AcuServer's multiple-record mode. In this mode, AcuServer returns multiple records in each network packet. (By default, it returns ten records at a time, but this number is configurable.) Not all files are suited for multiple-record mode, however, and some limitations are imposed. Refer to **section 7.4, "Multiple-Record Mode,"** for more information on this mode.

1.2.2 Record Locking

Record locking support is exactly the same as that provided in the local environment. Any indexed or relative file opened in I/O mode can have records locked.

Three server configuration variables, **LOCKS_PER_FILE, MAX_FILES, and MAX_LOCKS**, are used to tune AcuServer record locking support.

1.2.3 System Security

AcuServer includes a transparent layer of remote file access security. This security system allows you to configure AcuServer access to be as open or restrictive as suits your needs. AcuServer uses a site-configured *server access file* (database) to validate the access privileges of service requesters. Optional password protection allows you to assign each user an individual password, which the user must then supply to AcuServer when making an initial service request.

On a Windows NT, Windows 2000 - 2008 server, AcuServer system security is designed to work with files that reside on an NTFS (NT file system). (AcuServer can work with a FAT file system, but the files are less secure.) For NTFSs, you may set read and write access permissions on your files by using the Windows security features. Please refer to your Windows documentation for more information about NTFSs and security procedures.

In addition, whether you are on UNIX or Windows, you have the option of implementing native system security rather than AcuServer system security. This option is described in more detail in **section 6.1, "Security Overview."**

Encryption

AcuServer offers support for data encryption. For any given service requestor (client process), you can instruct AcuServer to encrypt all data exchanged with that requestor. The encryption option and how to enable it is described in detail in **section 6.6, "Encryption."**

Note: The use of encryption can impose a significant performance cost. The performance cost is determined by the quantity of data being exchanged, the speed and bandwidth of the network, and the computational power of both the client and AcuServer host machines. If you plan to use encryption, we strongly recommend that you test and benchmark your application with encryption enabled prior to deployment to ensure that performance meets your requirements.

1.2.4 System Load Balancing

For large networks, you can help to optimize performance by obtaining a multiple-server license for AcuServer. This license enables you to start two or more instances of the **acuserve** daemon or service on the same server machine simultaneously. This gives you two options to help you balance the network load:

- You can assign each client runtime to a specific instance of the daemon.
- You can designate one instance of the **acuserve** daemon as a master (primary) server, which distributes the load among multiple additional instances of the daemon (secondary servers). This process is described in **section 4.2.2.2, "ACUSERVER_MASTER_SERVER."**

1.2.5 Server Licensing

AcuServer 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 encoded with the maximum number of **acuserve** instances allowed for your site.

To confirm the number of instances supported on your server, log onto the server and enter:

```
acuserve  -v
```

If you require support for additional **acuserve** instances, you can easily upgrade your license by contacting your Micro Focus ***extend*** sales representative.

If you are deploying your application with the ACUCOBOL-GT Web Runtime (an ActiveX component) and will be accessing your data via AcuServer, special licensing considerations apply. Specifically, unless your users have a local or network runtime license file installed, you must have a multiple-user ACUCOBOL-GT runtime license file on the server. You need to have a license file that can accommodate the total number of concurrent remote users that you anticipate. (If you anticipate 100 concurrent users, you need a 100-user runtime license file on the server in addition to the

AcuServer license file.) The runtime license file must be located in the same directory or folder as the **acuserve** executable and must be named "runcbl.alc" (on Windows systems, this requires that you *make a copy* of the Windows runtime license file, "wrun32.alc", and name it "runcbl.alc".) When an application running in the Web Runtime makes a service request, AcuServer notices that the request is from a Web Runtime connection and attempts to secure an authorization for the connection from the runtime license file. When a Web Runtime user disconnects from AcuServer by exiting the COBOL program, the authorization is released and made available to subsequent users. If the number of Web Runtime users exceeds the number of users allowed by the license file, the COBOL program receives a file status code 9D,105 and the following message is output to AcuServer's error log.

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

Refer to *A Programmer's Guide to the Internet* for more information on using the ACUCOBOL-GT Web Runtime with AcuServer.

Note: Nothing in this document is intended to amend the terms and conditions of the applicable license agreement between you and Micro Foucs. Rather, this document is meant to summarize the various aspects of our licensing technology, which is required to operate AcuServer. The terms and conditions of your licensing of AcuServer shall continue to be governed by the applicable license agreement between you and Micro Focus.

1.2.6 Vision Files

The Vision file system is the default file system used with ACUCOBOL-GT for all environments other than VMS.

By default, AcuServer Version 8.0 creates Vision Version 5 files. (AcuServer can also be used to build and access Vision Version 3 and 4 files.) Version 5 files are generated in a dual file format, with data records stored in one segment and overhead key information stored in another.

To cause AcuServer to build Vision Version 3 or 4 files, see the configuration variable V_VERSION in **section 4.2.2, "Server Configuration Variables."**

1.2.7 International Character Handling

International character translation is supported in AcuServer environments. The translation process is handled by the client runtime, which uses an XFD (eXtended File Descriptor) and a user-defined map file to accomplish the translation. A detailed description of international character handling is located in Chapter 5, section 5.4, of the *ACUCOBOL-GT User's Guide.*

To take advantage of international character mapping support you must:

- Create an XFD for each data file that contains fields that you want to have translated. Use the "-Fx" or "-Fa" compile option to create an XFD file (see Chapter 2 of the *ACUCOBOL-GT User's Guide*). Note that only fields of type CHARACTER can be translated.

- Create a map file. The map file specifies the values that characters are translated into before being passed to the server or translated back from before being passed to the program. The map file can map any character value in the range 0–255. Map files are described in detail in section 5.4 of the *ACUCOBOL-GT User's Guide*.

- On the client, set the DEFAULT_MAP_FILE or *server*_MAP_FILE configuration variable to point to your map file (see Appendix H of Book 4 of the ACUCOBOL-GT documentation set).

1.3 Technical Services

You can reach Technical Services in the United States Monday through Friday from 6:00 a.m. to 5:00 p.m. Pacific time, excluding holidays. You can also raise and manage product issues online and follow the progress of the issue or post additional information directly through the website. Following is our contact information:

Phone:	+1 858.795.1902
Phone:	800.399.7220 (in the USA and Canada)
Fax:	+1 858.795.1965
E-mail:	**support@microfocus.com**
Online:	**http://supportline.microfocus.com**

For worldwide technical support information, please visit **http://supportline.microfocus.com**.

2 Preparing Your UNIX Network

Key Topics

Getting Started on UNIX 2-2
UNIX System Requirements 2-5
Installing a UNIX Server 2-6
AcuServer System Security 2-7
Installing the Client 2-10

2.1 Getting Started on UNIX

There are three basic steps to using AcuServer® file server software in a UNIX environment.

1. **Install AcuServer**. If you have not already installed AcuServer on your UNIX server, please refer to **section 2.2** and **section 2.3** of this chapter for a list of installation requirements and procedures. If AcuServer is already installed, proceed to the next step. There is nothing on the AcuServer distribution media to install on the client machine. However, you should ensure that every client system that will use AcuServer has a licensed copy of an ACUCOBOL-GT® runtime, the same version as AcuServer. You may need to set up client passwords, user names, and host names. This is described in **section 2.5, "Installing the Client."**

2. **Configure the AcuServer system**. AcuServer system configuration consists of:

   - *Assigning values to the runtime configuration variables*. With the exception of the FILE_PREFIX and possibly CODE_PREFIX variables (discussed later in this chapter), none of the runtime configuration variables requires modification. For information about runtime configuration variables, see **section 4.2.1, "Runtime Configuration Variables."**

   - *Assigning values to the AcuServer configuration variables*. None of the server configuration variables requires modification; however, you may want to modify them to gain control over or initiate certain functions like file locking, multiple-record mode, or error trace flushing. If you want to implement UNIX security rather than AcuServer system security, you must set the **SECURITY_METHOD** variable to "LOGON" in both the runtime and server configuration files. For information about server configuration variables, see **section 4.2.2, "Server Configuration Variables."**

   - *Creating the server access file*. AcuServer file access security is managed by a site-configured access file called the "server access file". You must create a server access file (default name "/etc/AcuAccess") before AcuServer will start or establish connections

with clients. Step-by-step instructions are included in **section 6.4.1, "Creating or Opening an Access File."** General information about the file is included in **section 6.2, "The Server Access File."**

- *Assigning and verifying UNIX ownerships and permissions on AcuServer's executable, access, and configuration files, as well as existing data files and directories.* AcuServer will not run on a UNIX server unless proper ownerships and permissions are set. Setting ownerships and permissions requires *root* privileges. Use the UNIX utilities **chown** and **chmod** to set ownerships and permissions, as described in **section 2.4.1, "Ownerships and Permissions."**

- *Modifying your runtime configuration file or application code to use remote name notation*. To use AcuServer, your applications must use remote name notation to reference files located on the server. The ACUCOBOL-GT runtime looks for remote name notation to identify requests to AcuServer. Remote name notation has the format "@server-name:path-name". You may add a remote path to the FILE_PREFIX or CODE_PREFIX configuration variables. Alternatively, you can define file name aliases in the runtime configuration file. A file name alias is a string that will replace the literal name in the ASSIGN TO clause of a SELECT statement. For more information on remote name notation, please refer to **section 7.2, "Accessing Remote Files."**

3. **Issue AcuServer commands**. AcuServer services are handled by the **acuserve** daemon running on the server. The **acuserve** command can be invoked from the command line to start and stop AcuServer (the **acuserve** daemon), retrieve AcuServer operation status, unlock stranded files, and create and maintain the server access file. For complete details, see **Chapter 5, "Administrator Utilities and Functions."**

Please be aware that the configuration of AcuServer system security is very important to safeguarding your data files and network computers. We urge you to read **Chapter 6, "System Security,"** before placing AcuServer into open service.

Running AcuServer on HP MPE/iX Systems

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

AcuServer runs as *root* and checks permissions for each access on UNIX. On Windows, it uses "Impersonate LoggedOnUser()" to assume the correct user ID.

In the MPE/iX environment, the operating system does not provide a way for a program (in this case **acuserve**) to change its user ID. Therefore, the daemon always uses the ID of the account that started **acuserve**. Any action **acuserve** 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 AcuServer.

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

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

2.2 UNIX System Requirements

For UNIX platforms, AcuServer software is shipped on CD-ROM, 4 mm Dat, 8 mm tape, or cartridge tape in TAR format.

To install and use AcuServer:

- All servers must be networked to clients with TCP/IP. (TCP/IP is not sold or supplied by Micro Focus.)
- Unless you have an unlimited license for AcuServer, all UNIX servers must run the current version of **acushare** (see section 2.11 of the *ACUCOBOL-GT User's Guide*), which is included on the AcuServer distribution media.
- All servers must have a copy of the license file activated by the product installation script. This file is named "acuserve.alc".
- Servers being accessed by the ACUCOBOL-GT Web Runtime must have a multiple-user ACUCOBOL-GT runtime license that accommodates each concurrent user that is anticipated. (If you anticipate 100 concurrent users of the Web Runtime, you need a 100-user runtime license on the server in addition to the AcuServer license file.) Alternatively, runtime users can install a local or network floating license for the runtime themselves.

What is the license file?

AcuServer is shipped with a product code and product key that define 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, AcuServer must be able to locate the license file to function.

AcuServer searches for the license file in the same directory as the executable, "**acuserve**". The license file is named "acuserve.alc."

If you move the AcuServer 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, AcuServer exits with an error message.

What is acushare?

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

2.3 Installing a UNIX Server

To install AcuServer on a UNIX server:

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

2. Change directory ("cd") into the target directory and extract the AcuServer software as described on the *Getting Started* 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 "etc/a_srvcfg" into the "/etc" directory. This may require *root* or *superuser* privileges. The server configuration file does not need to reside in its default location, but if the file is located elsewhere, the name and location of the configuration file must be specified with the "-c" option each time **acuserve** is started. Regardless of where the server configuration file is located, the file must not be writeable by anyone other than *root* for AcuServer to function. For more information about server configuration, see **section 5.6, "Starting and Stopping acuserve."**

AcuServer is now fully installed and ready for configuration on the UNIX server. See **section 4.1, "Configuring AcuServer,"** for the settings that can be configured for the server.

Note: If you do not have a network card in your machine, or if your network card is not working properly, AcuServer will fail to install and run.

2.4 AcuServer System Security

To establish a secure and functional AcuServer system, it is important to restrict access to the **acuserve** executable, server configuration files, and server access files.

In addition, you will need to set appropriately restrictive ownerships and permissions for your existing data files and directories.

2.4.1 Ownerships and Permissions

Proper ownerships and permissions on the **acuserve** executable file, server configuration files, server access files, and existing data files and directories are essential to establishing a secure and functional AcuServer system.

UNIX settings

Setting ownerships and permissions requires *root* privileges on UNIX systems. Use the commands **chown**, **chgrp**, and **chmod** to set ownerships and permissions. For details regarding UNIX file permission masks and the use of **chown**, **chgrp**, and **chmod**, see your UNIX operating system manuals.

UNIX ownerships and permissions on AcuServer files

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

FILE NAME	OWNER	PERMISSIONS
acuserve (executable file)	root	755
AcuAccess (and server access files having other names)	root	600
a_srvcfg (and server configuration files having other names)	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 (though more restrictive permissions could prevent some AcuServer users from using some AcuServer functions. For example, if the **acuserve** executable file were assigned permissions of 700, no user other than *root* could execute the "**acuserve** -info" command to generate a report of current AcuServer system status).

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

Ownerships and permissions on new files

When a client application makes its initial request to AcuServer for services, the requester is validated for permission to use AcuServer. If the requester is permitted to use AcuServer, a user name is assigned to the requester based on the Local Username field of the matching server access record (see **section 6.5, "AcuServer Connection Logic"**). Files created for that requester by AcuServer get the *user* and *group* ownerships of the assigned Local Username.

umask

The read and write permissions set on new files are determined by the umask specified in the matching server access record (because all files created are data files, execute permission is not applicable).

On UNIX servers, 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, AcuServer 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 about umask, see your UNIX operating system documentation.

2.4.2 File Access Security

AcuServer allows users of UNIX networks to choose between the system security that AcuServer offers and the security features offered by their own operating system. By default, AcuServer system security is used to govern file access. (This is discussed in **Chapter 6**.)

If you use AcuServer security, the AcuAccess and AcuAccess.vix files are used to establish user access privileges. Remember that these two files must be readable and writable *only* by Administrator and System, or AcuServer will not start.

Note: If you experience a problem with file access, it can be helpful to give the users Full Control of the files and directories they need to use. After everything is working smoothly, reduce access to Read-Only if desired. However, be sure to test each program to make sure that everything continues to function as you expect with the reduced privileges.

If you prefer to use UNIX security mechanisms, you may do so by setting the configuration variable **SECURITY_METHOD** to "LOGON" in both the runtime configuration file and server configuration file. For more information about the SECURITY_METHOD configuration variable, see **section 4.2.2, "Server Configuration Variables."**

Note that if you decide to use UNIX security instead of AcuServer system security, your users will have whatever permissions the system administrator has set up for them.

2.5 Installing the Client

There is nothing on the AcuServer distribution media to install on the client machine. However, you should ensure that every client system that will use AcuServer has a licensed copy of an ACUCOBOL-GT runtime and that AcuServer and the runtime share the same version. To check a runtime's version, use the "-v" option on the client runtime command line. The runtime displays information similar to the following:

```
ACUCOBOL-GT runtime version 8.0
Serial number 1234
Licensed for 2 processes
AcuServer client
Vision version 5 file system
Copyright (c) 1988 - 2008, Micro Focus (IP) Ltd.
```

Look for the line "AcuServer client" and a runtime version number that matches the version of AcuServer that you are running. If these two criteria are not met, the client machine cannot access data using AcuServer.

2.5.1 Passwords for Clients

If you require a client password to access the server machine, you can prompt the user for this password and store it in the variable Acu_Client_Password, or you can let the runtime prompt for it. Be sure to refer to **section 6.5.1, "Passwords,"** if you plan to employ user passwords.

2.5.2 Setting Up the Host Name

The runtime system on each client machine passes the client host name to the server machine.

Windows 2000/2003/XP/Vista

For Windows 2000 or 2003 and Windows XP clients, right-click on the "My Computer" icon on the desktop and select "Properties" from the pop-up menu.

- On a Windows 2000 or 2003 client, select the Network Identification tab of the System Properties window.
- On a Windows XP or Vista client, select the Computer Name tab of the System Properties window.

In most instances, the computer name listed in the "Full computer name" field is the host name used by the runtime.

You can also verify the host name by typing the command "hostname" at the Windows command prompt.

Windows NT

1. On Windows NT clients, open the Network folder in the Control Panel and select "Protocols".
2. Select TCP/IP and click **Properties**.
3. Select the DNS tab and find the "Host Name" entry.

   The name you specify for "Host Name" is the one that the runtime uses.

UNIX

On UNIX clients, the runtime uses the host name that is in the file "/etc/hosts".

2.5.3 Setting Up the User Name

For Windows clients logged into a Windows NT network, the runtime uses the NT domain user name that the user logged in with. For Windows clients that are not logged into a Windows NT network, the runtime uses the user name value that is set by the environment variable USERNAME. If USERNAME is not set, then the runtime uses the value that is set by the environment variable USER. (The values assigned to these variables are case-sensitive. Be sure that the case used in the AcuAccess file matches the case of the value set in the variable.) If neither of these environment variables is set, then the runtime uses the name "USER".

The **acuserve** service is usually run as the "SYSTEM" account. If you need to change the account used by the **acuserve** service, be sure that the new account has all of the appropriate permissions for a file server.

For UNIX clients, the runtime uses the user name that the user logged in with.

2.5.4 Confirming Network Services

On UNIX and Windows clients, confirm that your TCP/IP software is loaded and running.

If desired, use "ping" to confirm that the server is up and responding to the network. From your client system, type:

```
ping hostname
```

Ping should return:

```
hostname is alive
```

Alternatively, you can use our AcuPing program, which is found in the Start\Programs\Acucorp 8.*x*.*x*\AcuServer menu. AcuPing is discussed in **section 8.3, "Diagnosing Errors with C$PING."**

3 Preparing Your Windows Network

Key Topics

Getting Started in Windows 3-2
Windows Installation Requirements 3-4
Installing a Windows NT/2000/2003/2008 Server 3-6
Setting Up Accounts 3-8
AcuServer System Security 3-9
Installing the Client 3-12
Using AcuServer's Graphical User Interface for Windows 3-14

3.1 Getting Started in Windows

There are six basic steps to using AcuServer® file server software in a Windows environment.

1. **Install AcuServer.** If you have not already installed AcuServer on your Windows NT, or Windows 2000 - 2008 server, please refer to **section 3.2** and **section 3.3** of this chapter for a list of installation requirements and procedures. If AcuServer is already installed, proceed to the next step.

   There is nothing on the AcuServer distribution media to install on the client machine. However, you should ensure that every client system that will use AcuServer has a licensed copy of an ACUCOBOL-GT® runtime, and you may need to set up client passwords, user names, and host names. This is described in **section 3.6, "Installing the Client."**

2. **Set up user and/or group accounts.** Decide how users will access resources on the server machine (individual accounts or group accounts). **Section 3.4, "Setting Up Accounts,"** shows the tradeoffs. Create the user accounts and grant user rights, including access to the server from the network.

3. **Edit the AcuAccess file on the server.** AcuServer will not start if the "server access file" cannot be found. By default, this file is named "AcuAccess" and is located in the "c:\etc" directory. (If your server's operating system is located on a drive other than "c," AcuServer will look for the AcuAccess file in the "\etc" directory on the drive containing the operating system.)

   Step-by-step instructions for configuring the access file are included in **section 6.4.1, "Creating or Opening an Access File."** General information about the file is included in **section 6.2, "The Server Access File."**

   - If you choose to implement AcuServer system security, each record in the server access file should contain client machine name, client username, local username, and umask data. Passwords may also be defined if desired.

- If you choose to implement Windows security (recommended) rather than AcuServer system security, the access records need only contain client machine, username, and if using the LOGON option, password data.

Note: On Windows 2008 it is essentially required that you use the operating system's security methods. This is mainly due to a new security layer called User Account Control (UAC). Consult Windows Help for information on how UAC impacts file and program access.

If the access file is not owned by Administrator or the Administrators group, or is writable by users that do not have Administrator privileges, AcuServer will not start.

4. **Create or modify the directory structure that will be used by AcuServer clients.** Ensure that user accounts have FULL CONTROL access to the directory containing the data and object files. If files already exist, modify the permissions for each file to give the users FULL CONTROL access.

5. **Configure the AcuServer system.** AcuServer system configuration consists of:

   - *Assigning values to the runtime configuration variables.* With the exception of the FILE_PREFIX and possibly CODE_PREFIX variables (discussed later in this chapter), none of the runtime configuration variables requires modification. For information about runtime configuration variables, see **section 4.2.1, "Runtime Configuration Variables."**

   - *Assigning values to the AcuServer configuration variables.* Most of the server configuration variables require no modification; however, you may want to modify them to gain control over or initiate certain functions like file locking, multiple-record mode, or error trace flushing. If you want to implement Windows security rather than AcuServer system security, you must set the **SECURITY_METHOD** variable in both the runtime and server configuration files. If running on Windows 2008, using Windows security is essentially required. For information about server configuration variables, see **section 4.2.2, "Server Configuration Variables."**

- *Modifying your runtime configuration file or application code to use remote name notation.* To use AcuServer, your applications must use remote name notation to refer to files located on the server. The ACUCOBOL-GT runtime looks for remote name notation to identify requests to AcuServer. On Windows NT/2000-2008 servers, the format for this notation is "@servername:drive-letter:\pathname". You may add a remote path to the FILE_PREFIX or CODE_PREFIX configuration variables. Alternatively, you can define file name aliases in the runtime configuration file. A file name alias is a string that will replace the literal name in the ASSIGN TO clause of a SELECT statement. For more information on remote name notation, please refer to **section 7.2, "Accessing Remote Files."**

6. **Issue AcuServer commands.** AcuServer requests are handled by the **acuserve** service running on the server. You can use the Windows AcuServer Control Panel or the command line to start and stop AcuServer (the **acuserve** service), retrieve AcuServer operation status, unlock stranded files, and create and maintain the server access file. For details about the AcuServer Control Panel and the **acuserve** command, see **Chapter 5, "Administrator Utilities and Functions."**

Note: To use the AcuServer Control Panel on Windows 2008 where UAC 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.

Please be aware that configuration of AcuServer system security is very important to safeguarding your data files and network computers. We urge you to read the security information covered in **Chapter 6, "System Security,"** before placing AcuServer into open service.

3.2 Windows Installation Requirements

For Windows networks, AcuServer software is distributed via FTP or on CD-ROM.

To install and use AcuServer:

- All servers must be networked to clients with TCP/IP. (TCP/IP is not sold or supplied by Micro Focus).
- All servers must have a copy of the AcuServer license file activated by the product installation script. This file is named “acuserve.alc”.
- Servers being accessed by the ACUCOBOL-GT Web Runtime must have a multiple-user ACUCOBOL-GT runtime license that accommodates each concurrent plug-in user that is anticipated. (If you anticipate 100 concurrent users of the plug-in, you need a 100-user runtime license on the server in addition to the AcuServer license file.)
- Windows clients can run any TCP/IP software that uses a WINSOCK-compliant WSOCK32.DLL.
- Client machines must have the current runtime. Use the “-v” option at the runtime command line to verify runtime version.

What is the license file?

AcuServer 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. The license file is named “acuserve.alc”. After installation, AcuServer must be able to locate the license file in order to function. AcuServer searches for the license file in the same directory as the executable, “**acuserve**”.

If you move the AcuServer 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, AcuServer exits with an error message.

3.3 Installing a Windows NT/2000/2003/2008 Server

To install AcuServer on a Windows NT, Windows 2000 - 2008 server:

1. Install and configure TCP/IP before installing AcuServer. If you do not have a network card in your machine, or if your network card is not working properly, AcuServer will fail to install and run.

2. Log onto your server using the Administrator account or an account that belongs to the Administrators group. For Windows 2008, if you do not log in using the Administrator account you will need to use Windows elevated privileges to install AcuServer.

   Note: If you are using an earlier release of AcuServer, you can either install the new release while AcuServer is running or stop AcuServer and then install the new software. If you install the new release while AcuServer is running, you must reboot the machine before the new software will take effect.

   ---

3. Insert the product installation CD-ROM into your disk drive. If the installation program does not start automatically, click **Start**, select **Run**, and enter the path to the setup file. The path will be something like "*d*:\setup.exe", where *d* is the drive letter for your CD-ROM drive.

   Follow the instructions on the screen. When prompted to select a component to install, select AcuServer.

4. If you already have the files "\etc\AcuAccess" and "\etc\a_srvcfg", the **setup** utility will detect them and will ask you if you want to overwrite them.

Caution: Do not overwrite these files unless you have a backup copy.

   The "AcuAccess" file contains one access record that gives all users access to AcuServer. The file "a_srvcfg" contains the server configuration variables; when the file is first installed, these are all commented out. You can modify both of these files later, if desired.

5. If you want the AcuServer file server to start automatically on boot, **acuserve** must be installed and started as a Windows NT/Windows 2000 - 2008 service. You can install and start the service after initial setup either through the AcuServer Control Panel or by using the **acuserve** command from the Windows command line, as described in **Chapter 5, "Administrator Utilities and Functions."**

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

   The default service is named "AcuServer." All other service names include the port on which to run.

   The Start Menu folder contains shortcuts to the AcuServer Control Panel, the AcuPing utility, and online documentation. The Start Menu folder is not required to run AcuServer. You can safely remove it.

6. If you choose to start AcuServer during the installation process, a command prompt window is displayed showing the status of the Windows NT/Windows 2000 - 2008 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:

```
acuserve -kill
      Open/Control Service failed
acuserve -remove
      Open/Control Service failed
acuserve -install
      AcuServer service installed
acuserve -start
      STATE: START PENDING
```

   The Windows NT/Windows 2000 - 2008 services that are needed for AcuServer can also be started and stopped manually. See **section 5.6, "Starting and Stopping acuserve,"** for a detailed explanation of how to start and stop **acuserve** services.

   Should you need to remove a service after installation, see **section 5.5, "Installing and Removing acuserve as a Windows Service."** Note that removing the service does not delete the executables.

To help in resolving service problems, we display some messages from these services in Microsoft Event Viewer's Application Log. The Event Viewer is found in the **Administrative Tools** folder of the Windows Control Panel.

3.4 Setting Up Accounts

As you configure AcuServer, you need to establish user accounts on the Windows NT, Windows 2000 - 2008 host system. This section lists some key considerations for performing this activity. For specific instructions on setting up user accounts, consult your operating system documentation.

When setting up user accounts, determine whether each user will have a unique account, or whether groups of users will share an account (such as "Accounting") on the server. The table below shows some tradeoffs to help you make that decision:

Account Characteristics:

Feature	Individual Accounts	Group Accounts
Security is configured user-by-user.	yes	
Log files can track usage by individual user.	yes	
Requires Administrator to create an account for every user.	yes	
Existing configuration and permissions are easily changed.		yes
Quickly configure a large group of users who need identical access to the same resources.		yes
Easily grant identical access to a bank of machines shared by a department.		yes

Note: Members of a group can be logged on simultaneously with the same account.

If you choose group accounts on the server, users may still have individual accounts on their client machines. The username returned by ACCEPT FROM SYSTEM-INFO will be the local (client) login-ID.

The **acuserve** service is usually run as the "SYSTEM" account. If you need to change the account used by the **acuserve** service, be sure that the new account has all of the appropriate permissions for a file server.

3.5 AcuServer System Security

To establish a secure and functional AcuServer system, it is important to restrict access to the **acuserve** executable, server configuration files, and server access files.

In addition, you will need to set appropriately restrictive ownerships and permissions for your existing data files and directories.

3.5.1 Setting Ownerships and Permissions

User Privileges

Windows NT and Windows 2000 - 2008 control access to resources with Access Control Lists (ACLs). An ACL specifically grants access to a user or to a group. Privileges are additive, meaning that users have the highest access given to their account and to any groups to which they belong. The exception is "No Access," which overrides any other privileges.

The group "Everyone" contains every account on the system. Using this group is a handy way to set privileges, but can be a risky way to deny them. If a file or directory has "No Access" for Everyone, it will be unusable until someone (such as the Administrator) takes ownership and resets the privileges.

File Permissions

File permissions are set by the account that owns the file. To override permissions, a non-owner must have the "Take Ownership" privilege and use it to take ownership of the file before setting permissions.

Setting file permissions requires Administrator privileges. The following sections list some key considerations for performing this activity. For specific instructions, consult your operating system documentation.

3.5.2 File Access Security

AcuServer allows users of Windows NT and Windows 2000 - 2008 networks to choose between the system security that AcuServer offers and the security features offered by their own operating system.

Note: On Windows 2008 it is essentially required that you use the operating system's security methods. This is mainly due to a new security layer called User Account Control (UAC). Consult Windows Help for information on how UAC impacts file and program access.

By default, AcuServer system security is used to govern file access. (This is discussed in **Chapter 6**.)

If you use AcuServer security, the AcuAccess and AcuAccess.vix files are used to establish user access privileges. Remember that these two files must be readable and writable by *only* "Administrator" and "System" or AcuServer will not start.

Note: If you experience a problem with file access, it can be helpful to give the users Full Control of the files and directories they need to use. After everything is working smoothly, reduce access to Read-Only if desired. However, be sure to test each program to make sure that everything continues to function as you expect with the reduced privileges.

If you prefer to use Windows security mechanisms (recommended), you may do so by setting the configuration variable **SECURITY_METHOD** in both the runtime configuration file and server configuration file. For more information about the SECURITY_METHOD configuration variable, see **section 4.2.2, "Server Configuration Variables."**

Note that if you decide to use Windows security instead of AcuServer system security, your users will have whatever permissions the system administrator has set up for them.

3.5.3 Setting Permissions on New Files

When a client application makes its initial request to AcuServer for services, the requester is validated for permission to use AcuServer. If the requester is permitted to use AcuServer, a user name is assigned to the requester based on the Local Username field of the matching server access record (see **section 6.5, "AcuServer Connection Logic"**). Files created for that requester by AcuServer get the *user* and *group* ownerships of the assigned Local Username.

umask

The *read* and *write* permissions set on new files are determined by the *umask* specified in the matching server access record (because all files created are data files, *execute* permission is not applicable).

On Windows NT and Windows 2000 - 2008 servers, when AcuServer security is being used, the *umask* is used to set read and write permissions for the *owner* (leftmost digit) and for *other* (rightmost digit). The *group* digit (middle) is ignored. *umask* is not used when Windows NT security is enabled. See **Chapter 6, "System Security,"** for more information about AcuServer security.

3.6 Installing the Client

There is nothing on the AcuServer distribution media to install on the client machine. However, you should ensure that every client system that will use AcuServer has a licensed copy of the current ACUCOBOL-GT runtime. To check that your runtime is current, use the "-v" option on the runtime command line of the client machine. The runtime will display information (either at the command line or in a Windows message box) similar to the following:

```
ACUCOBOL-GT runtime version 8.1
Serial number 1234
Licensed for 2 user(s)
AcuServer client
Vision version 5 file system
Copyright (c) 1985 - 2008, Micro Focus (IP) Ltd.
```

Look for the line "AcuServer client" and a runtime version number that matches the AcuServer version. If either of these criteria is missing, the client machine will not be able to access data using AcuServer.

3.6.1 Passwords for Clients

If you require a client password to access the server machine, you can prompt the user for this password and store it in the variable Acu_Client_Password, or you can let the runtime prompt for it. Be sure to read **section 6.5.1, "Passwords,"** if you plan to employ user passwords.

3.6.2 Setting Up the Host Name

The runtime system on each client machine passes the client hostname to the server machine.

Windows 2000/2003/XP/Vista

For Windows 2000 or 2003 and Windows XP clients, right-click on the "My Computer" icon on the desktop and select "Properties" from the pop-up menu.

- On a Windows 2000 or 2003 client, select the Network Identification tab of the System Properties window.
- On a Windows XP or Vista client, select the Computer Name tab of the System Properties window.

In most instances, the computer name listed in the “Full computer name” field is the host name used by the runtime.

You can also determine the host name by typing the command “hostname” at the Windows command prompt.

Windows NT

1. On Windows NT clients, open the Network folder in the Control Panel and select “Protocols”.
2. Select TCP/IP, then click on “Properties.”
3. Select “DNS”, and find the “Host Name” entry.

The name you specify for “Host Name” is the one that the runtime uses.

UNIX

On UNIX clients, the runtime uses the host name that is in the file “/etc/hosts”.

3.6.3 Setting Up the User Name

For Windows clients logged into a Windows NT network, the runtime uses the NT domain user name that the user logged in with. For Windows clients that are not logged into a Windows NT network, the runtime uses the user name value that is set by the environment variable USERNAME. If USERNAME is not set, then the runtime uses the value that is set by the environment variable USER. (The values assigned to these variables are case-sensitive. Be sure that the case used in the AcuAccess file matches the case of the value set in the variable.) If neither of these environment variables is set, then the runtime uses the name “USER”.

The **acuserve** service is usually run as the "SYSTEM" account. If you need to change the account used by the **acuserve** service, be sure that the new account has all of the appropriate permissions for a file server.

For UNIX clients, the runtime uses the user name that the user logged in with.

3.6.4 Confirming Network Services

On UNIX and Windows clients, confirm that your TCP/IP software is loaded and running.

If desired, use "ping" to confirm that the server is up and responding to the network. From your client system, type:

```
ping hostname
```

Ping should return:

```
hostname is alive
```

Alternatively, you can use our AcuPing program, which is found in the Start/Programs/Acucorp 8.*x*.*x*/AcuServer menu. AcuPing is discussed in **section 8.3, "Diagnosing Errors with C$PING."**

3.7 Using AcuServer's Graphical User Interface for Windows

Executing **acuserve** in Windows automatically launches the AcuServer Control Panel (ACP), a graphical-based administrative utility based on AcuServer's command-line requirements.

Note: To use the AcuServer Control Panel on Windows 2008 where UAC 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 use the ACP to perform the same tasks that you would otherwise perform by issuing the **acuserve** command at the command line. You can:

- Edit an access file (add, change, or delete user security information)
- Set and view current configuration settings for a running server
- View, unlock, and close open files and connections
- Install, start, stop, modify, or remove AcuServer services on any port
- Monitor a port for status and open files

Chapter 5, "Administrator Utilities and Functions," describes how to perform AcuServer functions from the command line and the ACP. You can access the AcuServer documentation from the ACP by clicking **Help** or by pressing **F1**.

4 Configuration

Key Topics

Configuring AcuServer 4-2
Runtime Configuration Variables 4-3
Server Configuration Variables 4-8

4.1 Configuring AcuServer

Before AcuServer® file server software is ready for use, you must evaluate and configure several system attributes. AcuServer configuration consists of:

- assigning values to runtime configuration variables (**section 4.2.1**)
- assigning values to server configuration variables (**section 4.2.2**)
- creating the server access file/authorization database (**Chapter 6**)
- assigning ownerships and permissions to the **acuserve** executable file, the server configuration file, the server access file, and existing data files and directories

Enabling the application software to use AcuServer is covered in **Chapter 7, "Programming for AcuServer."**

4.2 Configuration Variables

Some attributes of AcuServer can be controlled using ACUCOBOL-GT® *runtime* configuration variables and AcuServer *server* configuration variables.

Runtime and server configuration variables are stored in separate plain-text files, located on their respective host machines. The content of these files is easily modified with the host system's standard text editor.

Each configuration variable entry consists of a single line in the file. Each entry starts with a keyword, followed by one or more spaces or tabs, and a value. Note that configuration file entries treat upper and lower case characters as equivalent, and hyphens and underscores as equivalent. A configuration variable definition might look like:

```
LOCKS_PER_FILE   10
```

4.2.1 Runtime Configuration Variables

On UNIX systems, the runtime configuration file is typically named "cblconfig" and is located in "/etc" by default. On Windows systems, the configuration file is typically named "cblconfi" and located, by default, in the "\etc" directory on the same drive as the operating system.

If the configuration file has a different name or is located in another directory, the full path and name of the file must be specified with the "-c" option when you start the ACUCOBOL-GT application (for example, "wrun32 -c *device:\path\config_file program_name program_parameters*"). See Chapter 2 of the *ACUCOBOL-GT User's Guide* for a complete description of the runtime configuration file and all options to **runcbl** and **wrun32**.

Several configuration variables that appear only in the runtime configuration file can affect AcuServer's behavior. These variables, which include **AGS_PING_TIME, AGS_SOCKET_COMPRESS, AGS_SOCKET_ENCRYPT, CACHE_DIRECTORY, filename_MRC**, and **USE_LOCAL_SERVER,** are discussed in this section.

Some specialized variables appear only in the server configuration file, while other configuration variables, including **ACUSERVER_PORT**, **DEFAULT_TIMEOUT**, **ENCRYPTION_SEED**, **LOCKS_PER_FILE**, **MAX_LOCKS**, **SECURITY_METHOD**, **TEXT**, and **V_BUFFERS**, can or must be used in both the runtime and server configuration files. Most of these variables are discussed in **section 4.2.2, "Server Configuration Variables."** The remainder (**FILE_PREFIX**, **CODE_PREFIX**, **APPLY_CODE_PATH**, and **APPLY_FILE_PATH**) are discussed in **section 7.2.2, "Using FILE_PREFIX and CODE_PREFIX."**

4.2.1.1 AGS_PING_TIME

AGS_PING_TIME is used in combination with the server configuration variable DEAD_CLIENT_TIMEOUT to enable a mechanism in AcuServer that automatically detects when a client "hangs" or "disappears" (see **section 8.1, "Machine Failures,"** for a complete description).

Note: If you use AGS_PING_TIME, the variable must be set *before* the socket connection between the client and the server is created. This means that the runtime configuration file must reside on the client, rather than on the server, for AcuServer to detect lost connections.

If you set AGS_PING_TIME to "-1", the default value, the client does not participate in the server's dead client detection service.

If you set AGS_PING_TIME to any other value, the client reporting mechanism is enabled and the connection is included in **acuserve**'s monitoring table (if it is enabled). The value of AGS_PING_TIME specifies the interval, in seconds, in which the client will send an "I'm alive" message (a "no-op" instruction) to **acuserve**. The value is communicated to **acuserve** at the time that the connection is established so that **acuserve** can determine, when it checks its monitoring records, whether the connection has missed two or more consecutive communication periods. If the client misses two or more consecutive communications periods, AcuServer determines that the client is no longer responding and it closes all associated files, releases all associated locks, and closes the associated socket.

Note: When you are running a program in debug mode, the runtime is not able to send ping messages to the debugger prompt. Therefore, if you are debugging a COBOL program, we suggest that you set AGS_PING_TIME to "-1".

See also

DEAD_CLIENT_TIMEOUT server configuration variable

4.2.1.2 AGS_SOCKET_COMPRESS

The AGS_SOCKET_COMPRESS variable allows you to specify one of three data compression settings:

NONE (default)	No compression is performed.
ZLIB	Socket data is compressed using the same algorithm as the "gzip" compression utility. All versions of Windows support ZLIB compression, but some UNIX machines do not. When ZLIB compression is specified on a machine that does not support the format, RUNLENGTH compression is automatically used instead.
RUNLENGTH	Socket data is compressed using simple compression based on counting repeated bytes of data.

Note that RUNLENGTH compression is much faster than ZLIB compression. ZLIB compression, however, can compress data much more than RUNLENGTH compression.

The AGS_SOCKET_COMPRESS variable must be set *before* any socket communication is performed and cannot be changed via SET ENVIRONMENT.

4.2.1.3 AGS_SOCKET_ENCRYPT

Set AGS_SOCKET_ENCRYPT to "1" (on, true, yes) to turn on AcuServer encryption. The default value is "0" (off, false, no). Note that encryption can have a significant impact on performance. See **section 6.6, "Encryption,"** for complete information about AcuServer encryption and use.

Note: The AGS_SOCKET_ENCRYPT configuration variable replaces AS_CLIENT_ENCRYPT. In a configuration file where the old AS_CLIENT_ENCRYPT variable is set to "1" (on, true, yes), AGS_SOCKET_ENCRYPT is also set to "1".

4.2.1.4 CACHE_DIRECTORY

By default, AcuServer creates a local, cached copy of any COBOL object files executed by the client in order to improve performance.

Use the CACHE-DIRECTORY configuration variable to specify the directory in which to store cached object files.

If you do not set CACHE-DIRECTORY, the runtime searches the other temporary directory variables (A_TMPDIR, TMPDIR, TEMP and TMP), uses the first found, and appends "/acu" to the name of the directory. The runtime attempts to create the new directory. If the directory does not exist and cannot be created, caching is not performed.

AcuServer creates a unique filename for the cached file based on the date, time, size, and location of the file, so that as long as the object file doesn't change, the name of the cached file will remain the same.

To disable local caching, use the server configuration variable NO_LOCAL_CACHE, described in **section 4.2.2**.

4.2.1.5 CACHE_DIRECTORY_SIZE

By default, AcuServer copies any COBOL object files executed by the client to a local cache directory to improve performance. This variable allows you to limit the size of the directory in which these files are stored.

Set CACHE_DIRECTORY_SIZE to a whole integer value to specify the maximum number of megabytes available in the storage directory. For example, when this variable is set to the default value of "5", the sum of the sizes of all files in the cache must be less than 5MB. File size is checked before a new file is downloaded from the server. If the directory reaches maximum size, the oldest file is deleted first. In the rare event that there are two or more files that are equally old, the largest is deleted first. Files are deleted until there is room to save the new file.

When the runtime determines the size of the cache directory, it considers only object files it has cached into that directory. In particular, it counts only the sizes of object files that are from the server it is talking to (including the port on which AcuServer is running). Other object files are not considered, so they are never automatically removed by the runtime.

4.2.1.6 *filename*_MRC

The variable *filename*_MRC (MULTIPLE_RECORD_COUNT) applies only to those using AcuServer's "multiple-record mode." Use this variable to specify that you want the named file to be opened in multiple-record mode whenever it is allowed (whenever the user has exclusive I/O access, or the file is open INPUT).

Note: Files opened in regular I/O mode will be transmitted a single record at a time, even if they are named in the runtime configuration file using this variable.

For *filename*, enter the final resolved name or physical filename on disk of the file to be used in this mode. Do not include any directory names and/or file extensions. For example, if the file that you wanted to open in multiple-record mode were named "/user1/gl.dat", create a variable GL_MRC. Assign a value to this variable to reflect the number of records to be read in each network packet, for instance, GL_MRC 10. Valid values range from "0" to "32767". Setting this variable to "0" is equivalent to not setting it at all. Set this variable to "1" if you want the server to determine how many records to send to the client at once. AcuServer then examines the server configuration variable MULTIPLE_RECORD_COUNT and uses that value to determine how many records to send.

There are some exceptions to these general rules. For instance, when the client executes a random READ, the server sends only a single record. The next time the client executes a READ NEXT, the server sends a full set of records.

See **section 7.4, "Multiple-Record Mode,"** for more information on multiple-record mode. Note that files opened in multiple-record mode have certain restrictions upon them. Refer to **section 7.4.1, "File Limitations,"** for details.

4.2.1.7 USE_LOCAL_SERVER

Set USE_LOCAL_SERVER when you want to run AcuServer client applications on the same machine as the AcuServer file server. This variable allows you to specify whether to use or bypass AcuServer when accessing local files that have remote name notation.

Set USE_LOCAL_SERVER to “0” (the default setting) to *prevent* the use of AcuServer to access local files that have remote name notation (the remote name is stripped off and the file I/O operation is handled by the runtime). Set USE_LOCAL_SERVER to “1” to *use* AcuServer to access local files that have remote name notation.

4.2.2 Server Configuration Variables

The server configuration file is named “a_srvcfg”. On UNIX servers, it is located in “/etc” by default. On Windows NT, Windows 2000 - 2008 servers, it is located in the directory “c:\etc” (where *c:* is the letter of the drive on which the operating system is installed). If the file is given another name, or located in another directory, you must provide the full location and name of the file.

To specify the name and location of a server configuration file at the command line, use the “-c” option when you start AcuServer. On a Windows server, you can perform the same function from the Services tab of the AcuServer Control Panel. See **section 5.6, “Starting and Stopping acuserve,”** for more information about specifying **acuserve** startup options.

To change configuration values on the server or local machine without changing the configuration file on disk, use the Config tab. Using the Config tab is equivalent to issuing a “SET ENVIRONMENT” command from the runtime to modify a runtime configuration value.

Click **Query** to select the **acuserve** service whose configuration you want to view or modify, then use the **New** and **Modify** buttons to add or change configuration variables in the server’s configuration file.

There are several server configuration variables. They are:

Name	Default Value
ACCESS_FILE	/etc/AcuAccess
ACUSERVER_MASTER_SERVER	undefined
AGS_SERVER_SOCKET_RESERVE	0
COUNT_STATISTICS	0 (off, false, no)
DEAD_CLIENT_TIMEOUT	-1 (off)

Name	Default Value
DEFAULT_TIMEOUT*	25
DEFAULT_UMASK	0
DEFAULT_USER	undefined
ENCRYPTION_SEED	undocumented for increased security
FILE_TRACE*	0
FILE_TRACE_FLUSH	0
FILE_TRACE_TIMESTAMP	0 (off, false, no)
filename_DATA_FMT*	decimal extensions
filename_INDEX_FMT*	decimal extensions
filename_VERSION*	0
LOCK_ALL_FILES	0
LOCKS_PER_FILE*	10
MAX_ERROR_LINES*	0
MAX_FILES*	32
MAX_LOCKS*	32
MULTIPLE_RECORD_COUNT	10
NO_LOCAL_CACHE	0 (off, false, no)
PASSWORD_ATTEMPTS	3
PROVIDE_PASSWORD_MESSAGES	0 (off, false, no)
SECURITY_METHOD*	none
SERVER_IP	undefined
SERVER_NAME	undefined
SERVER_PORT*	6523
TEXT	see entry for TEXT, below
V_BASENAME_TRANSLATION*	1 (on, true, yes)
USE_SYSTEM_RESTRICTIONS	0 (off, false, no)
V_BUFFER_DATA*	1 (on, true, yes)

Name	Default Value
V_BUFFERS*	64
V_INDEX_BLOCK_PERCENT*	100
V_READ_AHEAD*	1 (on, true, yes)
V_SEG_SIZE*	2147482623
V_STRIP_DOT_EXTENSION*	1 (on, true, yes)
V_VERSION*	0
WINNT-EVENTLOG-DOMAIN	undefined
WINNT-LOGON-DOMAIN	undefined

Variables marked with an asterisk (“*”) can appear in both the runtime (“cblconfig”) and server (“a_srvcfg”) configuration files. Those that *must* appear in both locations are indicated in the descriptions that follow. Note that values assigned to variables contained in the server configuration file are applied solely to operations performed by AcuServer.

Note that when file access is made through AcuServer, the values assigned to the following in the runtime configuration file are ignored:

- *filename*_DATA_FMT
- *filename*_INDEX_FMT
- *filename*_VERSION
- V_BASENAME_TRANSLATION
- V_BUFFER_DATA
- V_BUFFERS
- V_INDEX_BLOCK_PERCENT
- V_READ_AHEAD
- V_SEG_SIZE
- V_STRIP_DOT_EXTENSION

- V_VERSION

4.2.2.1 ACCESS_FILE

The variable ACCESS_FILE, as well as the variables DEFAULT_USER and PASSWORD_ATTEMPTS, affects AcuServer access security.

ACCESS_FILE must hold the full path and file name of the server access file, if it is other than the default ("/etc/AcuAccess" on UNIX systems; "\etc\AcuAccess" on Windows NT, Windows 2000 - 2008).

4.2.2.2 ACUSERVER_MASTER_SERVER

This variable is used for load-balancing in environments with multiple instances of the **acuserve** daemon/service. One instance of **acuserve** is designated the master, or primary, server. This primary server is responsible for distributing client requests among the available **acuserve** daemons. Note that you must start the daemon designated as the primary server *before* you start any secondary servers.

After a primary server is specified, you can start other **acuserve** instances as secondary, or slave, servers. Instances of **acuserve** that are configured as secondary servers register with the primary server to indicate that they are ready to allow connections. When a secondary server shuts down, it notifies the primary server that it is no longer available for client requests.

If the master server dies unexpectedly, the "-register" command can be used to re-register the secondary or subordinate servers with a newly started master server. This avoids having to shut down and restart the entire system. See **Registering Servers** for details.

In the following example, the primary server resides on a server called "jupiter" and listens at port 4331.

```
ACUSERVER_MASTER_SERVER jupiter.mycompany.com:4331
```

Note that you must specify a fully qualified domain name for your server.

If this value appears in the configuration files for all instances of **acuserve** in the environment, the primary server both manages communication with secondary servers and serves files itself.

To create a primary server that performs only load-balancing services, do not specify a value for ACUSERVER_MASTER_SERVER in the configuration file for the primary server. Configuration files for the secondary servers must still contain the machine name and port of the primary.

To configure the client machines to take advantage of the server load balancing, set the FILE_PREFIX and ACUSERVER_PORT runtime configuration variables to point to the instance of **acuserve** designated as the primary server. On the server side, set FILE_PREFIX to local file locations in the configuration file(s) used by the secondary servers.

When a client connects to the server, it first communicates with the primary server, which determines which secondary is least busy. Then the primary server passes the client connection to a secondary server, and all further communication occurs between the client and the secondary server, without further involvement of the primary server.

To determine which secondary server is least busy, the client considers the following criteria in the following order:

1. percentage of idle time in the last 15 minutes
2. number of active users
3. number of open files
4. if all other factors are equal, registration order

4.2.2.3 AGS_SERVER_SOCKET_RESERVE

AcuServer depends on the “select()” system call to determine when a client is requesting services. On most UNIX machines, this call requires an array of socket identifiers on which to await results. If this array is sparse, select() can take more time than is necessary.

To address this issue, set AGS_SERVER_SOCKET_RESERVE to a positive integer to reserve that many socket handles for exclusive use with clients. This allows select() to be more efficient, and it allows UNIX machines that allow more open files than open sockets to reserve socket identifiers with low values.

Note that this variable is valid only on UNIX servers, and that the variable must be set no larger than the size of the array used in the call to SELECT. To determine the maximum value, use the command "**acuserve** -vv".

4.2.2.4 COUNT_STATISTICS

When COUNT_STATISTICS is set to "1" (on, true, yes), AcuServer has the capability to collect and report various server statistics, described in **section 5.9**. Because a small performance hit is incurred when AcuServer tracks its statistics, the default setting for this configuration variable is "0" (off, false, no).

4.2.2.5 DEAD_CLIENT_TIMEOUT

DEAD_CLIENT_TIMEOUT is used in conjunction with the runtime configuration variable AGS_PING_TIME to allow AcuServer to automatically detect when a client "hangs" or "disappears" (see **section 8.1, "Machine Failures,"** for a complete description).

When DEAD_CLIENT_TIMEOUT is set to "-1" (the default), the dead client detection mechanism is disabled.

When DEAD_CLIENT_TIMEOUT is set to any other value, the dead client detection mechanism is enabled and the variable's value is interpreted as the interval, in seconds, at which **acuserve** will analyze its records to detect dead connections.

For example, if DEAD_CLIENT_TIMEOUT is set to "300" (300 seconds = 5 minutes), the dead client detection mechanism is enabled on the server and **acuserve** analyzes its communication records every five minutes to look for dead connections.

See also

AGS_PING_TIME runtime configuration variable

4.2.2.6 DEFAULT_TIMEOUT

The variable DEFAULT_TIMEOUT defines the length of time, in seconds, that the runtime or **acuserve** will wait for a response before timing out. The default value is 25 seconds. Note that some networks have long connect times and, depending on network loading, the default value may not be long enough to allow the application to connect.

Set this configuration variable on the client when you want the client to stop attempting to communicate with the server should the server fail to respond within the specified period. This is particularly helpful when you want to keep the client program from "hanging" when the server is down or inaccessible. If the timer expires without a response from the server, an error 98,104,10060 is returned to the application. For the mechanism to work, a value must be assigned to the variable *before* the first file is accessed. If you use AcuServer to get the configuration file from a remote host, you must set DEFAULT_TIMEOUT in the local environment.

On the server, DEFAULT_TIMEOUT allows you to tell **acuserve** how long to wait for a client to complete a communication that the client has already started. If the timer expires without any further communication from the client, the socket is closed. This allows the server to detect when a client has crashed or disappeared in the middle of a communication. This variable does not help in the situation in which the client crashes or disappears between requests. For help in detecting dead connections, see **section 8.1, "Machine Failures."**

4.2.2.7 DEFAULT_UMASK

The variable DEFAULT_UMASK defines the file creation mask (umask) to be used for users whose umask has not been adequately defined in the AcuAccess file (that is, their umask was set to spaces). The default value is "000", which causes all files created by these users to be readable and writable by everyone. You can change DEFAULT_UMASK to provide whatever read/write permissions you desire for these undefined users. Valid values include any octal value between "000" and "777".

4.2.2.8 DEFAULT_USER

The variable DEFAULT_USER, as well as the variables ACCESS_FILE and PASSWORD_ATTEMPTS, affects AcuServer access security.

DEFAULT_USER holds the default user name given to AcuServer 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." For a description of when AcuServer assigns DEFAULT_USER to a requester, see **section 6.5, "AcuServer Connection Logic."**

4.2.2.9 ENCRYPTION_SEED

This variable holds a text string that is used to initialize the encryption module. The string can be any length. ENCRYPTION_SEED should be set in *both* the runtime configuration file (on the client) and in the server configuration file.

Note that the use of encryption can have a significant impact on performance. See **section 6.6, "Encryption,"** for complete information on AcuServer encryption and use.

4.2.2.10 FILE_TRACE

This variable allows you to start file tracing without specifying a "-t #" option on the command line when starting AcuServer. Set this variable to a nonzero value to save information about all file OPENs, READs, and WRITEs in the error file. Set it to "5," "6," or "7" to flush the error trace buffer to disk after every WRITE operation. The default is "0." Like other server configuration variables, you can change the value of FILE_TRACE while AcuServer is running by using the **acuserve** "-config" option.

4.2.2.11 FILE_TRACE_FLUSH

Setting FILE_TRACE_FLUSH to "1" (on) causes AcuServer to flush the error trace buffer to disk after every WRITE operation. This ensures that in the event that the **acuserve** daemon terminates abnormally, all file trace information, up to the last WRITE operation, is captured in the log file. If

this configuration variable is not used (set to “0” or “off”) and AcuServer aborts unexpectedly, an undefined amount of file trace information will be lost. FILE_TRACE_FLUSH is set to “0” by default. Note that you can also enable file trace flushing when you start the **acuserve** daemon by including the “-t 5”, “-t 6”, or “-t 7” command-line switch in the **-start** command (see **section 5.6, “Starting and Stopping acuserve,”** for more information) or by setting FILE_TRACE to “5”, “6”, or “7” in the server configuration file.

4.2.2.12 FILE_TRACE_TIMESTAMP

When you set FILE_TRACE_TIMESTAMP to “1” (on, true, yes), AcuServer places a timestamp at the beginning of every line in the trace file. The format of the timestamp is HH:MM:SS:mmmmmm, where *mmmmmm* is the finest resolution that ACUCOBOL-GT can obtain from the system.

Timestamp information is included only when file trace information is directed to a file. Timestamp information is never included in file trace output that is sent to the screen.

Note: Turning on timestamps can add significant overhead to file operations and may have a noticeable impact on performance.

4.2.2.13 *filename*_DATA_FMT

This configuration variable specifies a format for naming the data segments of a Vision 5 file. The configuration variable starts with the physical file’s base name followed by an underscore and the file’s extension, all in uppercase.

Suppose that the regular name of your COBOL file is “/usr1/gl.dat”. The variable you would use to set the data segment naming format for this file is GL_DAT_DATA_FMT.

The variable must be set to a pattern that shows how to create the segment names. The pattern shows how to form the base name and extension for each segment. Part of this pattern is a special escape sequence (such as “%d”) that specifies how the segment number should be represented. Choices include “%d” (decimal segment numbers), “%x” (lowercase hexadecimal numbers), “%X” (uppercase hexadecimal numbers), and “%o” (octal numbers).

For example, setting the variable GL_DAT_DATA_FMT=gl%d.dat would result in data segments named /usr1/gl.dat (remember that the first data segment is not affected), /usr1/gl1.dat, /usr1/gl2.dat, and so forth.

Escape Sequence Definitions:

The "%d" in the value of the *filename*_DATA_FMT above is a printf-style escape sequence. Most reference books on the C language contain an in-depth explanation of these escape sequences, and UNIX systems typically have a man page ("man printf") that explains them in detail. Here are the basics:

- "%d" expands into the decimal representation of the segment number.
- "%x" expands into the hexadecimal representation (with lowercase a-f) of the segment number.
- "%X" expands into the hexadecimal representation (with uppercase A-F) of the segment number.
- "%o" expands into the octal representation of the segment number.
- You can add leading zeros to the number (to keep all the file names the same length) by placing a zero and a length digit between the percent sign and the following character. For example, "%02d" would result in "00", "01", "02", and so forth, when expanded.
- To embed a literal "%" in the file name, use "%%".

The escape sequence can be positioned anywhere in the file name, including the extension.

Note: While AcuServer checks for this segment naming variable in the runtime configuration file as well as in the environment, utilities such as **vutil** and **vio** check only the environment. Therefore, if you are using this variable with **vio** or **vutil**, you must set the variable in the environment and not in the configuration file.

See *filename*_INDEX_FMT for details about naming the index segments. Both variables should be set to corresponding patterns.

4.2.2.14 *filename*_INDEX_FMT

This configuration variable specifies a format for naming the index segments of a Vision Version 5 file. The configuration variable starts with the file's base name followed by an underscore and the file's extension, all letters in upper case.

Suppose that the regular name of your COBOL file is "/usr1/gl.dat". The variable you would use to set the format for naming the file's index segments is GL_DAT_INDEX_FMT.

The variable must be set equal to a pattern that shows how to create the segment names. The pattern shows how to form the base name and how to form the extension for each segment. Part of this pattern is a special escape sequence (such as %d) that specifies how the segment number should be represented. Choices include %d (decimal segment numbers), %x (lowercase hexadecimal numbers), %X (uppercase hexadecimal numbers), and %o (octal numbers).

For example, setting the variable GL_DAT_INDEX_FMT=gl%d.idx would result in index segments named /usr1/gl0.idx, /usr1/gl1.idx, /usr1/gl2.idx, and so forth.

Escape Sequence Definitions:

The %d in the value of the *filename*_INDEX_FMT above is a printf-style escape sequence. Most reference books on the C language contain an in-depth explanation of these escape sequences, and UNIX systems typically have a man page ("man printf") that explains them in detail. Here are the basics:

- "%d" expands into the decimal representation of the segment number.
- "%x" expands into the hexadecimal representation (with lower case a-f) of the segment number.
- "%X" expands into the hexadecimal representation (with upper case A-F) of the segment number.
- "%o" expands into the octal representation of the segment number.

- You can add leading zeros to the number (to keep all the file names the same length) by placing a zero and a length digit between the percent sign and the following character. "%02d" would result in "00", "01", "02", and so forth when expanded.

- To embed a literal "%" in the file name, use "%%".

The escape sequence can be positioned anywhere in the file name, including the extension.

Note: While AcuServer checks for this segment naming variable in the runtime configuration file as well as in the environment, utilities such as **vutil** and **vio** check only the environment. Therefore, if you are using this variable with **vio** or **vutil**, you must set the variable in the environment and not in the configuration file.

See *filename*_DATA_FMT for details about naming the index segments. Both variables should be set to corresponding patterns.

4.2.2.15 *filename*_VERSION

This variable sets the file format on a file-by-file basis. Replace *filename* with the base name of the file (the filename minus directory and extension). The meaning of the variable is the same as for V_VERSION. This variable is useful if you want to have all your files in one format, with a few exceptions. For example, you might want to maintain most of your files in Version 5 format, but have a few files in Version 3 format to conserve file handles. Note that this variable affects the file format only when it is created. You can always rebuild the file in another format later.

This variable (and the V_VERSION variable) is most helpful when you are using transaction management. The transaction system does not record the format of the created file if an OPEN OUTPUT is done during a transaction, because the transaction system is not tied to any particular file system. If you need to recover a transaction, the system will recreate the OPEN OUTPUT files using the settings of the VERSION variables.

4.2.2.16 LOCK_ALL_FILES

This configuration variable is designed for UNIX systems that are dedicated AcuServer servers. If the variable LOCK_ALL_FILES is set to "1", it causes AcuServer to lock all files that it opens. Access to locked files is faster than access to files that are not locked. As a result, LOCK_ALL_FILES can improve data access speed. If AcuServer is running on a UNIX system that other users are also using, this configuration variable should be set to "0" (the default), or the other users may not have access to the files. On Windows systems, AcuServer always locks the files it uses, so it is not necessary to configure this option.

Note that ACUCOBOL-GT runtimes that do not use **acuserve** may also be locked out if this variable is set to "1". Therefore, if your site uses this option, all runtimes must use **acuserve** to access the files (set USE_LOCAL_SERVER to "1").

When LOCK_ALL_FILES is set to "0", files are no longer automatically locked, and access times are increased.

4.2.2.17 LOCKS_PER_FILE, MAX_FILES, and MAX_LOCKS

The values of LOCKS_PER_FILE, MAX_FILES, and MAX_LOCKS are applied to the corresponding executable of each configuration file. That is, the values of the variables defined in the runtime configuration file are applied to the executing client program, and the values defined in the server configuration file are applied to the **acuserve** process. Remote file access is restricted by the smaller of the two values.

For example, if we have the following configuration values:

runtime (cblconfig)	MAX_FILES	64
server (a_srvcfg)	MAX_FILES	32

the runtime is able to open 64 files with a limit of 32 remote files.

Alternatively, if the configuration values are:

runtime (cblconfig)	MAX_FILES	32
server (a_srvcfg)	MAX_FILES	64

the runtime will be able to open, at most, 32 remote files.

Determine values for LOCKS_PER_FILE, MAX_FILES, and MAX_LOCKS by noting the values set in "cblconfig" and then multiplying the value of each variable by the number of runtimes likely to access the server simultaneously. The resultant values will be sufficient to support the case where all runtimes have the maximum number of files open with the maximum number of locks applied. We recommended that the values assigned these variables in "a_srvcfg" not be smaller than the values assigned in "cblconfig."

For example, if you anticipate having four runtimes accessing the server simultaneously, and the settings of your runtime configuration variables are:

```
LOCKS_PER_FILE  10
MAX_FILES  30
MAX_LOCKS  40
```

the corresponding variables in "a_srvcfg" might be:

```
LOCKS_PER_FILE  40
MAX_FILES  120
MAX_LOCKS  160
```

The value of LOCKS_PER_FILE and MAX_FILES in the server configuration file cannot exceed "32767" for Vision files. The value of MAX_LOCKS cannot exceed "8191" for Vision files. To conserve system resources and improve runtime and server performance, set these variables to the smallest value that still meets your requirements.

4.2.2.18 MAX_ERROR_LINES

This variable sets the maximum number of lines that can be included in the error file. When the size of the error file reaches the number of lines specified in this variable, the error file is *rewound* to its beginning, and

subsequent lines of output overwrite existing lines in the file. For example, if you set the maximum to 300, then lines 301 and 302 would overwrite lines 1 and 2 in the file.

To help you locate the end of the file, the message "*** End of log ***" is output whenever AcuServer shuts down. Line numbers are included in each line of the file, in columns one through seven. The default value is "0", which means do not limit the size of the file.

4.2.2.19 MULTIPLE_RECORD_COUNT

The variable MULTIPLE_RECORD_COUNT applies only to those using AcuServer's "multiple record mode." If you plan to open records in this mode, use this variable to specify how many records can be read in a single network packet. In other words, use this variable to specify how many records the server should send to the client each time the client requests more records. Note that this variable is checked only if the *filename*_MRC variable on the client is set to "1". Any other value on the client causes this variable to be disregarded.

Valid values of MULTIPLE_RECORD_COUNT range from "0" to "32767". The default value is "10". If this variable is set to "0", files opened in multiple record mode requesting a single record are treated as if they were being opened in standard mode.

Refer to **section 7.4, "Multiple-Record Mode,"** for more information on using AcuServer's multiple record mode.

4.2.2.20 NO_LOCAL_CACHE

By default, AcuServer copies any COBOL object files executed by the client to a local cache file to improve performance. To prevent local caching, set NO_LOCAL_CACHE to "TRUE".

4.2.2.21 PASSWORD_ATTEMPTS

The variable PASSWORD_ATTEMPTS, like the variables DEFAULT_USER and ACCESS_FILE, affects AcuServer 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 one (“1”) is used. If a non-integer value is given, the default value (“3”) is used.

4.2.2.22 PROVIDE_PASSWORD_MESSAGES

When set to “true” (“on”, “1”), this variable causes the values of the password messages (TEXT_015, TEXT_016, and TEXT_017) to be sent to the client upon initial connection (prior to prompting the user for a password, if a password is required). This allows password messages to be defined on the server, in the server configuration file, and then downloaded to the client when a connection is established. See the entry for **TEXT**.

4.2.2.23 SECURITY_METHOD

This variable lets you specify the security method to use when accessing files via AcuServer. If desired, you can use your server’s native security mechanism in place of AcuServer’s so that users can use their usual passwords when accessing files rather than remembering or coordinating an AcuAccess password as well as a network password. On Windows 2008 it is essentially required that you use Windows security methods. This is mainly due to the Windows implementation of User Account Control (UAC) security feature.

Note: This feature works for Windows servers and UNIX servers that use shadow passwords. It does not use PAM (pluggable authentication module) libraries.

When native security is enabled, AcuServer impersonates the user who is logged onto the client machine. This allows AcuServer to spend less time managing security issues, and allows the full range of operating system security to be used on files and directories on the server. To use this feature, you must have the necessary user accounts set up and configured on the server. (Ask your system administrator whether security has been set up for each potential user.) Once a user is connected, it is as if that user is actually logged onto the server. Files that are available to the user when he or she is directly logged on to the server are available to the user who is connected via AcuServer.

SECURITY_METHOD can take any of the following three values:

NONE (false, no)

Do not use the native operating system security. Use AcuServer security instead. "NONE" is the default value.

LOGON

Use the system's native security to manage user logons.

On Windows, attempt to log the user onto the Windows NT domain specified in the WINNT-LOGON-DOMAIN configuration variable. If the access file allows the connection, check the access record for the domain password. If it is present, establish a connection. If the password is not present in the access record or does not match the domain password, prompt the user for the domain password of the local username account. The number of attempts the user has to supply the correct password is limited by the value of the configuration variable PASSWORD_ATTEMPTS ("3", by default). A successful logon grants users all of the same access rights they would have if they were directly logged onto the server. AcuServer allows the Windows NT/Windows 2000 server to manage all issues pertaining to access permissions.

On UNIX, if the access file allows the connection, check the access record for the native password. If it is present, establish a connection. If the password is not present in the access record or does not match the native password, prompt the user for the native password of the local username account.

Some UNIX machines have a the ability to restrict access to the machine based on various parameters. If you want to include those restrictions in AcuServer, set the configuration variable USE-SYSTEM-RESTRICTIONS to "TRUE". See your UNIX administration manuals for information on this restriction feature.

NAMED-PIPE (on, true, yes)

"NAMED PIPE" can be specified only when a Windows client is connecting to a Windows server. It is not valid for UNIX clients and is treated as "NONE" if specified for UNIX.

A value of “NAMED PIPE” tells AcuServer to use Windows security based on the connection made from the client to the server via a named pipe. When a named pipe is used, the password field in the AcuAccess file is ignored. The AcuAccess file is used only as a first check to see if the user connecting to the server is allowed to connect.

To use this option successfully, your client machine must have permission to connect to the named pipe that AcuServer creates. If your machine does not have permission, it may look to you as though your client runtime has hung and it may look to other users as though the server is down. Without the proper permission, the only way to resolve this situation is to kill the server using the Windows Task Manager. See your Windows system administrator for help in establishing named pipe permissions and resolving connection problems.

The SECURITY_METHOD configuration variable must be set in both the client runtime configuration file *and* the AcuServer configuration file. The values must match or the security method reverts to “NONE”, and only the AcuAccess password is used.

4.2.2.24 SERVER_IP and SERVER_NAME

The variables SERVER_IP and SERVER_NAME allow you to assign a specific IP address to an individual instance of the **acuserve** service/daemon. Use these variables when the server has two or more network addresses and you want to associate a particular network address with a given instance of **acuserve**.

To specify an *IP address*, include an entry for SERVER_IP in the configuration file and assign it the value of the desired IP address on the host. For example:

```
SERVER_IP = 192.215.170.107
```

To specify a *hostname*, make an entry for SERVER_NAME in the configuration file and assign it the value of a valid hostname on the server. For example:

```
SERVER_NAME = bigserve
```

where *bigserve* is the hostname for IP address 192.215.170.107.

If both variables are defined in the configuration file, **acuserve** uses the value of the first valid entry in the file.

By defining different values for these variables in separate server configuration files on the host and then specifying a particular configuration file when you start the **acuserve** daemon, you can associate a specific IP address/hostname with each instance of **acuserve**.

4.2.2.25 SERVER_PORT

The variable SERVER_PORT specifies a particular port number to be used for accessing **acuserve** 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 **acuserve** daemon or service can work with *privileged* port numbers (from "0" to "1023") and *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 **acuserve** is started.

Caution: The SERVER_PORT variable must be specified in both the server configuration file and the runtime configuration file on the client, and the values of both variables must match. If you change the default value in one location, you must also change it in the other.

4.2.2.26 TEXT

The TEXT configuration variable is used to control the text of selected runtime messages. Three runtime messages have been added to support AcuServer password handling. These messages are numbered 15, 16, and 17. The default text for each message is:

15 "A password is required to connect to host %s."

16 "Please enter a password:"

17 "Invalid password."

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

To change the text associated with a given message in the *runtime* configuration file, place the word "TEXT" at the beginning of a line, followed by a space, the message number, an "=" character, and the text to be used when that message number is displayed. For example:

```
TEXT 16=Enter your password now:
```

If you define these messages in the *server* configuration file ("a_srvcfg"), you can specify that they are downloaded and used by each client (the messages are downloaded upon initial connection with AcuServer). To turn this feature on, set the configuration variable PROVIDE_PASSWORD_MESSAGES to "true" (see above).

Messages defined in the server configuration file have a slightly different format. To change a message, place the variable TEXT_015, TEXT_016, or TEXT_017 at the beginning of a line, followed by an "=" character, and the text to be used when that message number is displayed. For example:

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

Note: Use of AcuServer password protection is optional. If password protection is not used, these messages are not displayed.

4.2.2.27 USE_SYSTEM_RESTRICTIONS

This configuration variable applies only to UNIX servers and is used only when the SECURITY_METHOD variable is set to "LOGON".

Some UNIX machines have the ability to restrict access to the machine based on various parameters. If you want to include those restrictions in AcuServer, set this variable to "TRUE". When set, this variable tells AcuServer to call the "loginrestrictions()" function to determine whether it is okay to log on or not. (AcuServer first verifies the password against the system password files). See the man page for "loginrestrictions()" to determine how to set up restrictions.

4.2.2.28 V_BASENAME_TRANSLATION

This variable determines whether or not Vision includes full path information in a filename. When V_BASENAME_TRANSLATION is set to the default value of "1" (on, true, yes), only the file's base name (the filename without any extension or path information) is stored. When the variable is set to "0" (off, false, no), complete path and filename information is stored.

Retaining path information can be helpful if you have Vision files with the same name stored in different locations and you want to map one of the segments from one directory to a new location.

4.2.2.29 V_BUFFERS

This variable determines the number of indexed block buffers to allocate on the client (if used in a runtime configuration file) or on the server (if used in the server configuration file). These buffers are used to improve the performance of indexed files. Each buffer is 512 bytes plus some overhead. Setting this number larger will generally improve file performance. Setting it smaller will save memory. The value can range from "0" (no buffering) to "2097152". The default value for use with AcuServer is "64".

4.2.2.30 V_BUFFER_DATA

This variable determines whether or not Vision indexed file data blocks (as opposed to key blocks) will be held in the memory-resident disk buffers. When it is set to "1" (on, true, yes), the default, both data blocks and key blocks will use the buffers. When set to "0" (off, false, no), only key blocks will use the buffers. Setting this value to "1" will usually improve performance unless very few buffers are being used.

Note: Holding data blocks in the buffers slightly increases the chances of losing data if a file opened for MASS_UPDATE is not closed properly (such as a power failure). The default setting of this variable is "1".

4.2.2.31 V_INDEX_BLOCK_PERCENT

Because the index data stored in index segments is often much smaller than the record data contained in data segments of Vision 5 files, a large pre-allocate or extension factor typically allocates many more blocks than are needed. V_INDEX_BLOCK_PERCENT allows you to determine the ratio of index block to data blocks that are created for a Vision file, which can help trim file size when disk space is tight.

When V_INDEX_BLOCK_PERCENT is set to a value smaller than “100”, fewer index blocks are created than data blocks. When this variable is set to a value greater than “100”, more index blocks are created than data blocks. The value range for V_INDEX_BLOCK_PERCENT is “1” to “1000”, and the default value is “100”.

For example, if you have a file with an extension factor of 10 and set V_INDEX_BLOCK_PERCENT to “50”, the next time the file is extended, 10 new data blocks and 5 new index blocks are created. If you set V_INDEX_BLOCK_PERCENT to “200” for the same file, the next time the file is extended, 10 new data blocks and 20 new index blocks are created.

4.2.2.32 V_READ_AHEAD

Setting this configuration variable to “0” (off, false, no) turns off Vision’s read-ahead logic. This may improve performance in cases where highly random file processing is being used. The default value is “1” (on, true, yes).

4.2.2.33 V_STRIP_DOT_EXTENSION

Ordinarily, Vision strips any trailing dot extension (such as “.dat”) from the logical name of a data file when generating file names for index and data segments (except for the first data segment). Setting V_STRIP_DOT_EXTENSION to “0” (off, false, no) allows you to prevent file extensions from being removed.

For example, the default first index segment name for the logical file "file.001" is "file.vix", which can cause problems if you also had a file called "file.002". When V_STRIP_DOT_EXTENSION is set to "0", the index segment name becomes "file.001.vix".

V_STRIP_DOT_EXTENSION is set to "1" (on, true, yes) by default.

4.2.2.34 V_SEG_SIZE

V_SEG_SIZE sets the maximum size of a Vision file segment. Vision Version 5 allows for very large logical files by creating additional physical files (called file segments) as needed.

The default value is 2 GB - 1 KB -1 (which is 2,147,482,623 or hexadecimal 7FFFFBFF). The default is also the maximum value allowed. For best performance, set this value as high as possible, to minimize the number of files created.

4.2.2.35 V_VERSION

This variable specifies the version number of new Vision data files that are created by AcuServer. The default value is "5", which produces Vision files in the latest format (Version 5). Version 5 files are generated in a dual file format, with data records stored in one segment and overhead key information stored in another. The value "4" produces Version 4 files, which also use the dual file format, but do not support the larger maximum record size and larger block sizes available in Version 5. The value "3" produces Version 3 files, in which data and keys are stored in a single file.

4.2.2.36 WINNT-EVENTLOG-DOMAIN

This variable applies only to Windows servers. Set it to the Universal Naming Convention (UNC) name of the computer to which you want to send event log messages. UNC names begin with two backslashes (\), indicating that the resource exists on a network computer. This variable is not changeable outside the configuration file.

Note: The event log machine that you specify must be able to receive messages from the AcuServer machine or message delivery will fail. On the event log machine, use the Windows control panel, User Accounts function to add the AcuServer machine name to the Users or Groups.

4.2.2.37 WINNT-LOGON-DOMAIN

This variable applies only to Windows servers. Set it to the name of the Windows domain to log users onto. For this variable to have an effect, you must set the SECURITY_METHOD variable to “LOGON”. Note that if the domain does not exist, users are denied access to the server.

5 Administrator Utilities and Functions

Key Topics

Overview 5-2
acuserve Command-line Options and Formats 5-3
AcuServer Control Panel interface 5-4
Creating and Maintaining the Server Access File 5-5
Installing and Removing acuserve as a Windows Service 5-5
Starting and Stopping acuserve 5-11
Changing acuserve Properties 5-17
Checking System Status 5-19
Tracking Server Statistics 5-22
Closing Stranded Files 5-26
Checking Version Information 5-31
Registering Servers 5-32

5.1 Overview

Several activities are involved in the configuration and use of AcuServer® file server software:

- creating and maintaining the server access file
- installing and removing **acuserve** as a service on Windows NT and Windows 2000 - 2008 systems
- starting and stopping the **acuserve** daemon or service
- changing the attributes of a running process
- generating AcuServer system status reports
- closing files and connections in response to machine failures and special circumstances
- verifying AcuServer version information

With the exception of generating status reports and verifying version information, these activities can be performed only by someone who has privileged access to the system. On UNIX and Linux systems, you must have *root* or *superuser* privileges. On Windows systems, you must have *Administrator* privileges. On Windows 2008, if you do not log in as the Administrator you will need to use elevated privileges.

How these functions are executed varies depending on the host operating system and your personal preferences. On all systems, all functions can be performed using AcuServer's command-line interface utility. Alternatively, on Windows systems, most functions can also be performed using the AcuServer Control Panel (ACP) utility.

5.2 acuserve Command-line Options and Formats

The **acuserve** command options and formats on **UNIX servers** are as follows:

```
acuserve
acuserve -access

acuserve -config [server] [-n portnum]

acuserve -info [server] [-n portnum] [-w] [-f c]

acuserve -kill [server] [-f] [-n portnum]

acuserve -register [server] [-n portnum]

acuserve -start [-c config] [-e error] [-l] [-t #] [f]
                [-n portnum]
acuserve -statistics -t [parameter] [-n port]

acuserve -unlock [server] [-a] [-c client] [-u user] [-p PID]
                 [-f FID] [-n portnum]
acuserve -version
```

The **acuserve** command options and formats on **Windows NT** and **Windows 2000** - 2008 **servers** are as follows:

```
acuserve -access
acuserve -config [server] [-n  portnum]

acuserve -info [server] [-n portnum] [-w]

acuserve -install [server] [-depends svcname] ...
                  [valid start options]

acuserve -kill [server] [-f] [-n portnum]

acuserve -query [server] [-n port]

acuserve -register [server] [-n portnum]

acuserve -remove [server] [-n port]

acuserve -start [-c config] [-e error] [-l] [-t#] [-f]
                [-n portnum]
```

```
acuserve -statistics -t [parameter] [-n port]

acuserve -unlock [server] [-a] [-c client] [-u user] [-p PID]
                 [-f FID] [-n port]

acuserve -version
```

In UNIX environments, entering the **acuserve** command with no options displays a usage list for the command.

In Windows environments, entering the **acuserve** command with no options launches the AcuServer Control Panel (ACP).

5.3 AcuServer Control Panel interface

On Windows systems, you can perform AcuServer operations using the AcuServer Control Panel (ACP). To launch the ACP, do any one of the following:

- double-click on the **acuserve** icon.
- go to Start/Programs/Acucorp 8.*x.x*/AcuServer and select “AcuServer Control Panel”.
- enter the **acuserve** command at the Windows command line.

Note: To use the AcuServer 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 AcuServer.

5.4 Creating and Maintaining the Server Access File

The *server access file* is an encrypted Vision file that contains records that define which users and client systems are permitted to use AcuServer. The *access file manager* is used to create and maintain that database. The "**acuserve** -access" command is used to start the access file manager from the command line. In the AcuServer Control Panel, the Access tab is used to perform access file manager activities (when you start the ACP, the Access tab is active by default).

For an introduction to the server access file, see **section 6.2, "The Server Access File."** For detailed instructions on using the access file manager, see **section 6.4, "Using the Access File Manager."**

5.5 Installing and Removing acuserve as a Windows Service

acuserve is designed to run as a Windows service. When you install **acuserve** as a service, **acuserve** is started automatically each time the system is booted. If you do not install **acuserve** as a service, **acuserve** will stop whenever you log off of the server.

Note: With Windows you can configure any service to be either "automatic" or "manual." As the name suggests, an automatic service starts up automatically when the system is booted; a manual service must be started by an administrator any time the system reboots. By default, **acuserve** is installed as an automatic service.

When you first install the AcuServer software, you are given the option to automatically install **acuserve** as a service. (For more information, see **section 3.3, "Installing a Windows NT/2000/2003/2008 Server."**) If you did not install the service during the initial AcuServer installation process, if you need to install the service again, or if you want to install multiple instances of **acuserve**, this section describes the process of establishing **acuserve** as a service using either the AcuServer Control Panel or the Windows command line.

Note that if you have multiple instances of the **acuserve** service running at the same time, each instance must have a unique name. You can then assign client applications to a particular instance of the **acuserve** service via the ACUSERVER_PORT variable in the *runtime* configuration file. This can help to balance the client load and improve performance.

Removing the **acuserve** service shuts down AcuServer (if the server is currently running), removes **acuserve** as a Windows NT/2000 service, and deletes any stored start-up parameters for the service.

Remember that you must be logged in as *Administrator* or as a member of the *Administrators* group to install or remove the **acuserve** service.

Please note that installing a service on a particular port resets all start-up options for the service on that port. Any start-up options that you specify when you install the **acuserve** service on a particular port are stored and used each time the service is started.

5.5.1 Installing from the Command Line

The "**acuserve** -install" command installs **acuserve** as a Windows NT/2000-2008 service. Optional arguments to "-install" include:

Option	Description
server	Specifies the name of the server machine. If no server is specified, the server is assumed to be the local host.

Option	Description
-depends	Specifies the name of a service or service group on which **acuserve** depends. This option may be repeated multiple times. The effect of the "-depends" option(s) is to establish a dependency in which **acuserve** will not start unless each service named after a "-depends" is currently installed on the server as a service. "-depends" must be followed by a space and the name of a Windows service or service group. If the name is a service group, it must be preceded by a "+" character (for example, "-depends +Alerter"). The "-depends" option and service name must come after the *server* option and before any "-start" options. If AcuServer is already installed as a service and you want to add or change dependencies, you must remove the **acuserve** service and reinstall it with the new "-depends" options. For information on service dependencies, consult your Microsoft Windows documentation.
most start options	Most "-start" options can be used with "-install". The exception is "-f", which prevents **acuserve** from being installed as a service. (Refer to **section 5.6, "Starting and Stopping acuserve,"** 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: `acuserve -install -c c:\etc\server1.cfg -le c:\tmp\ server1.log` If there are no options 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 "**acuserve** -start".

If you are running multiple instances of **acuserve**, use the "-n" option to specify a number for a particular **acuserve** service. The service will be named AcuserveX, where "X" is the number you specified with this option. If you do not use the "-n" option, the name of the service will be Acuserve.

5.5.2 Installing from the Acuserver Control Panel

1. From the Start/Programs/Acucorp 8.*x*.*x*/AcuServer menu, select **AcuServer Control Panel**, then select the Services tab.
2. Click **New**. The Service Properties dialog appears.
3. Enter the port number, the name of the configuration file, the name of the error file, the desired level of tracing, and any dependencies for the service.

Option	Description
Port	Assign a port number to this instance of **acuserve**. The port number must be an integer, such as "6524".
Configuration file	Specify the name and path of the server configuration file. If this option is left blank, **acuserve** looks for the configuration file in its default location: "\etc\a_srvcfg".
Error file	Specify the name and location of the error file that **acuserve** uses for error output. If no error file is specified, **acuserve** will attempt to direct output to a file named "acuserve.err" in the Windows system directory. If this directory cannot be opened, **acuserve** will attempt to append to a file named "acuserve.err" in the current directory. If that file doesn't exist, or if the file append fails, **acuserve** will print the message "acuserve: can't open error output file" to standard output, and **acuserve** will terminate.
Trace Level	Tracing is used to help identify a problem. Specify a nonzero value only when you are actively isolating a problem. "0" indicates no tracing; "7" is the highest level of tracing. See "**Starting from the command line**" in **section 5.6** for a detailed explanation of tracing levels.

Option	Description
Dependencies	Specify the name of a Windows service or service group on which **acuserve** depends. You may include multiple dependencies. Click **Add** to bring up a list of possible services and service groups. If you enter dependencies in this field, **acuserve** 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.

4. Click **OK** to install the service.

5.5.3 Removing the acuserve Service

From the command line, use the "**acuserve** -remove" command to remove an instance of an **acuserve** service.

Optional arguments to "-remove" include:

Option	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 **acuserve** 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 "Acuserve") is removed. See the "**Port**" option above for more information about port numbers.

From the AcuServer Control Panel, select the Services tab. Highlight the service that you want to remove and click **Remove**.

5.5.4 Determining if acuserve is installed as a Service

AcuServer provides you with two easy ways to quickly determine whether **acuserve** is currently installed as a Windows service, as well as to view any start-up parameters that have been stored for the service. From the command line, use the "**acuserve** -query" command.

Optional arguments to "-query" include:

Option	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 **acuserve** 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 "Acuserve") is queried.

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:> acuserve -query
Service 'AcuServer' (AcuServer 8.0 on the default port (6523)) exists
Start Up: Automatic
Program: 'C:\Program Files\Acucorp\Acucbl810\AcuGT\bin\acuserve.exe'
Startup arguments: '-c c:\etc\server1.cfg -le c:\tmp\server1.log'
Status: running

C:>
```

Note that the startup arguments listed are the arguments used to install the service, and may not be the arguments given to the currently running server.

From the AcuServer Control Panel, select the Services tab. This tab provides a list of all installed services. Services are listed by port number, with a summary of the arguments used by each process listed in the "Command Line" field. A green bullet next to the port number indicates that the service is currently running, while a red bullet indicates a service that is stopped.

To obtain further information about the installed services, select a service and click **Properties**. This opens the Service Properties window, which provides a more detailed breakdown of the service properties summarized in the Command Line field on the Services tab.

5.6 Starting and Stopping acuserve

You must be logged onto a UNIX server as *root* or *superuser* or onto a Windows NT, Windows 2000 - 2008 server with *Administrator* privileges to start or stop **acuserve**.

When you start **acuserve**, you can specify a number of startup options. If you start **acuserve** without specifying any options, the current defaults are used. AcuServer returns a status to the operating system indicating success or failure of the start. (Refer to **section 8.2.3** for a list of return codes.)

When you stop the **acuserve** daemon/service, **acuserve** attempts to close all open files before shutting down.

Caution: If you terminate the **acuserve** process using UNIX commands, such as "kill", or using the Windows Task Manager, you prevent **acuserve** from performing an orderly shutdown. These methods should never be used when clients are actively using AcuServer. Instead, stop **acuserve** using one of the procedures described in this section.

5.6.1 Starting from the command line

Use the "**acuserve** -start" command to start **acuserve**. On Windows NT and Windows 2000 - 2008 servers, this option also installs AcuServer as a Windows NT/2000/2003/2008 service, if it is not already installed, unless the "-f" option is specified. (If "-f" is specified, AcuServer runs as a foreground process.)

Any "-start" options that you use when you first start **acuserve** as a service/daemon are stored and used as the default startup options. If you do not specify any "-start" options, **acuserve** uses the existing defaults.

The **acuserve** program is started in background unless the "-f" option is specified.

If **acuserve** is already running, AcuServer will output the message:

```
acuserve is already running on hostname
```

A new **acuserve** process will not be started. If you want to start AcuServer with new options, you must stop and restart **acuserve**.

Note: (Windows NT/2000/2003/2008): On Windows NT and Windows 2000 - 2008 systems, it is best to specify "**acuserve** -start" with no options, so that **acuserve** uses the start-up parameters that were stored when you installed **acuserve** as a Windows service.

Optional arguments to "-start" include:

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, **acuserve** looks for the configuration file in its default location: "/etc/a_srvcfg" for UNIX or "\etc\a_srvcfg" for Windows NT/2000.
-d	Indicates that **acuserve** should track memory usage. This option is used in conjunction with the "**acuserve** -statistics" command, as described in **section 5.9**.
-e	Causes error output from **acuserve** to be output to the named file. The "-e" must be followed by a space and the path and name of the error output file. If "-e" is not specified, **acuserve** will attempt to direct error output to /dev/console in UNIX or to a file named "acuserve.err" in Windows. This file resides in the Windows NT/2000 system directory (typically c:\winnt\system32). If these directories cannot be opened, **acuserve** will attempt to append to a file named "acuserve.err" in the current directory. If that file doesn't exist, or if the file append fails, **acuserve** will print the message "acuserve: can't open error output file" to standard output, and **acuserve** will terminate.

Option	Description
-f	By default, **acuserve** runs in background. Use the "-f" option to run **acuserve** in foreground. When run in foreground, the **acuserve** process traps normal keyboard signals, such as Control+C. If combined with the "-t" option, the "-f" option causes **acuserve** 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. **Note:** If you specify the "-f" option on a Windows NT/2000 machine, **acuserve** will *not* install itself or run as a Windows NT/2000 service.
-g	Causes the error file specified with the "-e" option to be compressed using "gzip" compression. Note that the name of the error file is not automatically modified to indicate that compression is being used; you may wish to assign the file a ".gz" extension.
-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 this instance of the **acuserve** program. The "-n" must be followed by a space and then an integer, for example, "6524". This value overrides the ACUSERVER_PORT value set in the server configuration file.
	
	The "-n" option associates a port number with one instance of the **acuserve** daemon. Client applications can, in turn, be assigned to a particular instance of the **acuserve** daemon via the ACUSERVER_PORT variable in the runtime configuration file.
	
	The **acuserve** 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.
	
	Note: If you start **acuserve** on two ports, you must also specify all start-up arguments, including the configuration file, as in:
	
	`acuserve -start -n 6523 -c c:\etc\config1 [other options]`
	`acuserve -start -n 6524 -c c:\etc\config2 [other options]`

Option	Description
-t #	Turns on file tracing and transaction logging. When combined with the "-e" option, file trace and transaction messages are placed in the named error file. "#" represents the type of tracing or logging to be performed: "1" turns on file tracing, and a message is printed to standard output every time a file operation is performed. The message includes the type of file operation performed, the name of the target file or file ID, and the values of any applicable keys. "2" turns on connection information logging, and a message is output whenever a connection request is made, a disconnect is performed, or an "**acuserve** -kill" or "**acuserve** -unlock" command is processed. "3" turns on file tracing and connection information logging. "5" turns on file tracing and flushes the error file after each write operation. (File trace flushing can also be controlled with the FILE_TRACE_FLUSH server configuration variable; see **section 4.2.2, "Server Configuration Variables"**). "6" turns on connection information logging and flushes the error file after each write operation. "7" turns on file tracing and connection information logging and flushes the error file after each write operation. **Note:** You can also set these trace levels using the FILE_TRACE server configuration variable.

Please note that when AcuServer is started as a Windows NT, Windows 2000 - 2008 service, all paths used in the configuration file or on the command line are relative to the Windows NT, Windows 2000 - 2008 system directory (for example, c:\winnt\system32). For instance, if your current directory is c:\Program Files\acucorp\acucbl8.*x.x*\AcuGT\bin, and you start AcuServer with the command "**acuserve** -start -le acuserve.log", the log file will not be 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.

5.6.2 Starting on UNIX

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

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

5.6.3 Stopping from the Command Line

The "**acuserve** -kill" command causes the **acuserve** process to be halted (killed). If no server is specified, the **acuserve** process is halted on the current host; otherwise, the process is halted on the named host.

Unless the "-f" option is specified, **acuserve** prompts for confirmation before the halt action is executed, as shown below.

```
Shutting down acuserve on: condor
There are 0 files in use by acuserve on: condor
Do you really want to shut down acuserve [N]?
```

Respond by entering "Y" or "N".

Optional arguments to "-kill" include:

Option	Description
server	Specifies the name of the server machine on which to kill the **acuserve** process. If no server is specified, the server is assumed to be the local host.
-f	Causes the **acuserve** process to terminate immediately, without prompting for confirmation. "-f" should be used when "**acuserve** -kill" is included in a program or script.

Option	Description
-n	Identifies a particular instance of the **acuserve** 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.

Note: If you issue the "**acuserve** -kill" command while an ACUCOBOL-GT Web Runtime is connected to the server, the connected client program (and browser window) will hang.

5.6.4 Starting and Stopping from the AcuServer Control Panel

1. From the Start/Programs/Acucorp 8.*x*.*x*/AcuServer menu, open the **AcuServer Control Panel** and select the Services tab.

   You will see a list of port numbers and startup options for installed **acuserve** services. Services that are currently running are marked with a green bullet, and services that are stopped are marked with a red bullet.

2. To start a service, highlight the service and click **Start**.

   To change the startup properties for a service, highlight the service and click **Properties**. Any modifications that you make to the startup properties for that service are stored and used as the default startup properties the next time the service is started.

   Note: You cannot start **acuserve** in the foreground using the ACP. If you want to start **acuserve** in the foreground, use the command "**acuserve** -start -f" at the Windows command line, as described earlier in this section.

3. To stop a service, highlight the service and click **Stop**.

5.7 Changing acuserve Properties

Both the "**acuserve** -config" command and the Config tab of the AcuServer Control Panel allow you to change the configuration of a running **acuserve** process. You can use these functions, for instance, to turn tracing on or off or to change configuration variables such as PASSWORD_ATTEMPTS.

For a list of valid variables, refer to **section 4.2.2, "Server Configuration Variables."**

5.7.1 Changing Properties from the Command Line

When you enter the "**acuserve** -config" command, you start communicating with the process 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:

Option	Description
server	Specifies the name of the server machine on which the **acuserve** process is to be configured. If no server is specified, the server is assumed to be the local host.
-n	Identifies a particular instance of the **acuserve** 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 "srvcfg>" is displayed. At this prompt, enter any of the following commands:

Command	Description

GET *variable-name*	Fetches and displays the value of a single variable. For example:
	`srvcfg> get max_files`
	`MAX_FILES: 32`

Command	Description

SET *variable-name new-value*	Sets the value of a variable. For example:
	`srvcfg> set max_files 320`
	Note that if given an invalid variable name, the **acuserve** process sets a variable of that name with the given value, even though that variable will not have any effect on **acuserve**.
	Be aware, also, that some variables are checked by **acuserve** only at initialization, so changing these variables on a running instance of the service will have no effect. For example, if you try to set ACUSERVER_PORT, SERVER_IP or SERVER_NAME with the "**acuserve** -config" command, you receive an error message similar to the following:
	`srvcfg> set SERVER_IP 192.215.170.34`
	`Setting the SERVER_IP has no effect`
	`srvcfg> get SERVER_IP`
	`SERVER_IP: 192.215.170.34`
	Note that the GET command will show the new value, but the value will not be used.

LIST [(>|>>) *file*]	Lists the names and current values of all variables of which AcuServer is aware, including some that do not directly affect AcuServer. 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	Exits the configure mode.

5.7.2 Changing Properties from the AcuServer Control Panel

1. To view configuration information, select the Config tab in the AcuServer Control Panel.

   Initially, the Config tab appears empty, because no server has been selected.

2. Click **Query**, then select the name of the server machine and the port on which **acuserve** is running.

3. Click **OK**. The configuration file entries for the service that you specified now appear on the Config tab.

4. If you want to make changes to the way that **acuserve** is configured, use the **New** button to add configuration variables or use the **Modify** button to remove or change configuration variables and their parameters.

   See the description of SET in the section titled "**Changing Properties from the Command Line**" for information about limitations and effects of changing variables for an active service.

5.8 Checking System Status

System status reports, which are available both from the command line ("-info) and within the ACP, include both a list of all open files and the total number of files open for a particular instance of **acuserve**. Connections that do not have any open files are also listed.

Status reports contain the following sections:

- The Client field identifies the client machine associated with the open file.
- The User field identifies the client user name associated with the open file.

- The PID is the *process ID* of the client application on the client system. Because service requests originating on different clients may have the same PID, the command "**acuserve** -unlock" requires that the PID be qualified with a client name.
- The TY field indicates the type of open data file: (I)ndexed, (R)elative, (S)equential, (O)bject, or (F)ile.
- The FID is a unique file process ID associated with each open file on the server (FIDs are guaranteed to be unique).
- The Opened field shows the date and time that a file was opened. To see this field from the command line, you must use the "-w" option.
- The File Name field displays the name of the open file.

Note that the ACP includes statistics for servers that are controlled by the server that was queried in a master/slave relationship, but it does not indicate which file is open by which server. Rather, it lists all the files open by the master and its slaves.

5.8.1 Generating Status Reports from the Command Line

Use the command "**acuserve** -info" to generate an AcuServer system status report.

Optional arguments to "-info" include:

Option	Description
server	Specifies the name of the server machine whose status is to be reported.
-f	Returns the full client name and user name, up to 170 characters. Without this option, the client and user names can contain up to only 14 characters each.
-n	Identifies a particular instance of the **acuserve** 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, status for the default port is reported.
-w	Generates a wide report that includes the date and time that each file was opened and shows file names as long as 190 characters.

The default status report has the following format:

```
Client       User         PID     TY   FID      File Name
----------   ----------   -----   --   ------   -------------------------
robin        chris        1813         <No open files for this connection>
starling     bernie       1281    I    5d7e8    /usr2/bsmi.../dat/IDX.DAT
starling     bernie       1281    S    5ce08    /usr2/bsmi.../dat/SUP.DAT

2 file(s) in use by acuserve on: condor
Port Number: 6523, AcuServer version 8.0
```

The File Name field displays up to 28 characters by default. To generate a report that displays file names of up to 190 characters and includes the date and time that the file was opened, use the "-w" option. Should the path and file name exceed 28 or 190 characters, the report will attempt to display all of the file name, preceded by the file's parent directory, preceded by the root directory. Path-name components that must be omitted are represented by an ellipsis (three dots). See the preceding example.

5.8.2 Checking Status from the AcuServer Control Panel

1. Open the AcuServer Control Panel and select the Info tab. By default, the Info tab originally appears blank.
2. Click **Query** and specify the server and the port on which the **acuserve** service is running.
3. The server name and port number appear at the top of the Info tab, and information about any clients currently using **acuserve** appears in the table in the center of the tab.

   To refresh the information on the Info tab, click **Refresh**.To activate an auto-refresh mode, click the **Auto Refresh** checkbox and enter the number of seconds to wait before a refresh. In the lower right of the Info tab, a text message displays the time until the next auto refresh.

5.9 Tracking Server Statistics

AcuServer has an option to collect and report server statistics. When the COUNT_STATISTICS configuration variable (described in **section 4.2.2**) is set to "true", you can track memory usage, network reads and writes, socket select calls, and time spent in kernel, user, and idle modes. You can also track access to Vision, relative, and sequential data files, recording the number of file reads, writes, rewrites, deletes, and so on. To get memory information, start AcuServer with the new "-d" option ("**acuserve** -start -d"). If you do not specify the "-d" parameter, memory is not tracked and is reported as "0" at all times.

To get a detailed server history, use the command:

```
acuserve -statistics -t [parameter] [-n port]
```

The available parameters are listed below.

Parameter	Short Description	Details
ALL (default)	All statistics	Returns all available statistics for an instance of **acuserve**
FILE	All data file statistics	Includes file statistics for all Vision, relative, and sequential files
MEMORY	Memory used	Total number of bytes of memory used by AcuServer
NETWORK	Users added	Total number of users who have connected
	Users removed	Total number of users who have disconnected
	Files opened	Total number of files that have been opened
	Files closed	Total number of files that have been closed
	Network bytes read	Total number of bytes read by AcuServer
	Network bytes written	Total number of bytes written by AcuServer

Parameter	Short Description	Details
	Immediate select calls	Total number of times select() was called with no timeout
	Immediate select sockets	Total number of sockets returned when select() was called with no timeout
	Timed select calls	Total number of times select() was called with a timeout
	Timed select sockets	Total number of sockets returned when select() was called with a timeout
TIME	% Kernel time, last minute	Percentage of time spent in kernel mode in the last minute
	% Kernel time, last 5 minutes	Percentage of time spent in kernel mode in the last 5 minutes
	% Kernel time, last 15 minutes	Percentage of time spent in kernel mode in the last 15 minutes
	% User time, last minute	Percentage of time spent in user mode in the last minute
	% User time, last 5 minutes	Percentage of time spent in user mode in the last 5 minutes
	% User time, last 15 minutes	Percentage of time spent in user mode in the last 15 minutes
	% Idle time, last minute	Percentage of idle time in the last minute
	% Idle time, last 5 minutes	Percentage of idle time in the last 5 minutes
	% Idle time, last 15 minutes	Percentage of idle time in the last 15 minutes
	Uptime (days hours:minutes)	Amount of time the server has been running
VISION	Vision create	Total number of Vision files created
	Vision open	Total number of Vision files opened
	Vision close	Total number of Vision files closed
	Vision read	Total number of Vision records read dynamically

Parameter	Short Description	Details
	Vision read next	Total number of Vision records read sequentially (next)
	Vision read previous	Total number of Vision records read sequentially (previous)
	Vision write	Total number of Vision records written
	Vision rewrite	Total number of Vision records rewritten
	Vision delete	Total number of Vision records deleted
	Vision start	Total number of Vision files started
	Vision info	Total number of Vision info calls
	Vision remove	Total number of Vision files removed
	Vision rename	Total number of Vision files renamed
	Vision copy	Total number of Vision files copied
	Vision unlock	Total number of Vision records unlocked
RELATIVE	Relative create	Total number of relative files created
	Relative open	Total number of relative files opened
	Relative unlock	Total number of relative records unlocked
	Relative close	Total number of relative files closed
	Relative read	Total number of relative records read dynamically
	Relative read next	Total number of relative records read sequentially (next)
	Relative read previous	Total number of relative records read sequentially (previous)
	Relative start	Total number of relative files started
	Relative write	Total number of relative records written
	Relative delete	Total number of relative records deleted

Parameter	Short Description	Details
	Relative rewrite	Total number of relative records rewritten
	Relative remove	Total number of relative files removed
	Relative rename	Total number of relative files renamed
	Relative copy	Total number of relative files copied
	Relative info	Total number of relative info calls
SEQUENTIAL	Sequential create	Total number of sequential files created
	Sequential open	Total number of sequential files opened
	Sequential close	Total number of sequential files closed
	Sequential read	Total number of sequential records read
	Sequential write	Total number of sequential records written
	Sequential rewrite	Total number of sequential records rewritten
	Sequential remove	Total number of sequential files removed
	Sequential rename	Total number of sequential files renamed
	Sequential copy	Total number of sequential files copied

Note: Information about select() can be used to determine if clients are waiting a long time before being noticed by AcuServer. When a client has to wait, the number of sockets returned by select() without a timeout will be large, in general, and also large in comparison to the number of calls to select() with a timeout. If the number of sockets returned by calls to select() without a timeout is small, clients are being serviced immediately.

5.10 Closing Stranded Files

Software and hardware errors can sometimes result in the stranding of records and files accessed via AcuServer. AcuServer is equipped to detect and close files opened by Windows clients using AcuServer's dead connection detection mechanism (see **section 8.1, "Machine Failures"**). AcuServer also detects and closes files belonging to client applications that terminate abnormally with a Control + C or "kill" signal. Other application or system failures may also result in stranded records and files. For example, if a UNIX client suddenly crashes, all files opened by that client remain open and all records locked by that client remain locked. If the client opened a file in a mode that prevents other applications from accessing the file (for example, EXCLUSIVE), that file remains unavailable to other applications.

5.10.1 Closing Files from the Command Line

The "**acuserve** -unlock" command provides a mechanism for closing stranded files from the command line.

Before you use "-unlock", it is best to generate a current AcuServer status report using "**acuserve** -info". The status report provides a complete list of open files on the server, including client name, user name, PID, FID, file type, and the file name of each open file. With information from this report, you can use "-unlock" to close specific files.

The "-unlock" command syntax is:

```
acuserve -unlock [server] [-a] [-c client] [-u user]
                 [-p PID] [-f FID] [-n portnum]
```

"-unlock" has two modes of operation: an **interactive mode**, in which the user is prompted for information about the files to be closed; and a **command-line mode**, in which the user specifies all of the requisite information as additional arguments on the command line.

Optional arguments to "-unlock" include:

Option	Description
server	Specifies the name of the AcuServer server. If no server is specified, the server is assumed to be the local host.

Option	Description
-a	Causes all open files on the server to be immediately closed. "-a" does *not* ask the user for confirmation before closing all files.
-f *FID*	Causes the file associated with the specified FID to be unlocked and closed. "-f" must be followed by a space and the FID of an open file.
-c *client*	When "-c" is used with no other options, **acuserve** unlocks and closes all files and sockets held open for the named client on the current host. "-c" must be followed by a space and the name of an AcuServer client. Use "-c" in combination with the *server* option to specify a server other than the current host. "-c" can be qualified by the "-u" or "-p" options.
-u *user*	Used as a qualifier to the "-c" option. "-u" must be followed by a space and the name of an AcuServer user. The "-c *client* -u *user*" combination causes **acuserve** to close all files and sockets associated with the named user on the named client.
-p *PID*	Used as a qualifier to the "-c" option. "-p" must be followed by a space and the PID of a client process. The "-c *client* -p *PID*" combination causes **acuserve** to close all files and sockets associated with the specified PID on the named client.
-n *port*	Identifies a particular instance of the **acuserve** program by port number. The "-n" must be followed by a space and then an integer, for example, "6524". This option causes only those files associated with the specified instance of the **acuserve** program to be unlocked and closed. If no port number is specified, files for the default port are unlocked and closed.

Interactive mode

To initiate interactive mode, enter:

```
acuserve -unlock [server] [-n port]
```

The absence of other options triggers interactive mode. If the *server* option is omitted, the server is assumed to be the current host. If the *-n port* option is omitted, the port number is assumed to be "6523".

Interactive mode prompts the user for specific information that describes the files to be closed (any locks are removed). The more general the information entered, the larger the classification of files to be closed. If you specify a client name, user name, or PID, in addition to closing all associated files, all

associated sockets are closed. For example, if only a client name is given, then all files and sockets associated with that client are closed (see the examples that follow).

After prompting for information, interactive mode redisplays the information entered and prompts for confirmation before closing any files and sockets.

The following examples demonstrate how interactive mode can be used to close specific files and classes of files. When you enter interactive mode, the unlock program displays the following text:

```
This program allows you to force acuserve to unlock and close
a single file or group of files.  To unlock and close a single
file, enter the file's unique hexadecimal identifier (FID).
Leave the FID blank if you want to specify files by client
name, user name or process identifier (PID).
Enter FID []:
```

To close a single file, enter the file ID at the prompt.

```
Enter FID []: 5fb58
```

The unlock program then redisplays the information and prompts for confirmation.

```
Server : condor
Port Number : 6523
FID : 5fb58
Is this correct [N]:
```

If you enter "y" (or "Y", "YES", "yes") the program will unlock and close the file and report the results:

```
acuserve: unlocked and closed 1 file(s) on: condor
```

To close all files open on the server and all sockets held open by this instance of acuserve, leave the FID prompt blank and accept the default value "ALL" for the client machine name prompt:

```
Enter FID []:
Enter client machine name [ALL]:
```

The program then redisplays your entry and prompts for confirmation:

```
Server : condor
Port Number : 6523
Client : ALL
Is this correct [N]:
```

To close all files and sockets held open by a single client, leave the FID prompt blank, specify the client machine name at the client machine name prompt, and accept the default value "ALL" at the user name prompt:

```
Enter FID []:
Enter client machine name [ALL]: starling
Enter name of user [ALL]:
```

The program then redisplays your entry and prompts for confirmation:

```
Server : condor
Port Number : 6523
Client : starling
User : ALL
Is this correct [N]:
```

To close all files and sockets associated with a specific user on a specific machine, leave the FID prompt blank, specify the client machine name at the client machine name prompt, specify the user name at the user name prompt, and accept the default value "ALL" at the PID prompt:

```
Enter FID []:
Enter client machine name [ALL]: starling
Enter name of user [ALL]: bernie
Enter PID [ALL]:
```

The program then redisplays your entry and prompts for confirmation:

```
Server : condor
Port Number : 6523
Client : starling
User : bernie
PID : ALL
Is this correct [N]:
```

To close all files and sockets associated with a single PID, leave the FID prompt blank, specify the client machine name at the client machine name prompt, specify the user name at the user name prompt, and specify the PID at the PID prompt:

```
Enter FID []:
Enter client machine name [ALL]: starling
Enter name of user [ALL]: bernie
Enter PID [ALL]: 1726
```

The program then redisplays your entry and prompts for confirmation:

```
Server : condor
Port Number : 6523
Client : starling
User : bernie
PID : 1726
Is this correct [N]:
```

Command-line mode

Command-line mode requires that the description of the files to be closed be specified as options to "-unlock" on the command line. The unlock and close actions are carried out immediately, *without* a prompt for confirmation, and the results are reported. Command-line mode is well suited to use in shell scripts and batch files that are run without user input.

To unlock and close a specific file you would enter:

```
acuserve -unlock [server] -f FID
```

The unlock program closes the file and reports its actions:

```
acuserve: unlocked and closed 1 file(s) on: condor
```

To unlock and close all files and sockets associated with a PID, enter:

```
acuserve -unlock [server] -c client -p PID
```

To unlock and close all files and sockets associated with a single user on a client, enter:

```
acuserve -unlock [server] -c client -u user
```

To unlock and close all files and sockets associated with a client, enter:

```
acuserve -unlock [server] -c client
```

To unlock and close all files and sockets held open by the server, enter:

```
acuserve -unlock [server] -a
```

Note that closing *all* files requires the "-a" option.

5.10.2 Closing Files from the AcuServer Control Panel

If you are using the AcuServer Control Panel, you can unlock and close stranded files from the Info tab.

1. Click **Query** and select the server name and port number on which **acuserve** is currently running.
2. Select the name of the file or files you want to unlock and close, then click **Unlock**.

   If you want to close all files opened by a user process, select the name of any file opened by that process and click **Disconnect**.

Note: The **Disconnect** button in the Info tab of the AcuServer Control Panel performs the same function as entering the command "**acuserve** -unlock -u *user* -c *client* -p *PID*" from the command line.

5.11 Checking Version Information

From the command line, enter "**acuserve** -version" to see AcuServer version information. "-version" must be the only argument on the command line.

To check version information from within the AcuServer Control Panel, click on the AcuServer icon in the left corner of the title bar to open the system menu, then select **About**.

5.12 Registering Servers

When using the AcuServer **load-balancing feature (see ACUSERVER_MASTER_SERVER)** if the master server dies unexpectedly, the "-register" command can be used to re-register the secondary or subordinate servers with a newly started master server. This avoids having to shut down and restart the entire system.

The syntax for the command is:

```
acuserve -register [server] [-n portnum]
```

This command tells the secondary acuserve running on “server” and listening on port “portnum” to re-register itself with the master server. Note that it registers with the master given in the configuration file when it was started. “server” defaults to the local host, and “portnum” defaults to the usual port.

6 System Security

Key Topics:

Security Overview 6-2
The Server Access File 6-4
Access Records 6-6
Using the Access File Manager 6-9
AcuServer Connection Logic 6-18
Encryption 6-21

6.1 Security Overview

System security for AcuServer® file server software is designed to address two fundamental security issues:

1. **controlling access to data files**

This is addressed in two ways: first, via the AcuServer *server access file* (discussed in detail in this chapter), and second, through the standard UNIX or Windows NT/2000-2008 file access provisions.

Whether a user of AcuServer can access to a given file depends on two things: (1) the user ID assigned the requester in the server access file, and (2) either the Windows NT/2000-2008 security set up for your files, or the UNIX ownerships and permissions set on the particular file.

On both Windows or UNIX networks, you have the option to use your operating system security rather than AcuServer system security. It is recommend that you use the operating system security and is essentially required when using Windows 2008. By setting the SECURITY_METHOD configuration variable on both the client and the server, you can override the server access file and use the full range of native security features on files and directories on the server instead. See SECURITY_METHOD in **section 4.2.2, "Server Configuration Variables,"** for information and considerations.

2. **preventing unauthorized use of AcuServer to perform privileged activities (such as modifying privileged files)**

This is addressed through strict enforcement of the security measures that you have established through the server's operating system.

On a Windows NT, Windows 2000 - 2008 server, AcuServer system security is designed to work with files that reside on an NTFS (NT file system). (AcuServer can work with a FAT file system, but the files are less secure and no longer supported.)

When using the NTFS, you may set read and write access permissions on your files by using the Windows security features. Please refer to your Windows documentation for more information about NTFSs and security procedures. Make sure that the AcuAccess file and the "a_srvcfg" file can be written only by those accounts and groups that you want to have write privileges.

NTFSs may also have shared directories. The permissions on the shared directory operate in addition to any NTFS permissions you have established. Shared directory permissions specify the maximum access allowed.

When AcuServer is running as a Windows NT/2000-2008 service, it usually belongs to an implicit group called "SYSTEM." Make sure that the "SYSTEM" group (or whichever group that you are using for your **acuserve** services) is added to your file permissions with "Full Control."

Files created by AcuServer are owned by the Administrators group and allow "Full Control" for "SYSTEM" and "Administrator." "Everyone" is given the permissions specified by the third digit in the *umask* in the AcuAccess file. UNIX ownerships and permissions can be set on key AcuServer 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 recommend against the inclusion of such entries.

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

- the AcuServer server access file (the database of authorized and excluded AcuServer users)
- the Windows NT, Windows 2000 - 2008 security protections set up for the **acuserve** executable file, server configuration file, server access file, and remote data files and directories. Recall that the NTFS is more secure.
- the UNIX ownerships assigned to the **acuserve** 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 **acuserve** executable file, server configuration file, server access file, and remote data files and directories.

UNIX ownerships and permissions on the **acuserve** executable, server configuration file, and server access file are specified in **section 2.4.1**. These specifications must be strictly maintained. If the ownerships and permissions are more permissive than those specified, AcuServer will not start.

6.2 The Server Access File

The foundation of AcuServer system security is the *server access file*. The server access file is an encrypted Vision file, named "AcuAccess" by default, which is located in the "/etc" directory on UNIX servers and the root drive, which is normally the "c:\etc" directory on Windows NT, Windows 2000 - 2008 servers. You may rename the access file, and you can have multiple access files (for multiple instances of **acuserve**, for example) if desired.

The server access file contains one or more *access records*. These records define which users of which clients are permitted access to AcuServer.

Caution: If you have upgraded from a version of AcuServer prior to 6.0.0, the server will detect and convert existing AcuAccess files the first time that they are opened. Updated AcuAccess files are not compatible with earlier versions of AcuServer. If you are operating in an environment that includes AcuServer version 8.*x*, 7.*x*, or 6.*x* mixed with earlier versions of AcuServer, you must use duplicate AcuAccess files. We do not recommend maintaining a mixed environment.

The server access file is designed to support a wide range of access security, from very open to very restrictive. You choose the level of security best suited to your needs.

Note: It is recommend that you use native system security rather than AcuServer 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. See SECURITY_METHOD in **section 4.2.2** for more information.

Access records may include wild cards that allow all clients or all users (except *root* under UNIX and *Administrator* under Windows NT, Windows 2000 - 2008) access to AcuServer. You can also create individual access records for each user of each client, as well as individual records listing users who are explicitly excluded from accessing files.

The individual access records allow you to specify the user ID that AcuServer 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, which the application or user must match before AcuServer will establish a connection. If this password is set to "*", the user is explicitly denied access to AcuServer.

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

Creation and modification of the server access file requires *root* privileges on UNIX, and *Administrator* privileges on Windows NT, Windows 2000 - 2008.

On UNIX servers, the access file must be owned by *root* and 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*, AcuServer will not start. On Windows NT and Windows 2000 - 2008 servers, the access file must be owned by *Administrator* or the *Administrators* group and cannot be writable by anyone without *Administrator* privileges. If the access file does not exist, is not owned by *Administrator* or the *Administrators* group, or is writable by users without *Administrator* privileges, AcuServer will not start.

6.3 Access Records

The server access file contains one or more *access records*. You must create access records for each user, whether you are using AcuServer's system security or native system security. Each access record comprises five fields:

Client Machine Name:	The name of the client system:
Client Username:	The user's login name on the client system
Local Username:	The local user name that AcuServer will use when fulfilling requests for the client user.
Password:	Optional password protection. When used, the requester must supply the password specified in this field. Ignored when you use the Windows NT security NAMED-PIPE option.
Umask:	A three-digit file creation mask. The umask is used when AcuServer creates a new file for the requester. The default value is "002".

From the command line, a typical server access record might look like:

```
Client Machine Name    Client Username    Local Username    Password    Umask
-------------------    ---------------    --------------    --------    -----
starling               bernie             bsmith             <none>      002
```

Using the AcuServer Control Panel in Windows, the record might look like the following:

Client Machine Name	Client Username	Local Username	Umask
starling	bernie	bsmith	002

This record will allow user *bernie* to connect from machine *starling*. AcuServer will use the local user name *bsmith* (Bernie's account on the file server) when executing requests for *bernie*. No password is required. Any files created for *bernie* will be created with a umask of 002 (*read* and *write* permissions for owner and group; *read*-only permissions for others).

Three fields, Client Machine Name, Client Username, and Local Username, can have a *wild card* value 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

When the string “same as client” is specified in the Local Username field, if the 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.

On Windows client systems, Client Username is set to the name given by the user at logon. If the operating system does not provide a logon, or if the user bypasses the logon, the value of the environment variable USER is applied. If USER is not defined, the value of the environment variable USERNAME is applied. Note that the values assigned to these variables are case-sensitive. Be sure that the case used in the AcuAccess file matches the case of the value set in the variable. If neither USER nor USERNAME is defined, the literal string “USER” is used.

For illustrative purposes, here is a set of common access records as seen from the command line:

```
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
falcon                 rjones             <same as client>  *           002
starling               felice             <same as client>  <none>      002
starling               baxter                               ......      002
swallow                hartley            hartley           <none>      002
swallow                                   acct              <none>      002
```

Here is the same set of access records seen in the AcuServer Control Panel's Access tab:

Client Machine Name	Client Username	Local Username	Umask
president-pc	diamond		002
robin			002
starling	baxter		002
starling	felice		002
support-pc		techie	002
swallow		acct	002
swallow	hartley	hartley	002
warehouse-pc			002

These entries are interpreted as follows:

The entry for *support-pc* allows any user of *support-pc* to use AcuServer. AcuServer 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 AcuServer. Because the Local Username field is empty, AcuServer 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 AcuServer. For this record to match, the environment variable USERNAME or USER must be defined with the value "diamond". AcuServer will also 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 AcuServer. If the requester has an account on the server by the same name, AcuServer will use that name; otherwise, AcuServer will use the value of DEFAULT_USER.

The entry for *falcon* specifies that user *rjones* will not be allowed to access files through AcuServer. If *rjones* tries to access an AcuServer file, she will receive an error 9D,103, indicating that access is denied.

One entry for *starling* allows user *felice* to access AcuServer. AcuServer will follow the same rules as the previous entry to assign a local user name.

The other entry for *starling* allows user *baxter* to access AcuServer. AcuServer 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.

One entry for *swallow* allows user *hartley* to access AcuServer. AcuServer will use the local user name *hartley* when executing requests for *hartley*.

The other entry for *swallow* allows all users of *swallow* to access AcuServer. AcuServer 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).

6.4 Using the Access File Manager

You create and maintain the server access file using the *access file manager* utility or, if you're running under Windows, the Access tab of the AcuServer Control Panel (ACP). For an introduction to the ACP, refer to **section 3.7, "Using AcuServer's Graphical User Interface for Windows."**

- To start the access file manager utility from the command line, use the command "**acuserve** -access".
- To start the access file manager utility using the graphical interface, go to Start/Programs/Acucorp 8.*x*.*x*/Acuserver to open the AcuServer Control Panel.

The access file manager allows you to:

- create an access file
- add an access record
- remove an access record
- modify an access record
- display an access record(s)

If you are managing access records from the command line, note that 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, simply press **Enter**. In the following example, "/etc/AcuAccess" is the default value.

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

6.4.1 Creating or Opening an Access File

Whether you use the command line or the AcuServer Control Panel (ACP) to create or open an access file, the procedure is similar. You first specify a path and file name. If the file does not exist, you choose whether to create a new file or re-enter path and file-name information.

6.4.1.1 Creating and Opening from the Command Line

1. Use the "**acuserve** -access" command to open the file access manager.
2. Enter the path and name of the server access file:

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

3. If the specified server access file is not found, either type "Y" to create the file or "N" to re-enter the file name.

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

4. The manager opens the access file and displays a menu of five options:

   ```
   Server Access File Options
   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]:
   ```

6.4.1.2 Creating and Opening from the AcuServer Control Panel

To open an existing AcuAccess file using the Access tab of the AcuServer Control Panel, click **Open**. Select the access file that you want to edit from the list of files provided, or browse to the correct directory.

To create a new access file:

1. Select the Access tab of the AcuServer Control Panel and click **Open**.
2. Browse to the directory where the new access file will reside (the default is “c:\etc”).
3. Type the name of your new access file and click **Open**.

   A dialog box appears, confirming the name and location of the new access file.
4. Click **Yes**.

You can now add records to your new access file.

6.4.2 Adding an Access Record

To add an access record, you must assign a value to each of the five access record fields:

- Client Machine Name

  The “*” symbol indicates that the record will match all clients for which there are no other records. (This will be the default setting for all clients that do not have a specific record.) You can also enter the official machine name for a client machine. You cannot use alias names in this field.

- Client Username

  If you do not specify a client username, leaving the field blank, any user name will match. For Windows clients that do *not* define a USERNAME or USER environment variable, this field should be left blank or set to “USER” (see **section 6.3, “Access Records”**).

- Local Username

  The Local Username is the name that AcuServer 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 (AcuServer returns an error 9D,103).

  If the local username is the same as the client user name, leave the Local Username field blank.

- Password

  Inclusion of a password is optional. AcuServer ignores any password included in the AcuAccess record when you use the Windows NT security named pipe option rather than AcuServer system security.

  Passwords can be 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 32−126). Delete, escape, and other non-printable characters are not allowed.

  To prevent a user from attempting to log in to AcuServer, assign that user the password "*". The excluded user will receive an error 9D,103, indicating that access is denied.

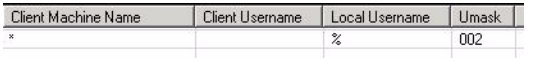
- Umask

  The umask is a three-digit code that sets the *read* and *write* permissions on new files created for the requester by AcuServer. For more about umask, see the end of and your UNIX operating system documentation.

If you accept all of the defaults when creating the record, and you are using the command line, the entry will look like:

```
Client Machine Name    Client Username    Local Username    Password    Umask
-------------------    ---------------    --------------    --------    -----
*                                         <same as client>  <none>      002
```

If you are using the ACP and accept all of the defaults, the entry will look like:



Client Machine Name	Client Username	Local Username	Umask
*		%	002

This is the most permissive access record that can be created. This record will match any client and allow any user to connect to the server, provided that either:

- the user has an account of the same name on the server
- 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 most any level in between.

6.4.2.1 Adding a Record from the Command Line

1. To add an access record, start the access file manager and select option [1] from the main menu.

   You are prompted to enter a client machine name. 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 [*]:
   ```

2. Enter a client machine name or accept the default value.

   You are prompted to enter the client username.

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

3. Enter a client user name or accept the default value.

   You are prompted to enter the local username.

```
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]:
```

4. Enter a local user name or accept the default value.

   You are prompted to enter a password.

5. If you want the specified user to have a password, enter the password. The password characters are not echoed on the screen.

   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:
   ```

   Should the password verification fail, you will see the following message:

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

   You are prompted to enter a umask.

   ```
   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]:
   ```

6. Enter a umask value or accept the default value. If an invalid umask value is entered, you will 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...
   ```

6.4.2.2 Adding a Record from the AcuServer Control Panel

To add an access record using the graphical interface, open the AcuServer Control Panel and open the server access file to which you want to add a record.

1. Click **New**. The Access User screen appears.
2. Enter the Client Machine Name, Client User Name, Local User Name, Password (if any), and Umask value.
3. Click **OK** to accept the record.

6.4.3 Removing an Access Record

To remove a record from the command line, select option [2] from the access file utility main menu. The manager will present two prompts: one for the client machine name and one for the client user name. The prompts will look something like this:

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

If a matching record is found, it is removed.

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

To remove a record from the Access tab of the AcuServer Control Panel, highlight the applicable record and click **Delete**.

6.4.4 Modifying an Access Record

Using either the command-line access file utility or the AcuServer Control Panel, you can modify the local user name, password, and umask value for an existing access record.

If the client machine name or client user name changes, you must delete the existing access record and create a new one.

6.4.4.1 Modifying a Record from the Command Line

Select option [3] from the access file utility main menu. The access file manager will present a series of prompts. The first two prompts ask for the client machine name and client user name, followed by a display of the matching record, as shown in the following example:

```
Enter official client machine name:  president-pc
Enter client user name:  diamond

Here is the record that was found:

Client machine name:  president-pc
Client user name   :  diamond
Local user name    :
Password           :  <none>
Umask              :  002
```

The next three prompts allow you to change the local user name, the password, and the umask.

```
Do you want to modify the local user name [N]:
Do you want to modify the password [N]:
Enter a new umask [002]:
```

After the umask value is entered, the record is updated.

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

6.4.4.2 Modifying a Record from the AcuServer Control Panel

1. On the Access tab of the AcuServer Control Panel, highlight the record that you wish to change and click **Modify**.

   The Access User box shows the Client Machine name and Client User Name and provides text boxes in which you can modify the Local User Name, Password, and Umask entries.

2. Enter the changes you wish to make and click **OK** to save the modified access record.

6.4.5 Displaying an Access Record

To display one or all of the access records from the command line, select option [4] from the access file utility 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]? y
```

If yes, the manager will respond with the following prompt:

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

If you want to see an individual record, respond no ("n") to "Display all records?" and the manager will display the following prompts:

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

If a match is found, the record will be displayed, otherwise the manager will state that no match was found and return to the main menu.

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

If you want to copy all of the records to a file, respond no to the "Display to the screen?" prompt. The manager will then prompt for a filename.

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

Enter a file name, or accept the default. The manager will copy the access records to the named file.

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

In the AcuServer Control Panel, the access records contained in a specific AcuAccess file are automatically displayed when you open the server access file. Note that regardless of whether or not passwords have been specified in the AcuAccess file, the list of records shown in the ACP does not include a Password column.

6.4.6 Exiting the Access File Manager

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

To exit the graphical manager, click the close button at the top right corner of the AcuServer Control Panel dialog, or click **Cancel**. Note that clicking Cancel will not undo any changes that you have made to the AcuAccess files.

6.5 AcuServer Connection Logic

The AcuServer connection validation logic is described here to clarify the use of the server access file and the DEFAULT_USER configuration variable.

When a client process (running application) makes its first request to AcuServer, AcuServer performs the following access validation procedure whether AcuServer system security or native system security is being used.

To validate the requester's access privileges, AcuServer:

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, and the named-pipe form of security is turned on (via the SECURITY_METHOD variable), and the client user has an account on the server, AcuServer automatically grants the user permission to connect. The AcuAccess file does not set the client user's local username, nor does it determine whether the client is required to enter a password.

When a match is found and the LOGON form of security is turned on, AcuServer attempts to use the value of the matching password field in the AcuAccess file to log the user on. If the password isn't valid or the password field is empty, the user is prompted to enter a password. If a valid password is given, the requester is logged on, otherwise the connection is refused.

When a match is found and AcuServer system security is being used:

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.

If the Local Username is valid and the password field is defined, a message is sent back to the requester asking for a password. For more about password handling, see **section 6.5.1, "Passwords."**

When the client process terminates, the client-server connection is broken. New client applications requesting AcuServer services will go through the verification process to establish a connection.

6.5.1 Passwords

Regardless of whether you're using AcuServer's built-in security or native system security, defining passwords in the access records of your AcuAccess file can be very useful.

If you are using LOGON security, defining the password field to hold the user's current Windows NT domain password can make the login process transparent. When a requester initiates a connection to AcuServer, AcuServer uses the value of the requester's password field, if defined, to log the requester onto the Windows NT domain. If the password is not valid, the user is prompted to enter a password. If the user enters a blank password, that entry counts as one password attempt, but AcuServer does not give an error message, and the user can try again. Note that the named-pipe form of security does not use the password field. See **SECURITY_METHOD** in section 4.2.2 for more information.

If you are using UNIX system security, you can simplify the login process by defining the password field to hold the user's current system password.

If you are using AcuServer's system security, you can use passwords to achieve added security. When a password is assigned to an entry in the server access file, requesters who match that entry must return a matching password to AcuServer. The client application has two options for acquiring and sending a password back to AcuServer:

- Using the *Acu_Client_Password* program variable
- Having the client runtime prompt users to enter a password

Option one: 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 AcuServer 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 AcuServer 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 (MOVE) 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).

Option two: user-entered password

If Acu_Client_Password is not defined, the client runtime will open 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, 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"). If you have a situation in which a client connects to **acuserve** many times in succession, you may want to set the *server*_PASSWORD and *server_port*_PASSWORD environment variables on the client. This allows AcuServer to verify the client password automatically, instead of repeatedly prompting the user to enter a password. For more information about setting these environment variables, see Appendix H in Book 4 of the ACUCOBOL-GT documentation set.

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 4.2.1, "Runtime Configuration Variables"**).

6.6 Encryption

If necessary, you can direct AcuServer to encrypt all data exchanged between AcuServer and a given requester (a client connection).

Note: The use of encryption can have a significant performance cost. The performance cost is determined by the quantity of data being exchanged, the speed and bandwidth of the network, and the computational power of both the client and AcuServer host machines. If you plan to use encryption, prior to deployment you should test and benchmark your application with encryption enabled to ensure that your performance requirements are met.

Encryption is enabled with two configuration variables: **ENCRYPTION_SEED** and **AGS_SOCKET_ENCRYPT**. Once encryption is turned on, it remains on until the program terminates and the connection with AcuServer is closed.

The configuration variable ENCRYPTION_SEED must be set in *both* the AcuServer configuration file and the runtime (client-side) configuration file. The value of ENCRYPTION_SEED is used to initialize the encryption facility. For more information, see **section 4.2.2, "Server Configuration Variables."**

The configuration variable AGS_SOCKET_ENCRYPT must be set to "on" in the runtime configuration file. If it is set within the program, it must be set to "on" before the application opens its first remote file. The default value of AGS_SOCKET_ENCRYPT is "off".

7 Programming for AcuServer

Key Topics:

Programming Considerations 7-2
Accessing Remote Files 7-2
Server Name Management 7-8
Multiple-Record Mode 7-14
Restrictions to Library Functions 7-18

7.1 Programming Considerations

There are three programming issues to consider with AcuServer® file server software:

- how to modify your application to use remote name notation to access remote files
- how to minimize the work required to add or change the name of an AcuServer file server (pertinent to large or rapidly changing networks)
- how and whether to use AcuServer's multiple-record mode (pertinent to those who read a large number of sequential records without user interruption, as when generating reports).

This chapter discusses all three programming issues. In addition, it discusses restrictions to three ACUCOBOL-GT® library routines of which programmers should be aware: C$COPY, C$FILEINFO, and RENAME (details in **section 7.5, "Restrictions to Library Functions"**).

7.2 Accessing Remote Files

Enabling your applications to use AcuServer requires that your applications reference remote files with *remote name notation*. To ensure that all references to remote files include remote name notation, you need to know how your application currently finds data files:

- Are file paths defined in the application code?
- Are file name aliases defined in the runtime configuration file?
- Are the runtime configuration variables **FILE_PREFIX** or CODE_PREFIX used to define search paths?

If your file paths are hard coded in your source code, you'll need to modify them to add remote name notation. A better approach is to remove hard coded paths and to replace them with the FILE_PREFIX configuration variable, or name aliases .

Note: If your application exclusively uses FILE_PREFIX and/or CODE_PREFIX to define data and code file search paths, all you need to do to enable your application to use AcuServer is to add remote paths to the definition of FILE_PREFIX and CODE_PREFIX. See the sections that follow.

7.2.1 Remote Name Notation

Accessing remote files requires that your application refer to remote files with *remote name notation*. The ACUCOBOL-GT runtime looks for remote name notation to identify requests to AcuServer. Remote name notation has the following format on UNIX:

```
@servername:/pathname
```

On Windows NT and Windows 2000 - 2008, the drive designation must be included along with the pathname:

```
@servername:c:\pathname
```

I/O requests to files prepended with "*@server-name:/path-name*" are routed to AcuServer on the host specified by *server-name*. AcuServer looks for the named file in the directory specified by *pathname*.

7.2.2 Using FILE_PREFIX and CODE_PREFIX

The easiest way to enable your applications to use AcuServer is to update the FILE_PREFIX and CODE_PREFIX configuration variables in the runtime and server configuration files. Because these variables can be used to define multiple search directories, they are powerful tools for accessing both local and remote files. Sites that use FILE_PREFIX to specify multiple directories or multiple servers (either multiple server host machines or multiple instances of the **acuserve** program on a single host) may note a significant performance enhancement using name aliasing instead of FILE_PREFIX. See **Section 7.2.4, "Using Name Aliases,"** for more information.

Caution: When using FILE_PREFIX on Windows NT, Windows 2000 - 2008 machines, it is best to use the full path to the directory, rather than using share names (a share name is the name assigned when a directory is set up for sharing using the Properties/Share/Share As dialog). Share names can seriously degrade file access performance because Windows NT and Windows 2000/2003/2008 are very slow to resolve them.

To add a remote search path, you have two options.

Option 1: Include the name and path of the remote directory to the variable's definition in the runtime configuration file (which will also include any local object and data directories). You do not have to make any changes to the server configuration file.

For example:

```
FILE_PREFIX  .  /usr/data  @condor:/usr/data/ap  @condor:/usr/data/ar
CODE_PREFIX  .  /usr/objects  @condor:/usr/objects/ap  @condor:/usr/objects/ar
```

In this example, whenever the application attempts to access a data file, it will first look for the file in the current directory ("."), then in the local "/usr/data" directory, and finally on file server "condor" in the directory "/usr/data/ap" or "/usr/data/ar."

To locate an object file, the application looks first in the current directory ("."), then in the local "/usr/objects" directory, and finally on the file server "condor" in the directory "/usr/objects/ap" or "/usr/objects/ar".

Option 2: Specify only the server name for the PREFIX variables in the runtime configuration file (along with any local object and data directories), then specify the complete server path in the server configuration file.

For example, the runtime configuration file might contain:

```
FILE_PREFIX  .  /usr/data  @condor:
CODE_PREFIX  .  /usr/objects  @condor:
```

The server configuration file would then include something like this:

```
FILE_PREFIX  /usr/data/ap  /usr/data/ar
CODE_PREFIX  /usr/objects/ap  /usr/objects/ar
```

In this situation, a single network request can search multiple server directories for a file. (Setting multiple server directories on the client requires multiple network requests to search the directories.)

Note that you can change the PREFIX variables in the server configuration file after **acuserve** has started using the "-config" utility.

Note that if you want to specify the root directory on the server as a search path in the runtime configuration file, you must include a forward slash ("/") after the server name. To search the root directory on a server called "condor," for example, you would specify "@condor:/".

Although FILE_PREFIX and CODE_PREFIX are usually not applied to file names specified with absolute path names, you can force these variables to be used when the path name begins with a forward slash ("/"). To do this, set the APPLY_FILE_PATH or APPLY_CODE_PATH configuration variables (either locally or on the server) to "on" ("1", "true", "yes"). For example, if your application specifies the file:

```
/usr/bernie/data/ind.dat
```

and FILE_PREFIX is set to:

```
FILE_PREFIX @condor:
```

and APPLY_FILE_PATH is "on", the runtime will attempt to access the file at:

```
@condor:/usr/bernie/data/ind.dat
```

APPLY_FILE_PATH and APPLY_CODE_PATH do not affect file names beginning with a device name (such as "c:").

If the application attempts to access a file in a location (path) that does not exist, the program will terminate, displaying an error similar to the following:

```
File error 35 on SEQ1.DAT
```

(This error also appears if the runtime is not AcuServer-enabled.)

Note that the maximum length of FILE_PREFIX and CODE_PREFIX is restricted by a number of factors. The ACUCOBOL-GT runtimes restrict runtime configuration variables to a maximum of 4096 characters. The host operating system may, in addition, restrict environment variables to less than 4096 characters (refer to your operating system documentation).

7.2.3 Command-Line and Configuration File Remote Name Notation

Most files that can be named on the command line or defined in the runtime configuration file can be specified with remote name notation. There are two general exceptions to this:

1. If you are redirecting standard input or standard output to a file, the file cannot have remote name notation.
2. If you are using an indexed file system other than Vision, those indexed files cannot have remote name notation.

Files that can have remote name notation include the following:

configuration files	(“-c” runtime option)
error files	(“-e” runtime option)
object file libraries	(“-y” runtime option)
object files	(specified on runtime command line or DEFAULT_PROGRAM configuration variable or CODE_PREFIX configuration variable)
hot-key programs	(HOT_KEY configuration variable)
data files	(FILE_PREFIX configuration variable or name aliases in the configuration file or specified in the program)
log files	(LOG_FILE and *filename*_LOG configuration variables)
log directory	(LOG_DIR configuration variable)
xfd directory	(XFD_DIRECTORY configuration variable)

Note that AcuServer may not be used for sort files with the SORT_DIR configuration variable.

Files that cannot have remote name notation include:

keyboard input file	(“-i” runtime option)
keystroke playback file	(“-k” runtime option)
display output file	(“-o” runtime option)
debug input file	(“-r” runtime option)

Below is an example of a runtime command line that uses remote name notation to specify some of its runtime files:

```
runcbl -c @host1:/usr1/config/cblconfig
     -e @host2:/usr2/errors/err.out
     @host3:/usr3/objects/sample.obj
```

The following are example definitions for runtime configuration variables that can have remote name notation.

```
CODE_PREFIX @host3:/usr3/objects
DEFAULT_PROGRAM @host3:/usr3/objects/sample.obj
FILE_PREFIX @host3:/usr3/data
HOT_KEY "@host3:/usr3/objects/sample.obj"=100,200
LOG_DIR @host3:/usr3/log
LOG_FILE @host3:/usr3/log/log.out
XFD_DIRECTORY @host3:/usr3/dictionaries
```

7.2.4 Using Name Aliases

The runtime configuration file offers a second method for accessing remote files. You can use the runtime configuration file to define *file name aliases*. A name alias is a substitute string for the literal name that appears in the ASSIGN TO clause of a SELECT statement.

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

```
input-output section.
file-control.
    select idx-file
    assign to disk "IDXDAT"
    binary sequential
    status is idx-status.
```

you could alias “IDXDAT” in your runtime configuration file by adding:

```
IDXDAT  @condor:/usr/data/idx.dat
```

This line aliases “IDXDAT” to be file “idx.dat” on file server “condor” in directory “/usr/data.”

7.3 Server Name Management

At large installations, with many clients and servers, changes in file location and network configuration can mean frequent tedious maintenance of client runtime configuration files. Every time a new server comes on-line, or a network reorganization moves data files from one server to another, the FILE_PREFIX and CODE_PREFIX definitions and name aliases must be updated.

One way to simplify the network maintenance would be to create a file that contains the name of the file server. Each application would then be written so that it reads this file and uses the name it finds there when referencing the server. If you need to move your files to a new server, you have to change only this single file.

Here's an example of this approach:

First determine a permanent location for the server name file, and create it there. The following program creates a server name file on machine "faithful," on our hypothetical network:

```
identification division.
program-id. svnamcrt.
environment division.
input-output section.
file-control.
    select namefile
    assign to
    disk "@faithful:/usr/acucobol/srvrnam.dat".

data division.
file section.
fd  namefile.
01  namerec    pic x(60).

screen section.
01  enter-screen.
    03  "REMOTE DIRECTORY: " line 5.
    03  using namerec.

procedure division.
main-logic.
```

```
    open output namefile.
    display window erase.
    display enter-screen.
    accept enter-screen.
    write namerec.
    close namefile.

    stop run.
```

For the purposes of this example, we enter "@helios:/usr2/sales/records" at the ACCEPT. This string designates the machine (helios) and the directory on that machine ("/usr2/sales/records") where our data files can be found.

The next step is to create a front-end program that your application will use to read the server name file. It is possible for this program to be written in such a way that you can avoid any recompilation of existing programs within your application. The following is an example of such a front-end program:

```
identification division.
program-id. srvrnam.
environment division.
input-output section.
file-control.
    select namefile
    assign to
    disk "@faithful:/usr/acucobol/srvrnam.dat".

data division.
file section.
fd  namefile.
01  namerec    pic x(60).

working-storage section.

01  ws-prefix  pic x(80).

procedure division.
main-logic.
    open input namefile.
    read namefile.
    close namefile.

    string namerec
        delimited by spaces into ws-prefix.
    set environment "FILE_PREFIX" to ws-prefix.
```

```
    call "your_app".

    stop run.
```

In the above example, the program reads the server name file (found on "faithful"), creates the environment variable "FILE_PREFIX" set to the value found in the server name file ("@helios:/usr2/sales/records"), and then calls the existing application. Because the variable is defined in the same run unit in which the application runs, the variable is available to the application. File operations are now routed to the machine "helios."

A file trace for this program might look like:

```
Configuration file = '/etc/cblconfig'
Try loading 'srvrnam'...
srvrnam loaded
@faithful:/usr/acucobol/srvrnam.dat: open input
@faithful:/usr/acucobol/srvrnam.dat: next
@faithful:/usr/acucobol/srvrnam.dat: close
Set parameter 'FILE_PREFIX' to '@helios:/usr2/sales/records'
Try loading your_app...
your_app loaded
Set keystroke to 'Edit=Next Terminate=13 ^M'
thelp.dat: open i-o
tword.dat: open i-o
users.dat: open i-o
```

Note that FILE_PREFIX is set to the directory on "helios" where the data files are found. The application behaves normally, and the FILE_PREFIX variable works just as if it had been changed in the runtime configuration file, but without the effort.

There are other approaches that the front-end program might take to define FILE_PREFIX. In the following program, FILE_PREFIX includes the value held in the server name file, as well as any values that exist in the configuration file:

```
identification division.
program-id. srvrnam.
environment division.
input-output section.
file-control.
    select namefile
    assign to
    disk "@faithful:/usr/acucobol/srvrnam.dat".
```

```
data division.
file section.
fd  namefile.
01  namerec     pic x(60).

working-storage section.

01  ws-prefix   pic x(80).
01  full-path   pic x(120).

procedure division.
main-logic.

    open input namefile.
    read namefile.
    close namefile.

    accept ws-prefix from environment "FILE_PREFIX".

    move spaces to full-path.
    string namerec, " ", ws-prefix
        delimited by "  " into full-path.
    set environment "FILE_PREFIX" to full-path.

    call "your_app".

    stop run.
```

This program accesses the server name file and calls the application, just as in the first example. However, the construction of the FILE_PREFIX variable includes both the value found in the server name file and the existing definition of FILE_PREFIX, as defined in the configuration file.

In the next example, the front-end program creates a full pathname for each file in the application, using a different variable to hold the location of each file:

```
identification division.
program-id. srvrnam.
environment division.
input-output section.
file-control.
     select namefile
     assign to disk "@faithful:/usr/acucobol/srvrnam.dat".
```

```
data division.
file section.
fd  namefile.
01  namerec      pic x(60).

working-storage section.
01  first-file   pic x(20) value "THELP_FILE".
01  second-file  pic x(20) value "KEYWORDS_FILE".
01  third-file   pic x(20) value "USER_FILE".
01  file-1       pic x(20) value "thelp".
01  file-2       pic x(20) value "tword".
01  file-3       pic x(20) value "users".
01  ws-suffix    pic x(10).
01  full-path    pic x(90).

procedure division.
main-logic.

    open input namefile.
    read namefile.
    close namefile.
    accept ws-suffix from environment "FILE_SUFFIX".

    move spaces to full-path.
    string namerec, "/", file-1, "."
        ws-suffix
        delimited by spaces into full-path.
    set environment first-file to full-path.

    move spaces to full-path.
    string namerec, "/", file-2, "."
        ws-suffix
        delimited by spaces into full-path.
    set environment second-file to full-path.

    move spaces to full-path.
    string namerec, "/", file-3, "."
        ws-suffix
        delimited by spaces into full-path.
    set environment third-file to full-path.

    call "your_app".

    stop run.
```

Notice that the program constructs the full path name of each file. If there is a file suffix, and it is known, it could be hard coded into this program. If there is a file suffix, but it is variable, this program could extract that information from the environment and append this variable suffix to the full path name. However, be aware that this approach *may* require recompilation of the existing programs within the application. In the case of the above example, the ASSIGN clause in each of the affected SELECT statements includes a hard coded file suffix. The period character (".") in that name cannot be used in the name of the environment variable, so the program must be recompiled after the code change:

```
input-output section.
file-control.

    select optional thelp-file
*       assign to "thelp.dat"
        assign to disk ...

    select optional keywords-file
*       assign to "tword.dat"
        assign to disk ...

    select optional user-file
*       assign to "users.dat"
        assign to disk ...
```

A file trace for this example might look like:

```
Configuration file = '/etc/cblconfig'
Try loading 'srvrnam'...
srvrnam loaded
@faithful:/usr/acucobol/srvrnam.dat: open input
@faithful:/usr/acucobol/srvrnam.dat: next
@faithful:/usr/acucobol/srvrnam.dat: close
Assign 'THELP_FILE' to '@helios:/usr2/sales/records/thelp.dat'
Assign 'KEYWORDS_FILE' to '@helios:/usr2/sales/records/tword.dat'
Assign 'USER_FILE' to '@helios:/usr2/sales/records/users.dat'
Try loading 'your_app'...
your_app loaded
THELP-FILE: open i-o
>>>translated name =   @helios:/usr2/sales/records/thelp.dat
KEYWORDS-FILE: open i-o
>>>translated name =   @helios:/usr2/sales/records/tword.dat
USER-FILE: open i-o
>>>translated name =   @helios:/usr2/sales/records/users.dat
```

Note that in the above example, *only* those files specifically named in an environment variable will be looked for or created on the file server.

7.4 Multiple-Record Mode

AcuServer includes a multiple-record mode designed for programs that read a large number of sequential records without user interruption, such as when you are generating a report. Rather than reading one record at a time, AcuServer returns multiple records in a single network request, dramatically improving performance. Files opened in this mode are optimized for sequential reading, but have restrictions placed on the types of operations they can do. (These restrictions are outlined in **section 7.4.1, "File Limitations."**)

To use multiple-record mode, specify the name of the file to be opened in this mode using the runtime configuration variable, *filename*_MRC (Multiple-Record Count). You can then specify the number of records to be sent in each packet in the *filename*_MRC variable on the client, or you can set this variable to "1" and specify the number of records to send using the server configuration variable MULTIPLE_RECORD_COUNT. Both variables are described in Chapter 4.

Only files open in modes that do not require record locking can use multiple-record mode. Files that are opened in EXCLUSIVE I/O mode or open INPUT are good candidates. Note that this only affects files that read records, so files opened OUTPUT or EXTEND cannot use this mode.

In multiple-record mode, the client caches records that are read from the server. The first time a client does a READ NEXT, the server actually sends multiple records to the client. Until that collection of records has been processed, there is no more communication between the client and the server for future READ NEXT operations.

There are some exceptions to these general rules. When the client executes a random READ, the server sends only a single record. The next time the client executes a READ NEXT, the server sends a full set of records.

7.4.1 File Limitations

All of the standard rules of COBOL apply in multiple-record mode, with the following limitations:

- Only files that require no record locking can be opened in multiple-record mode. This limitation is enforced silently on both the client and the server. If a user tries to open a file that requires record locking in this mode, the file is still opened, but not in multiple-record mode. In other words, multiple-record mode does not affect the lock mode specified by the COBOL program, but the lock mode may affect whether records are sent in single or multiple-record packets.

- Files open in this mode may not be read backward. (that is, READ-NEXT commands are supported; READ-PREVIOUS commands are not.) The purpose of this mode is to improve sequential performance. If you attempt to read a file backward that has been opened in multiple-record mode, an error 9B results (requested operation is not supported).

7.4.2 Other Considerations

There are several things of which you should be aware when using multiple-record mode:

1. In multiple-record mode, the client requires more memory than if it has no files open in this mode. In particular, the client requires memory for all the records in the cache from the server. For example, if three files are open in multiple-record mode with maximum record sizes of 100, 250, and 800, and MULTIPLE_RECORD_COUNT is set to “10”, the client requires 100 * 10 + 250 * 10 + 800 * 10 bytes of storage for holding all the records from the server. This is compared to needing only 100 + 250 + 800 bytes of storage if these files were not open in multiple-record mode.

2. The server also requires more memory when you use files in this mode. This should not be a significant addition to the usual amount of memory that the server needs, because the server reads all the records it sends and then does not save these records. The server needs only a

single large internal buffer to hold the records temporarily. This buffer should have a size of the largest MAXIMUM_RECORD_SIZE multiplied by the largest MULTIPLE_RECORD_COUNT.

3. This mode is inherently subject to race conditions, including those described below. (Note that these are not the only race conditions that you may encounter.) If your application is written in such a way that it will experience race conditions, you should not use multiple-record mode.

   Condition 1: If two users are accessing a single file, and one of those users has the file open in multiple-record mode, the following situation could happen:

   - User A opens and reads the first record. At this time the server sends multiple records to user A.
   - User B opens the file and writes a record in a location within the record set that the server has already sent to user A.
   - User A never sees that record unless the file is closed and reopened.

   In other words, when the server sends a collection of records to a client, that client now has a snapshot of the records on the server. The client does not query the server about any new records, since this would remove the performance gain of using multiple-record mode.

   Condition 2: Suppose a user opens the file I/O for exclusive access (or allowing only readers) in multiple-record mode. Suppose this user reads the first record, and then writes a record that is after the first record, but before the last record in the client cache.

   As the user executes READ NEXT, that record never appears. For example, suppose the file has a key that is a PIC 99 data item, and that ten records are sent in each batch. Also suppose that the file has records with keys 05, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95. The user opens the file and executes a READ NEXT. At this time, the first ten records are in the client cache (records 05, 10, 15, 20, 25, 30, 35, 40, 45, 50) and the COBOL program has record 05 in its record area. Suppose the user then executes a WRITE to write a record with the key of 08 and then executes another READ NEXT. The record that the COBOL program has in its record area is now record 10, not

record 08. Again, the client and server will not communicate about this situation, since it would completely remove the performance gain of using multiple-record mode.

Condition 3: Now suppose you have the same file as in the previous case, and that the user has executed the first READ NEXT. Suppose the user then executes a REWRITE on record 10 to change data in that record and then executes a READ NEXT. The COBOL program will now have the original data from the file for record 10, not the data that the COBOL program just rewrote.

Since multiple-record mode is inherently subject to race conditions, you should use this mode only in situations where you are sure you understand all of the consequences.

7.5 Restrictions to Library Functions

There are four restrictions on the usage of the library functions **C$COPY** and **RENAME** when they are used with remote name notation:

- RENAME does not support file copies or renames from client systems to file servers, or vice versa. In other words, renames are not supported across machines.
- For a rename, remote name notation is not allowed on the destination file parameter.
- When copies or renames are executed on a file server, the location of the copied or renamed file is determined by the relative or absolute pathnames of the parameter values.
- FILE_PREFIX and name aliases are not used to find files named in the function parameters.

These restrictions are explained in detail in the following sections.

7.5.1 C$COPY

C$COPY creates an exact duplicate of *SOURCE-FILE* in *DEST-FILE*. The usage of C$COPY is:

```
CALL "C$COPY"
    USING SOURCE-FILE, DEST-FILE, FILE-TYPE
    GIVING COPY-STATUS
```

Remote name notation is permitted on both the *SOURCE-FILE* and the *DEST-FILE* parameter.

If either the *SOURCE-FILE* or *DEST-FILE* parameter specifies a remote file:

- The copy takes place on the specified machine.
- The location of the source and destination files on the server is dependent on the paths specified by *SOURCE-FILE* and *DEST-FILE*. Parameters that do not specify an absolute pathname (an absolute path starts with a "/"), are considered to be located relative to the directory in which the server (**acuserve**) was started. For example:

  If **acuserve** was started on condor in the /acucobol directory, and *SOURCE-FILE* is:

  ```
  @condor:index.dat
  ```

  AcuServer expects to find index.dat in directory /acucobol (note the absence of a "/" after "condor").

  If *SOURCE-FILE* is:

  ```
  @condor:/usr2/sales_records/index.dat
  ```

  AcuServer expects to find index.dat in the absolute path /usr2/sales_records.

  These pathing rules hold for both *SOURCE-FILE* and *DEST-FILE*.

FILE-PREFIX is not used to search for file names held in any of the function's parameters. *Name aliases* are not substituted for names held in the function's parameters. Full path names to the files, including remote name notation on the *SOURCE-FILE* and *DEST-FILE*, must be hard coded in the

function call. A more portable solution is to use the ACCEPT FROM ENVIRONMENT statement to get pathnames during program execution. See the entry for ACCEPT in Chapter 6 of the *ACUCOBOL-GT Reference Manual*.

C$COPY returns a zero (“0”) if successful, a one (“1”) if the file does not exist or is not a regular disk file, or a two (“2”) if there is a network error (for example, AcuServer is not running). This value is returned to *COPY-STATUS*.

For a complete description of C$COPY, excluding the above restrictions, see Appendix I of the ACUCOBOL-GT documentation.

7.5.2 C$FILEINFO

C$FILEINFO retrieves operating system information for the named file. The syntax of C$FILEINFO is:

```
CALL "C$FILEINFO"
   USING FILE-NAME, FILE-INFO,
   GIVING STATUS-CODE
```

There are no restrictions on the use of C$FILEINFO. Note, however, that *STATUS-CODE* holds a zero (“0”) if successful, a one (“1”) if the file does not exist or is not a regular disk file, and a two (“2”) if there is a network error (for example, AcuServer is not running).

FILE_PREFIX is not used to search for *FILE-NAME*. A full path name to the file, including remote name notation, must be hard coded in the function call. A more portable solution is to use the ACCEPT FROM ENVIRONMENT statement to get pathnames during program execution. See the entry for ACCEPT in Chapter 6 of the *ACUCOBOL-GT Reference Manual*.

7.5.3 RENAME

RENAME renames the file specified by *SOURCE-FILE* to the name specified by DEST-FILE. The syntax of RENAME is:

```
CALL "RENAME" USING SOURCE-FILE, DEST-FILE,
        RENAME-STATUS, FILE-TYPE.
```

Remote file name notation is not permitted on the *DEST-FILE* parameter. If the *SOURCE-FILE* parameter specifies a remote file:

- The rename takes place on the specified server.
- The location of source and destination files on the server is dependent on the paths specified by *SOURCE-FILE* and *DEST-FILE*. Parameters that do not specify an absolute pathname (an absolute path starts with a "/"), are considered to be located relative to the directory in which the server (**acuserve**) was started. For example:

  If **acuserve** was started on condor in the /acucobol directory, and *SOURCE-FILE* is:

  ```
  @condor:sales_records/index.dat
  ```

  AcuServer expects to find index.dat in directory /acucobol/sales_records (note the absence of a "/" after "condor:").

  If *SOURCE-FILE* is:

  ```
  @condor:/usr2/sales_records/index.dat
  ```

  AcuServer expects to find index.dat in directory /usr2/sales_records.

  These pathing rules hold for both *SOURCE-FILE* and *DEST-FILE*; however, *DEST-FILE* can never include the name of the remote host (@condor).

FILE_PREFIX is not used to search for file names held in any of the function's parameters. *Name aliases* are not substituted for names held in the function's parameters. Full path names to the files, including remote name notation on the *SOURCE-FILE*, must be hard coded in the function call. A more portable solution is to use the ACCEPT FROM ENVIRONMENT statement to get pathnames during program execution. See the entry for ACCEPT in Chapter 6 of the *ACUCOBOL-GT Reference Manual*.

RENAME-STATUS holds a zero ("0") if the routine is successful; otherwise, it is set to the operating system's error number, or "19" if there is a network error (for example, AcuServer is not running).

For a complete description of RENAME, excluding the above restrictions, see Appendix I of the ACUCOBOL-GT documentation.

8 System Management

Key Topics:

Machine Failures 8-2
Error and Status Codes 8-4
Diagnosing Errors with C$PING 8-8
Troubleshooting 8-10
Frequently Asked Questions 8-25

8.1 Machine Failures

There are two major concerns regarding client and server machine failures:

- What happens to open files when a client or server crashes?
- How are other clients and servers affected by a crash?

When a client application is terminated with a Control + C or **kill** command (other than a "kill -9"), the AcuServer® software detects the termination and closes all files held open for that client process. However, other terminal software and hardware failures may not be detected.

AcuServer offers an optional mechanism for detecting connections that terminate unexpectedly. The mechanism works by establishing regular, periodic communication from the UNIX/Linux or Windows client to AcuServer. The **acuserve** process keeps a record of each participating connection's periodic communication and regularly analyzes these records for evidence of a dead connection. If a connection fails to send its periodic communication for two consecutive periods, **acuserve** concludes that the connection is dead, closes any open files associated with the connection, and disconnects the socket. The detection mechanism is enabled with two configuration variables, one on the server and one on the client.

On the server, the detection mechanism is enabled with the DEAD_CLIENT_TIMEOUT configuration variable. When DEAD_CLIENT_TIMEOUT is set to "-1", its default value, the detection mechanism is disabled. When DEAD_CLIENT_TIMEOUT is set to any other value, the detection mechanism is enabled and the variable's value is interpreted as the interval, in seconds, at which **acuserve** will analyze its records to detect dead connections.

On the client, the AGS_PING-TIME configuration variable enables or disables the detection mechanism. When AGS_PING_TIME is set to "-1", the periodic communications mechanism is disabled and the connection cannot be automatically disconnected. When AGS_PING_TIME is set to any other value, the reporting mechanism is enabled and the connection is included in **acuserve**'s monitoring table. The value of AGS_PING_TIME specifies the interval, in seconds, at which the client will send an "I'm alive" message (a "no-op" instruction) to the server. The value is communicated to

acuserve at the time that the connection is established so that **acuserve** can determine, when it checks its monitoring records, whether a connection has missed two or more consecutive communication periods.

The following example illustrates the system's behavior. If DEAD_CLIENT_TIMEOUT is set to "300" (300 seconds = 5 minutes), the dead connection detection mechanism is enabled on the server and **acuserve** analyzes its communication records every five minutes to look for dead connections. If, on the client, AGS_PING_TIME is set to "60", when a connection is established with the server, the server is told to expect an "I'm alive" message every 60 seconds and the connection proceeds to send those messages. If for some reason the connection fails to send the message for two or more consecutive periods, the next time that **acuserve** checks its records (which it's checking every five minutes), it will detect that the connection is dead and automatically close all files opened by that connection, release all locks held by that connection, and close the associated socket.

Note: The AGS_PING_TIME mechanism depends on regular system calls, which may slow some systems (notably some versions of Solaris) and affect the runtime's performance. Please test for performance before using this configuration variable in a production environment.

Also note that AGS_PING_TIME starts an internal runtime thread to send its periodic communication to the server. This means that any activity that suspends thread switching will prevent this mechanism from working.

With clients that do not participate in the dead client detection mechanism, if a client system crashes while using AcuServer, the server will hold the client's open files open until the unlock function is used to close the files, or until the **acuserve** daemon is stopped and restarted (when AcuServer is stopped, all open files are closed). For a description of the use of the unlock function, see **section 5.10, "Closing Stranded Files."**

Should the server go down, all clients actively using AcuServer will get access errors when attempting to communicate with the server. Client applications must disconnect (see C$DISCONNECT, Appendix I, Book 4 of the ACUCOBOL-GT® documentation set) or shut down and wait for the return of the server. All files that were open on the server at the time of the crash are left in an unknown state and may be corrupt.

Note: If **acuserve** is automatically started when the server boots, **acuserve** should be immediately halted. Before AcuServer is started, all files that might have been affected by the crash should be checked and, if necessary, rebuilt. After all files have been verified, the **acuserve** daemon can be started.

8.2 Error and Status Codes

AcuServer reports errors and status information in many ways, including:

- **Error Logging**
- **Event Logging**
- **Start Status Codes**
- **File Access Failures, "9D" Errors**

These are described in this section.

8.2.1 Error Logging

If you want AcuServer to output error messages to an error log, specify the "-e" option when starting AcuServer with "**acuserve** -start". On Windows, you can also specify the error log file using the AcuServer Control Panel (ACP).

If "-e" is not specified in UNIX, **acuserve** will attempt to direct error output to /dev/console.

In Windows, **acuserve** will attempt to direct error output to a file named "acuserve.err" in the Windows NT/2000 system directory (typically c:\winnt\system32). If these directories cannot be opened, **acuserve** will attempt to append to a file named "acuserve.err" in the current directory. If

that file doesn't exist, or the file append fails, **acuserve** will print the message "acuserve: can't open error output file" to standard output, and **acuserve** will terminate.

For more information on the "-e" option, refer to **section 5.6, "Starting and Stopping acuserve."**

8.2.2 Event Logging

acuserve writes information to the Windows event log or UNIX syslog when something of significance to a system administrator happens, such as **acuserve** starting and stopping, **acuserve** halting because of an unrecoverable error, or other emergency-type events. If a UNIX system does not support syslog, AcuServer writes these messages to /dev/console instead. This information is also output to the error log.

On Windows, the event messages to go to any computer in the network. Use the **WINNT-EVENTLOG-DOMAIN** variable to set the UNC name of the computer(s) to which you want to send event log messages. The machine that you specify must give **acuserve** access privileges or the event log output will fail.

8.2.3 Start Status Codes

Whenever you try to start AcuServer, either by the "**acuserve** -start" command or the ACP, AcuServer returns a status to the operating system indicating success or failure. The way to get this status depends on your shell. When using the Bourne shell (or a compatible shell), echo $? to see the return status of any program executed from the shell.

The return codes are as follows:

Return Code	Description
0	Success. AcuServer is running.
1	Invalid argument.
2	Missing license.

Return Code	Description
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, but the socket is used by something other than AcuServer.
10	AcuServer is already running on this port.

8.2.4 File Access Failures, "9D" Errors

AcuServer returns an error code whenever an AcuServer file access fails.

An error code of "9B" indicates that the requested operation is not supported. Other AcuServer error codes have the following format:

9D,*xxx*

"9D" indicates a server error and is returned in your *file-status* variable.

You can retrieve the extended code with the "-x" runtime switch, or by calling C$RERR. Note that you can pass two parameters to C$RERR to fetch interface errors. The first parameter retrieves the two- or three-digit extended error code. This parameter must be at least PICTURE X(5). The second parameter retrieves a message associated with the error condition and should be at least PICTURE X(80).

AcuServer errors

AcuServer errors have a three-digit extended code. These errors include the following:

Error Code	Description
9D,100	Invalid syntax for FILE_PREFIX. The correct syntax for FILE_PREFIX is: FILE_PREFIX *@server-name:pathname*
9D,101	The version of the server is not compatible.
9D,102	Invalid connection password specified by client.
9D,103	Connection to server refused, access denied.
9D,104	PC/TCP resident kernel is not loaded, or the communication stream between the client and the server was interrupted, or a file open failure occurred.
9D,105	User count exceeded.
9D,106	Attached to a server that does not appear to be AcuServer.
9D,109	Unknown command sent between AcuServer runtime and AcuConnect server or between AcuConnect runtime and AcuServer.

An error 9D,101 indicates that the version of the ACUCOBOL-GT runtime and the version of AcuServer are incompatible. AcuServer Version 8.0 and later will not work with earlier versions of the ACUCOBOL-GT runtime. Conversely, earlier versions of AcuServer will not work with a Version 8.0 or later runtime.

An error 9D,105 indicates that the number of users attempting to use AcuServer exceeds the number of users authorized by your license. For connections with the ACUCOBOL-GT Web Runtime, it means that the number of Web users trying to access AcuServer exceeds the number of ACUCOBOL-GT runtime licenses on the server machine. If your COBOL program receives a file status code 9D,105 in connection with plug-in users, it should display a message box stating that the Web file server is busy, and to please retry later. To extend the number of authorized users, please call your Micro Focus ***extend*** Sales Professional.

Client runtime errors

Whenever a client runtime passes a file handle to AcuServer with a request for file access, AcuServer validates the file handle before performing any I/O using that handle. If AcuServer determines that a file handle is not valid (for example, if the file pointer is invalid or the file is not open), it returns an error to the client runtime.

Client runtime error codes have the following format:

30,*xx*

"30" indicates a system error. Client runtime errors include the following:

Error Code	Description
30,02	No such file or directory
30,09	Bad file number

8.3 Diagnosing Errors with C$PING

ACUCOBOL-GT includes a library routine, C$PING, that provides diagnostic information for debugging network performance with AcuServer. If you write a small COBOL program that calls C$PING, you will obtain information such as whether or not the client is AcuServer-enabled, whether or not the parameter passed from the COBOL program is valid, whether the connection was refused or accepted by the server, and whether or not a socket error occurred.

A compiled COBOL program called **acuping** is included in the \AcuGT\bin directory in which you installed ACUCOBOL-GT. This program contains a call to the C$PING library routine, along with a graphical screen that runs well on Windows and UNIX platforms. If desired, you can run this program to diagnose AcuServer errors, or you can modify the source code for the program as required. You'll notice that AcuPing has four entry fields: log file name, server to ping, number of pings, and ping delay (1/10th second). The program shows results for Message ID, Time at Client, Time at Server, and Round Trip Time.

If you choose to write your own COBOL program, use the following calling sequence for C$PING:

```
call "C$PING" using server-to-ping, server-time [, client-data].
```

where:

server-to-ping is a PIC X(64) data item that should be filled in by the COBOL program before calling C$PING. This is the server that will be pinged.

server-time is a pic 9(8) data item that is filled in by C$PING and that designates the time the server got and returned the request. Note that if this is a group item, the time is left-justified instead of right-justified (as is done via a COBOL MOVE statement). For this reason, a PIC 9(8) elementary data item is highly recommended.

client-data is a PIC X(n) data item that is passed verbatim to the server. The server displays this data in the trace file (if tracing is enabled) and returns it verbatim to the client. The result is that the data in this data item is unchanged, even though it came from the server. This argument is optional.

When the library routine finishes, it sets the external variable *return-code* to the status of the ping, as follows (these values are defined in acucobol.def.):

```
78  CPING-OK                    VALUE 0.
78  CPING-NO-CLIENT             VALUE 1.
78  CPING-PARAM-ERROR           VALUE 2.
78  CPING-CONN-REFUSED          VALUE 3.
78  CPING-VERSION-ERROR         VALUE 4.
78  CPING-SOCKET-ERROR          VALUE 5.
```

CPING-OK	everything worked, and *time-at-server* has a valid time from the server
CPING-NO-CLIENT	this runtime is not AcuServer-enabled
CPING-PARAM-ERROR	the COBOL program passed an invalid parameter
CPING-CONN-REFUSED	the server refused the connection, possibly because it is not running or is running on a different port than the one for which the client is configured

CPING-VERSION-ERROR	the version of the server is not compatible with this version of the runtime
CPING-SOCKET-ERROR	some unknown socket error occurred

If the server has tracing enabled, the ping request is logged in the trace file.

8.4 Troubleshooting

This section is a collection of step-by-step diagnostic procedures for finding and resolving common system problems. Many of these procedures require access to *root* or *Administrator* privileges and are intended for use by your site's AcuServer or system administrator.

If you would like assistance from our Technical Services group, please contact us via phone, fax, or e-mail, using the contact information listed in **Chapter 1, section 1.3, "Technical Services"**.

8.4.1 Ambiguous File Error

Problem: Your application gets an error and aborts, returning an ambiguous AcuServer error message (9D). For example:

```
FILE ERROR #9D ON SEQ1.DAT
MYAPP ABORTED
```

Diagnostics:

Run your application again with the "-x" option:

```
runcbl -x your-usual-options program-name
```

This time when your application gets the error, the runtime will report the extended error code and associated error text. For example:

```
File system error value = 102
Invalid connection password specified by client
```

If your runtime options include the "-e" switch to redirect error output to a log file, look for the extended error message in your log file.

8.4.2 Connection Times Out

Problem: An attempt to connect to AcuServer times out, or your application gets a time-out error during execution.

A typical time-out error message looks like:

```
File error 9D,14 on @condor:/usr2/bsmith/fio_seq.dat
condor: Timed out
```

Time-out errors occur when the server is down, the server crashes while your program is running, or network conditions produce very protracted response times (the result of an overburdened network, or faulty network hardware/ software).

Diagnostics:

1. Confirm that the server is up and responding to the network:

   From your client system use “ping” to query the server:

   ```
   ping hostname
   ```

   Ping will return:

   ```
   hostname is alive
   ```

   or

   ```
   no answer from hostname
   ```

   If there is no answer from the server, determine the cause of the failure (machine down; network connection faulty) and restore the server. With “ping”, confirm that the network connection is working.

2. Confirm that **acuserve** is running on the server:

   - If you are using the Windows graphical environment, check the Services tab in the AcuServer Control Panel. Services that are running are marked with a green bullet and services that are stopped are marked with a red bullet.

     If the service is not running, click **Start**.

- If you are in a command-line environment, enter the command:

  ```
  acuserve -info
  ```

  If **acuserve** is running, the command will return a summary of AcuServer's current status. For example:

  ```
  0 file(s) in use by acuserve on: hostname
  ```

  If **acuserve** is not running the command will return:

  ```
  acuserve is not running on hostname
  ```

  If **acuserve** is not running, start **acuserve** and repeat the "**acuserve** -info" test.

Once **acuserve** is running, rerun your application on the client. If the connection still times out, proceed to step three.

3. Set the runtime configuration variable DEFAULT_TIME_OUT to "100" seconds (four times the default value) and run your application.

   If the connection succeeds, analyze your network for the cause of the prolonged connection time. You should investigate network capacity, collision and retransmission rates, and network hardware or software malfunction.

   If necessary, repair your network and confirm its proper operation. Reset DEFAULT_TIME_OUT to its default setting of "25" seconds and rerun your application.

4. If the application times out, you will probably need to work with your network system administrator, network hardware and software vendors, and our Technical Services to resolve the problem.

8.4.3 File Access Denied

Problem: Attempts to access a file on the server fail with an "access denied" error. A typical "access denied" error looks like:

```
File error 37,07 on @condor:/usr2/bsmith/fio_seq
```

The 37, 07 error code (ANSI 85 standard error code) indicates that the user does not have sufficient permissions to access the file. If your application uses one of the alternate sets of file status codes, your error code will differ. Refer to your *ACUCOBOL-GT Runtime Manual*.

"Access denied" errors occur because a file's permissions, or the permissions of a directory in which the file resides, prohibit access. Finding and correcting "access denied" errors requires familiarity with AcuServer access security and Windows NT/2000 or UNIX file security. You may need access to *Administrator* or *root* privileges. We recommend that you work with your site's AcuServer system administrator to resolve the problem.

AcuServer adds complexity to finding "access denied" errors, because AcuServer handles requests with a *user ID* (user name) designated in the server access file (AcuAccess). The user name designated may not be the user name that you expect (see **section 8.4.4, "Unexpected User Name"**). Therefore, finding and resolving "access denied" errors can involve inspecting two components of the system: file protections and the AcuServer server access file. AcuServer also adds complexity when implementing native system security, because the server impersonates the connecting client.

File permissions diagnostics are used to verify that the ownerships and permissions on the file and directories do, in fact, allow access to the user ID that AcuServer is using (or the user name that you *assume* that AcuServer is using).

The AcuServer user name diagnostics are used to determine the actual user name that AcuServer is using when the access denied error occurs. Step-by-step diagnostic procedures follow.

Whether you first verify the user ID, or first check the file permissions, is up to you. If you're familiar with file security, you may find it faster to check the file permissions first, under the assumption that AcuServer is using the user ID that you expect. Then, if necessary, you can verify the user ID, followed by repeating the file permissions inspection if the actual user name differs from the assumed name.

The steps below break the diagnostic process into three activities:

1. **getting the full path of the file generating the error**

2. checking the file permissions **(UNIX, Windows)**

3. **determining the user ID used by AcuServer**

To review the structure and function of the server access file, see **section 6.2, "The Server Access File."** To review the procedure used by AcuServer to get the requester's user ID, see **section 6.5, "AcuServer Connection Logic."** For a brief description of file permissions, see **section 2.4.1**.

Diagnostics:

1. Getting the file name

Get the name and path of the file that produces the access denied error.

Most applications will report the full path and name of the file in the standard error output. If your application doesn't, run your application again with the "-x" runtime option:

```
runcbl -x your-usual-options program-name
```

If the "-x" option doesn't produce the full name and path, run the application in the debugger with FILE TRACE enabled and examine the log file for the first occurrence of an access error. The file name that precedes the error is the file generating the error.

To run the debugger enter:

```
runcbl -dle error-file your-usual-options program-name
```

-d turns on the debugger. *-l* causes the contents of the runtime configuration file to be included in the error output. *-e* causes file trace and error output to be placed in error-file.

After starting your program, you will be at the debugger screen. Turn on file tracing by entering

```
tf
```

"FILE TRACE" will be echoed to the screen. To start program execution, enter:

```
g
```

Allow the program to run until you encounter the error condition, and then exit. Your log file will contain the error information, all COBOL configuration file variables that have been set, and a record of every OPEN, READ, and WRITE.

Examine the log file and find the error output for the access denied error. The file name that precedes the error is the file generating the error. Note the full path of the file and retain the log file for future reference.

2a. Checking UNIX permissions

If you are using a UNIX server, verify that the file and directory permissions grant access to the user name:

1. Log on to the server with the user name used (assumed to be used) by AcuServer. If that's not possible (you don't know the password or the login attempt fails), skip to step 3.

2. After logging onto the server, perform an "ls -l" on the file using the full path (beginning with "/"). For example:

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

   If the "ls -l" fails with a "cannot access directory" (or similar) error, the permissions on one or more of the directories are too restrictive. To access a directory, the directory must allow you EXECUTE permissions. Adjust the permissions or move the file to a directory that will allow access. If you are not the owner of the directories, you will need to ask the owner, or someone with root privileges, for assistance.

   After adjusting the permissions, verify directory access to the file by performing an "ls -l" on the full path to the file. If the "ls -l" succeeds, the permissions set on the file are displayed. For example:

   ```
   -rw-r--r--  1 prod  general  4870  Aug 18 1993
      /usr2/bsmith/fio_seq
   ```

   Evaluate the file's ownerships and permissions, including the *group* permissions, to determine if they allow the type of access required by the requesting application. Except as noted below, all files opened by AcuServer must allow READ *and* WRITE access. Under Sun OS with ACUCOBOL-GT version 2.1.2 or later, files having READ only permissions may be opened "INPUT".

If the file's permissions need changing and the file is not owned by you, you will need to enlist the assistance of the owner of the file or someone with *root* privileges.

Once the file and directories permit access, run the application again. If the program fails when attempting to access the same file *and* you have not yet confirmed the user name that AcuServer is using, perform the *Confirming the user name* diagnostics (below).

3. If you can't log onto the server under the user name used by AcuServer, ask your UNIX system administrator to evaluate whether the file in question is accessible to the user name. If the file or directories do not allow access, ask the owner of the file, or the system administrator, to adjust the permissions to allow access.

   After the permissions have been adjusted, run the application again. If the program fails attempting to access the same file *and* you have not yet confirmed the user name that AcuServer is using, proceed to the next section. Otherwise, contact our Technical Services.

2b. Checking Windows NT/Windows 2000/ 2003/2008 permissions

If you are using a Windows NT, Windows 2000 - 2008 server, verify that the file and directory permissions grant access to the user name.

1. Log on to the server with the user name used (assumed to be used) by AcuServer. If that's not possible (you don't know the password or the login attempt fails), skip to step 3.

2. After logging onto the server, select the file or directory. Then select:

   File/Properties/Security/Permissions

   from the menus. If the permissions are too restrictive, adjust the permissions or move the file to a directory that will allow access.

   When AcuServer is running as a Windows service, it belongs to an implicit group called "SYSTEM." Otherwise it should belong to the "Administrators" group. If you are using AcuServer system security,

make sure that the “SYSTEM” and “Administrators” groups are added to your file permissions with “Full Control.” This step is not necessary if you are implementing Windows NT security.

Under AcuServer system security, files created by AcuServer are owned by the Administrators group and allow “Full Control” for “SYSTEM” and “Administrator.” “Everyone” is given the permissions specified by the third digit in the *umask* in the AcuAccess file.

If you are not the owner of the directory or file, you will need to ask the owner, or someone with *Administrator* privileges, for assistance.

Once the file and directories permit access, run the application again. If the program fails when attempting to access the same file *and* you have not yet confirmed the user name that AcuServer is using, perform the *Confirming the user name* diagnostics (below).

3. If you can’t log onto the server under the user name used by AcuServer, ask your system administrator to evaluate whether the file in question is accessible to the user name. If the file or directories do not allow access, ask the owner of the file, or the system administrator, to adjust the permissions to allow access.

   After the permissions have been adjusted, run the application again. If the program fails attempting to access the same file *and* you have not yet confirmed the user name that AcuServer is using, proceed to the next section. Otherwise contact our Technical Services.

3. Confirming the user name

Confirm the user name that AcuServer is using when the access denied error occurs (see also, **section 8.4.4, “Unexpected User Name”**).

Note that some of the procedures in this section require *root* or *Administrator* privileges. If you don’t have these privileges, enlist the assistance of your AcuServer or system administrator.

1. If there is any doubt about your *client user name*, confirm your client user name.

   On UNIX systems:

Enter “who am i” at the UNIX prompt.

On Windows systems:

If a user is logged into a Windows NT/2000 workgroup or domain, AcuServer will use the user’s log-in name. Otherwise, it uses the user name value that is set by the environment variable USERNAME. If USERNAME is not set, the runtime uses the value that is set by the environment variable USER. (The values of these variables are case-sensitive.) If neither of these environment variables is set, the runtime uses the literal string “USER”.

To confirm the presence (or absence) of the “USERNAME” environment variable, enter “SET” at the command prompt. SET will display the values of all environment variables.

```
COMSPEC=C:\COMMAND.COM
PATH=C:\;C:\WINDOWS;C:\MOUSE;
PROMPT=$P$G
USERNAME=BERNIE
```

If USERNAME is defined, the value of USERNAME is the name passed to AcuServer. Otherwise, the variable USER is checked. If USER is not set, then the literal string “USER” is passed.

2. If **acuserve** is running with level 2, 3, 6, or 7 tracing enabled (see the “-t” option to “**acuserve** -start”, **section 5.6, “Starting and Stopping acuserve”**), all connections and disconnections are reported in the AcuServer log file. Inspect the log file and look for the last connection entry for your client machine name/client user name combination. The trace messages associated with that connection request will include the *Local Username* that AcuServer has assigned to that connection. Find the line that reports the “mapping” of the client user name to the local user name.

```
acuserve: 06/13/94 12:43:44 -
   Server version: 010000
   Client version: 010000
acuserve: 06/13/94 12:43:44 -
   AUTHORIZATION request from user:bernie
>> client:starling,
   PASSWORD-ATTEMPTS allowed:3
>> Mapping user:bernie to local user:bsmith
>> Socket:5, uid:205, gid:101
```

```
>> Password is not required
acuserve: adding
   client:starling, user:bernie,
   pid:18397, fid:57a28
```

If the local user name is not the one you expect, consult with your AcuServer system administrator to verify that it is the correct user name. Your system administrator can make any necessary changes to the server access file.

3. If **acuserve** is *not* running with tracing enabled, you can turn it on using the **acuserve** "-config" option. At the command line, type "**acuserve** -config", then type "set FILE_TRACE 2" (or 3, 6, or 7) at the prompt, then type "exit". After tracing is turned on, run your application again and follow the procedure outlined in step 2, above.

   When you no longer want tracing enabled, type "**acuserve** -config" again, then type "set FILE_TRACE 0" at the prompt.

8.4.4 Unexpected User Name

Problem: AcuServer 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 isn't matching 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 AcuServer server access configuration, and have access to *root* or *Administrator* privileges. We recommend that you work with your AcuServer system administrator.

The diagnostic procedures include:

1. **confirming your client user name**
2. **confirming your client machine name**
3. looking up your **client user name/client machine name combination** in the server access file

Diagnostics:

1. Confirm your client user name.

   On UNIX clients:

   a. Log onto the client system using the same user name and UNIX environment that resulted in the unexpected user name.

   b. If the system is a UNIX system, enter “who am i” at the UNIX prompt.

      Is the user name returned the name you expected?

   On Windows clients:

   If a user is logged into a Windows NT/2000 workgroup or domain, AcuServer will use the user’s log-in name. Otherwise, it uses the user name value that is set by the environment variable USERNAME. If USERNAME is not set, the runtime uses the value that is set by the environment variable USER. (The values of these variables are case-sensitive.) If neither of these environment variables is set, the runtime uses the literal string “USER”.

   To confirm the presence (or absence) of the “USERNAME” environment variable, enter “SET” at the command prompt. SET will display the value of all environment variables.

```
COMSPEC=C:\COMMAND.COM
PATH=C:\;C:\WINDOWS;C:\MOUSE;
PROMPT=$P$G
USERNAME=BERNIE
```

If USERNAME is defined, the value of USERNAME is the name passed to AcuServer. Otherwise, the variable USER is checked. (Both of these variables are case-sensitive.) If USER is not set, the literal string "USER" is passed.

2. Confirm the name of the client system.

   On UNIX clients, enter:

   ```
   hostname
   ```

   or

   ```
   uname -n
   ```

   The system will return its official network host name.

   For Windows NT/2000 clients:

   a. Go to Start/Settings/Control Panel/Network/Protocols

   b. Select TCP/IP

   c. Choose Properties/DNS/Host Name

   The name you specify for the "Host Name" entry is the one that the runtime uses.

3. On the server, examine the server access file for the record that matches the client machine name/client user name combination. This process requires *root* privileges on a UNIX server, and *Administrator* group privileges on a Windows NT, Windows 2000 - 2008 server. It should be done by the AcuServer system administrator.

   a. Run the server access file manager utility:

      ```
      acuserve -access
      ```

   b. Be sure to enter the name of the *working* server access file when prompted.

   c. Select menu item [4] - "*Display one/all security records*."

   d. Respond "N" to the prompt: "*Display all records?"*

e. To the next two prompts, provide the client machine name and client user name, respectively. The matching record will be displayed.

The AcuServer 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.

8.4.5 Connection Refused

Problem: Attempts to connect to the server fail with a 9D, 103, connection refused error.

Connection refused errors occur for two reasons:

1. The matching access record specifies an invalid *Local Username*, or DEFAULT_USER holds an invalid user name. Note that use of the "same as client" string in the *Local Username* field of the access record can lead to the attempted use of an invalid user name ("same as client" directs AcuServer 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.

Diagnostics:

Confirm your client user name and client machine name, and find the combination's matching entry in the server access file.

1. Follow diagnostic steps 1 and 2 from **section 8.4.4, "Unexpected User Name."**
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 AcuServer system administrator).
   a. On a UNIX server, become *superuser* or log on as *root*. On a Windows NT, Windows 2000 - 2008 server, log in as *Administrator* or from an account that belongs to the *Administrators* group.

b. Run the server access file manager utility ("**acuserve** -access").

c. Be sure to enter the name of the *working* server access file in response to the utility's first prompt.

d. Select menu item [4] - "*Display one/all security records*."

e. Respond no ("N") to the prompt "*Display all records?"*

f. 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") or the Windows NT/2000 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 UNIX password file ("/etc/passwd") or the Windows NT/2000 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 UNIX password file ("/etc/passwd") or the Windows NT/2000 User Manager for the presence of a valid entry for that name.

8.4.6 Invalid Password

Problem: Attempts to connect to the server fail with a 9D, 102, invalid password error.

```
File system error value = 102
Invalid connection password specified by client
```

The causes and remedies of password match failures are straightforward. Invalid password errors occur for two reasons:

1. The password supplied via interactive input does not match the password in the access record.

   These failures are obvious and simply require that you work with your AcuServer system administrator to establish a new password or to have the password requirement removed.

2. The password is supplied by the application via the "Acu_Client_Password" variable, and the value of Acu_Client_Password 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. To change or remove the password requires assistance from the application's technical support group. Contact your application vendor.

8.4.7 Client and Server Both Hang When Connecting

Problem: Attempts to connect to a Windows server cause both the client and the server to hang.

To use Windows NT security with AcuServer rather than AcuServer system security, your network must be set up with the appropriate permissions. If it is not, the client and server may both hang when the offending client attempts to connect.

When you are using Windows NT security, the server actually runs with the permissions of the client temporarily whenever the client requests access to a file. The method used to execute in this mode is called "impersonation" and requires a named pipe connection between the client and the server. A problem occurs when the client does not have the appropriate permissions to connect to the named pipe. Since the named pipe is not the primary means of communication, the client, failing to connect to the named pipe, tells the server it is shutting down. The server, however, is stuck waiting for the client to connect to the named pipe (which it will now never do). With the server

stuck at the named pipe and the client stuck at the socket, both systems are hung. Each must be killed in whatever way your system allows (Task Manager, and so on).

8.5 Frequently Asked Questions

Question: Is AcuServer faster or more secure than using mapped drives in Windows or Novell environments?

Answer: Some operating environments implement aggressive file caching techniques, such as mapped drives, to improve file processing speed. However, errors in these buffering techniques can occasionally lead to corrupted files. AcuServer cannot always match the file processing speeds of the caching methods, but it is faster in certain situations and provides better file integrity. One situation in which the use of file caching techniques (instead of AcuServer) could lead to a caching error and corrupted files is described below.

When an ACUCOBOL-GT runtime asks that a record be rewritten to a file that is shared among multiple clients, the runtime might receive a response indicating that the operating system has correctly rewritten the record, when in fact the record may still be in a memory cache. If a second runtime (on a separate system) then asks for the same record, it will receive a copy of the unchanged record from the shared disk. When this second runtime "rewrites" the record, the data may actually be cached on this second system, just as it was cached on the first. In this situation, somebody's update to the record is going to be lost when the actual rewrites occur. If the rewrites contain index information as well as data, the problem might include structural file damage, leading to "broken" files.

Windows-based clients with mapped drives on a Windows NT server can easily fall into this category because of a caching bug. An ACUCOBOL-GT runtime executing on Windows client machines with data files on mapped drives on a Windows NT server can run faster than AcuServer, because the Windows client systems are caching data writes in their local memory. This approach can give increased file performance as long as the number of clients remains small. This file performance advantage begins to disappear as user counts increase. However, at *any* number of users, this scheme is at risk for

data corruption, which is entirely dependent on timing vagaries. Even with a small user count, the scenario noted in the previous paragraph could occur. AcuServer provides better file integrity than the caching approach.

AcuServer *can* improve file performance in certain situations. For example, we have found that AcuServer is faster than shared drives in situations where two or more users are accessing a single file in an interactive mode. This is because Windows caches mapped disk network files if only one user is in the file, but as soon as another user enters the file caching is terminated. Windows does not seem to know when caching can recommence.

In addition, consider the case of a keyed READ or START to a Vision file that has an average index tree height of four. The root node of the index tree is located by a field in the header, so a read of the header is necessary, followed by an average of four reads to move through the levels of index records, finally followed by a read of the data record. If a client runtime is using Novell or mapped drives, each of those reads must be transmitted across the network, with possibly hundreds of bytes of information returning to the client runtime for each read. If AcuServer is being used, the client runtime calls the appropriate file routine in AcuServer via one socket call. AcuServer knows how to issue all the reads to find the root node of the indexes, to track the needed record through the index tree and to read the data record, and all of this is done on the server with no network traffic. The requested data record is then transmitted back to the client runtime over the network. This approach represents significantly less network traffic than having the client runtime itself do each of the I/Os across the network.

Question: In my Windows NT or Novell environment, I would like AcuServer to serve some of the COBOL objects and data files for my application, but I would also like to access some of the objects and data files via mapped drives. May I do so?

Answer: Yes. You can use CODE_PREFIX, FILE_PREFIX, or file aliases in the client runtime configuration file. For example:

```
CODE_PREFIX @server1:C:\path\sub H:\path\sub
FILE_PREFIX @server1:C:\path\sub H:\path\sub
filea @server1:C:\path\sub
fileb H:\path\sub\fileb
```

The first two examples above rely on the fact that a runtime will use the CODE_PREFIX or FILE_PREFIX entries left to right, in a continuing search to find the COBOL object or data file. The first two examples will direct the client runtime to first ask AcuServer to serve the COBOL objects and data files. If AcuServer can't find them, the runtime is directed to look for them on the mapped drive. You could reverse the search by reversing the sequence of the entries in CODE_PREFIX or FILE_PREFIX. Specifying individual file aliases will be faster than FILE_PREFIX in opening data files.

Question: Which releases of AcuServer are currently supported?

Answer: Our Technical Services continues to offer telephone and fax support for every release of AcuServer and would be pleased to answer your questions about any version of the product. However, all releases of AcuServer prior to Version 5.2 have been "retired" from our sales portfolio. This means that the older versions are no longer sold, and that you obtain a correction to a problem uncovered in a retired version by upgrading your site to the latest version that contains the fix.

Question: Is AcuServer able to print files?

Answer: AcuServer is a file server. This means that it services *file operations*. To print, the runtime submits a *process* to the printer (in the case of Windows NT, it submits a spooler job). Because AcuServer is not intended to service (or start) processes, it is not able to print files to a printer. However, the programmer could use AcuConnect® in conjunction with AcuServer to achieve this functionality. Another option is to assign the print file to disk by establishing an alias for the file in the runtime configuration file with the "-f" flag, as shown in this example:

```
printer1 -f /usr2/users/jsmith/printer1
```

This causes a file named printer1 to be created in the /usr2/users/jsmith directory. This file is a line sequential file with additional formatting characters.

Question: Is AcuServer "multi-threaded"?

Answer: AcuServer is single-threaded. AcuServer processes each request to completion, on a first-come-first-served basis. This means that while AcuServer is engaged in handling a request from one client runtime, other runtime requests are queued in a manner appropriate to the server's operating system, awaiting AcuServer's completion of the current request.

Multiple instances of AcuServer can be executed on the same server or on multiple servers. Each separate instance of AcuServer is a single-threaded program.

Question: AcuServer's response to "-info" is larger than will fit on my screen, and I lose the information that scrolls past the top. How can I see it all?

Answer: Use the properties ability of a command prompt to increase the size of the window or:

On UNIX:

For the sh and ksh shells, use:

```
acuserve –info 2> somefilename
```

for the csh shell, use:

```
acuserve –info >& somefilename
```

On Windows NT:

```
acuserve –info 2> somefilename
```

Then use any editor to view "somefilename".

Question: The entries I previously used for FILE_PREFIX and CODE_PREFIX in the configuration file when AcuServer was on UNIX don't work for Windows NT (or vice versa). Which is correct?

Answer: The entries need to comply with operating system naming conventions.

On UNIX, the use of the colon and the direction of the slash are relevant:

```
FILE_PREFIX @servername:/path
CODE_PREFIX @servername:/path
```

On Windows NT, the direction of the slash is not important, but the drive designation must be included along with the pathname:

```
FILE_PREFIX @servername:C:\path
CODE_PREFIX @servername:C:\path
```

(where *C* is a drive letter)

Question: I'm sure I have sufficient values specified in MAX_FILES, MAX_LOCKS, and LOCKS_PER_FILE in the client configuration file and in the AcuServer configuration file. But I still can't open all the files my application needs, or my application runs out of locks. How can I solve this?

Answer: In most UNIX operating environments, there is a limit to the maximum number of files that can be opened by any process and the maximum number of locks that can be held by any process. These limits are set in the UNIX kernel. If the settings of these kernel parameters are too low, AcuServer may reach these kernel limits before ever reaching the values you have specified in MAX_FILES, MAX_LOCKS, or LOCKS_PER_FILE.

If you have a C compiler on your UNIX system, you might want to make use of a C utility program (numfiles.c) that is available from our Technical Services. This utility can help you determine how many files your UNIX system will allow a process to open. The utility simply attempts to open 1,000 files and reports the number of the last file it was able to open successfully (if you would like to attempt a smaller or larger number of file opens, change the "#define NUM_FILES").

The tuning of the kernel parameters on your UNIX systems is specific to your site and should be determined by your local specialist. Please contact your UNIX system administrator to research or change the values of the kernel parameters that control maximum files per process or maximum locks per process at your site.

Question: I'm sure my client system has a valid connection to the server and that the files I'm trying to reach exist, but my attempts to access the files lead to an "access denied" error (File error 37,07 on @servername:/path/sub/file). How can I solve this?

Answer: This file error indicates that file permissions are probably not set up in a way that allows access. It may stem from variables set in your client system, so let's examine those variables first.

When a client runtime attempts to connect to AcuServer, it passes two variables to AcuServer: the client machine name and client user name. AcuServer checks the values of these two variables against the server access file (called "AcuAccess" by default). AcuAccess contains one or more access records. These access records define which users of which clients are

permitted access to AcuServer. AcuAccess is designed to support a wide range of access security, from very open to very restrictive. You choose the level of security best suited to your needs.

The first variable passed from the client is matched against the AcuAccess field "Client-Machine-Name," and the second is matched against "Client-Username." (See **section 8.3, "Diagnosing Errors with C$PING,"** for a complete description of where those variables come from on various client systems. **Section 6.5, "AcuServer Connection Logic,"** describes how AcuServer uses these variables to determine which account will be used on the server for file access.) AcuAccess will look first for a specific match for the client variables, and (if that's not found) will then look for a wildcard that might allow access to the client. Let's look at a specific example that shows how you can uncover the source of a permissions problem at your site.

Execute AcuServer with a trace file, as shown here:

```
acuserve –start –e errorfile –t3
```

Attempt to connect to AcuServer, then after the failure, examine "errorfile". You should see lines that resemble this:

```
acuserve: 08/13/99 17:48:39 – AUTHORIZATION request from user: klevine
>> Socket:0, client:klevine-ntw, pid:282, PASSWORD-ATTEMPTS allowed:3
>>>key 0 = 'klevine-ntw         klevine         '
>>>key 0 = 'klevine-ntw                         '
>>>key 0 = '*                   klevine         '
>>>key 0 = '*                                   '
>> Mapping user: klevine to local user:guest   acuserve: LookupAccountName for guest
```

In the above example, *klevine-ntw* is the variable from the client that is matched to "Client-Machine-Name," and *klevine* is the variable that is matched to "Client-Username" in AcuAccess.

In the above example, the AcuAccess file on the server had a single access record:

```
Client Machine Name  Client Username  Local Username   Password   Umask
-------------------  ---------------  --------------   --------   -----
*                                     guest            <none>     002
```

Note that the Client-Machine-Name is a wildcard (matches any name). The Client Username is set to spaces. As outlined in **Section 6.5, "AcuServer Connection Logic,"** AcuServer attempts to find an AcuAccess record that has "klevine-ntw" as Client-Machine-Name AND "klevine" as Client-Username. There is no such record in the file.

Next, AcuServer attempts to find a record that has "klevine-ntw" as Client-Machine-Name AND spaces as Client-Username. There is no such record in the file.

Next, AcuServer attempts to find a record that has "*" as Client-Machine-Name AND "klevine" as Client-Username. There is no such record in the file.

Finally, AcuServer attempts to find a record that has "*" as Client-Machine-Name AND spaces as Client-Username. Such a record exists in the AcuAccess file, with the Local-Username set to "guest".

AcuServer verifies that "guest" (in the sample above) is valid on this server. Because "guest" is valid on this server, future file requests from this client will be treated on the server as if "guest" had made the requests.

If the user specified in Local Username of the AcuAccess file is not valid on the server, the AcuServer configuration variable DEFAULT_USER will be used. If this is a valid user on the server, future file requests from this client will be treated on the server as if DEFAULT_USER had made the requests.

(If neither the Local Username of the appropriate record in the AcuAccess file nor DEFAULT_USER is valid on the server, the Authorization Request is denied.)

In our example, suppose we received an "access denied" error (File error 37,07 on @servername:/path/sub/file). We know that the Client-Machine-Name and Client-Username are acceptable for the AcuAccess file shown above. Thus, in this case the error message must indicate that Local Username "guest" doesn't have permission to access the files the application needs.

Question: I'm sure my client system has a valid connection to the server and that the files I'm trying to reach exist, but my attempts to access the files lead to a "File not found" error (File error 35 on @servername:/path/sub/file). How do I solve this?

Answer: This might be caused by insufficient file permissions or by a client runtime that is not AcuServer-enabled.

1. Verify that the client runtime being used is AcuServer-enabled. At a command line prompt, type:

   ```
   <name-of-your-runtime> –v
   ```

   Depending on your client platform and runtime version, the response may include a line like:

   ```
   AcuServer client
   ```

   or

   ```
   Network enabled
   ```

   Absence of a line similar to one of the two shown above indicates that your runtime does not have the ability to communicate with AcuServer. Contact your Micro Focus ***extend*** sales representative to obtain a runtime that does have that capacity.

2. See the discussion in the preceding question about how AcuServer authorizes a client runtime. The most likely cause of this error is that the local user who was finally selected on the server machine does not have "write" permissions in the named directory, and the COBOL program is trying to open the file OUTPUT.

Question: Client runtimes experience slow response times during heavy loads (perhaps sometimes even exhausting the DEFAULT_TIMEOUT value and returning "9D,05"). Is there anything I can do?

Answer: Here are several options to consider:

1. AcuServer is single-threaded, handling a single request from a single client runtime to completion before attending to requests from other client runtimes, which are queued up in a manner dependent on the

server's operating system. Slow response time may indicate that a single copy of AcuServer cannot keep ahead of the queued requests at times of peak load.

If the load on AcuServer is variable, the DEFAULT_TIMEOUT tolerance allowed by client runtimes may be sufficient most of the time, but may be too brief to allow for the times when AcuServer is heavily burdened. In those cases, setting DEFAULT_TIMEOUT to a higher value can avoid the timeout errors. (On the other hand, if AcuServer is actually unable to access the data for some other reason–such as insufficient file permissions—then increasing DEFAULT_TIMEOUT only increases the amount of time that transpires before the client runtime returns the error message.)

If there are sufficient system resources available on the server (memory, CPU cycles, I/O bandpass to multiple disks, and so forth), you may gain relief by acquiring an appropriate license and executing multiple instances of AcuServer on the same server. On the server, the operating system will use the time while one instance of AcuServer is waiting for the completion of an I/O process to allow another instance of AcuServer to have the CPU. This second instance of AcuServer will also then issue some I/O command of its own, and (assuming that there are multiple I/O subsystems capable of independent action), the I/O wait time of the multiple instances will interleave. A server with multiple CPUs may gain even more improvement with this approach.

If you choose to run multiple instances of AcuServer, please note that there is no automatic balancing of the load among them. You must direct each specific client runtime to use a particular AcuServer by (a) starting each instance of AcuServer with a particular ACUSERVER_PORT number and (b) setting the client runtime configuration variable of the same name to that value.

2. You can choose to execute multiple instances of AcuServer on separate servers. In this case, you need not start each AcuServer with a separate ACUSERVER_PORT setting, as noted above. Rather, you would use FILE_PREFIX or file aliases in the client runtime configuration file to direct specific client runtimes to AcuServers running on specific servers. For example:

In the runtime configuration file of client A:

```
FILE_PREFIX @server1:/path/sub
```

In the runtime configuration file of client B:

```
FILE_PREFIX @server2:/path/sub
```

3. You could combine approaches (1) and (2) and run multiple instances of AcuServer on multiple servers. In that case, each instance of AcuServer would need to be started with a ACUSERVER_PORT unique on that server. FILE_PREFIX or file aliases would be needed in each client runtime configuration file to point to the appropriate server. Also, ACUSERVER_PORT would need to be set to point to the appropriate instance of AcuServer on that server.

4. If there are no runtimes executing on the server, or if the runtimes executing on the server don't use the same files that are being requested by the client runtimes, you can improve performance by directing AcuServer to open data files for exclusive use. To do this, set the AcuServer configuration variable LOCK_ALL_FILES to "1". With this variable on, AcuServer will open data files for exclusive use rather than using individual record locks; this will improve throughput. Please note that this is not a workable option if multiple instances of AcuServer will be executed on the same server and will be accessing the same data files.

   (If the runtimes executing on the server do use the same files as client runtimes, you can direct the runtimes on the server to also use AcuServer, rather than opening the files themselves, by setting USE_LOCAL_SERVER to "1" in their configuration files. This will allow use of LOCK_ALL_FILES as noted above, so AcuServer can again open the data files for exclusive use, which is faster.)

5. If client runtimes need to open files that may be located in various directories on one server or may be located on various servers, it is faster to use file aliases rather than a FILE_PREFIX that points to multiple paths or multiple servers. A great deal of network traffic can be avoided in those cases where the file that a client runtime is seeking happens to be in the last directory or on the last server specified in FILE_PREFIX. For example:

```
FILE_PREFIX @server1:/path1a/sub1a @server1:/path1b/sub1b @server1:/path1c/sub1c
```

The above method would be slower at opening files than:

```
filea @server1:/path1a/sub1a/filea
fileb @server1:/path1b/sub1b/fileb
filec @server1:/path1c/sub1c/filec
```

And notice:

```
FILE_PREFIX @server1:/path/sub @server2:/path/sub @server3:/path/sub
```

The above settings would be slower at opening files than:

```
filea @server1:/path/sub/filea
fileb @server2:/path/sub/fileb
filec @server3:/path/sub/filec
```

6. The AcuServer server configuration file can use the following variables:

   ```
   V_BUFFERS
   V_BUFFER_DATA
   V_READ_AHEAD
   ```

   Setting those variables in the AcuServer server configuration file overrides any setting of the identically named variables in the client runtime configuration file. These variables have the same function on AcuServer as they have on a client runtime, and they can affect file performance. Consult Appendix H in the ACUCOBOL-GT manual set for specifics.

7. You have chosen AcuServer for its ability to serve COBOL object files and data files for your application. However, these data accesses may be overloading your network. If analysis of network performance indicates that network overloading is contributing to your application response time slowdown, consider that some data files in your application might be able to reside on the client system. The goal is to reduce the amount of traffic over the network. Some possibilities in this regard are:

   a. If the application creates temporary files, also consider placing them on a local drive. If the file is useful only on a particular client, and need never be seen by other client systems on the network, you can reduce the network burden by using a file alias in the client configuration file to place this file on a local drive.

b. If the application has some data files that do need to be seen by every client runtime, but are only updated periodically, consider placing copies of these files on each client system and pointing at them with file aliases. Again the goal is to reduce network traffic. You would have to balance the administrative work of getting correct copies of these files placed on each client in a timely fashion against the gain in network performance by keeping access to these files off the network.

Glossary of Terms

ACL

Access Control List; used byWindows NT and Windows 2000 - 2008 to control access to resources. An ACL specifically grants access to a user or to a group. Privileges are additive, meaning that users have the highest access given to their account and to any groups to which they belong. The exception is “No Access,” which overrides any other privileges.

ACP

AcuServer Control Panel; a Windows graphical user interface for performing AcuServer functions.

acushare

License manager for UNIX servers.

daemon

A memory resident program.

multiple-record mode

A mode in which AcuServer returns multiple records in a single network packet rather than one record at a time. Designed for reading a large number of sequential records without user interruption, such as when you are generating a report. Files opened in this mode are optimized for sequential reading, but have restrictions placed on the types of operations they can do.

name alias

A substitute string for the literal name that appears in the ASSIGN TO clause of a SELECT statement.

NTFS

NT file system, the file system used by Windows NT.

remote name notation

Notation used to refer to files located on the server.

server access file

Access database that specifies which client systems and client users are permitted to use AcuServer and what their user privileges are.

umask

File creation mask. The *read* and *write* permissions set on new files are determined by the *umask* specified in the matching server access record.

On UNIX servers, 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.

On Windows NT and Windows 2000 - 2008 servers, the *umask* is used to set read and write permissions for the *owner* (leftmost digit) and for *other* (rightmost digit). The *group* digit (middle) is ignored.

XFD

eXtended File Descriptor. A data dictionary useful for translating COBOL records into other data formats.

Index

Numerics

9D errors 8-6

A

-a option, -unlock argument 5-27
a_srvcfg file 4-8
    ownership and permissions 2-7
access control lists (ACLs) 3-9
access denied, reasons for 6-12, 8-12
access file manager 6-9
    adding an access record 6-11
    default values 6-10
    displaying records 6-17
    exiting 6-18
    modifying records 6-15
    removing records 6-15
    starting 5-5
access file, creating or opening 6-10
-access option, starting access file manager from command line 5-5, 6-9
access records
    adding 6-11
        from command line 6-13
        with ACP 6-15
    displaying 6-17
    examples of 6-7
    fields in 6-6
    modifying
        from command line 6-16
        with ACP 6-16
    passwords 6-12

access records (continued)
removing 6-15
wildcards in 6-7
access, restricting on UNIX 4-28
ACCESS_FILE configuration variable 4-11
accessing remote files, name aliases 7-7
ACL (access control list) 3-9
ACP (AcuServer Control Panel)
adding access records with 6-15
changing **acuserve** properties using 5-19
checking system status with 5-21
closing stranded files with 5-31
configuring **acuserve** with 5-19
modifying access records with 6-16
starting and stopping **acuserve** using 5-16
using to install **acuserve** 5-8
Acu_Client_Password 2-10, 3-12, 6-20, 8-24
AcuAccess, file ownership and permissions 2-7
ACUCOBOL-GT Web Runtime, and -kill command 5-16
acuping 8-8
AcuPing, using to confirm network services 3-14
acuserve
Windows 2008 5-4
acuserve
access file manager 5-5
automatic startup on UNIX 5-15
command 5-2
configuring 5-17
daemons 4-11
enabling file tracing 5-14
error log 5-12
file permissions 1-7, 2-7
UNIX 2-7
Windows 2-9, 3-9
installing as Windows service 5-5
ACP 5-8
command line 5-6

acuserve (continued)
port number, specifying 4-26
querying the Windows service 5-10
removing the Windows service 5-5, 5-9
secondary servers 4-11
starting 5-11
ACP 5-16
command line 5-11
status report 5-19
ACP 5-21
command line 5-20
stopping 5-11
ACP 5-16
command line 5-15
using under MPE/iX 2-4
version number of 5-31
will not start 2-8, 5-8, 6-5
UNIX 6-4
Windows 3-3
Windows NT/2000 service 5-5
acuserve command formats 5-3
acuserve daemon, specifying IP address for 4-25
acuserve properties
changing from the command line 5-17
changing using the ACP 5-19
acuserve -vv command 4-13
acuserve.err file 5-8
AcuServer
access security 4-11, 4-15, 4-22
configuring 4-2
encryption, turning on 4-5
errors 8-8
event log message capability, adding in Windows 4-31
file ownerships and permissions 2-7
installing and configuring
UNIX 2-2
Windows 3-2

AcuServer (continued)
programming considerations 7-2
security 6-20
system security 6-3
using password to deny access 6-5
will not start 3-10
AcuServer Control Panel (ACP) 3-14, 5-4
AcuServer errors, diagnosing with **acuping** 8-8
ACUSERVER_MASTER_SERVER configuration variable 4-11
ACUSERVER_PORT configuration variable
assigning port number to **acuserve** 5-13
defined 4-26
FILE_PREFIX and load balancing 4-12
load balancing 5-6
using FILE_PREFIX instead of 8-33
AcuServer-enabled runtime, verifying
UNIX 2-2, 2-10
Windows 3-2, 3-12
acushare, defined 2-6
AGS_PING_TIME configuration variable
connection detection mechanism 8-2
defined 4-3
with DEAD_CLIENT_TIMEOUT 4-13
AGS_SERVER_SOCKET_RESERVE configuration variable 4-12
AGS_SOCKET_COMPRESS configuration variable 4-5
AGS_SOCKET_ENCRYPT configuration variable 4-5, 6-21
aliases
and remote name notation 7-6
name 7-7
not used in library functions 7-19
ALL parameter, server history 5-22
APPLY_CODE_PATH runtime configuration variable 7-5
APPLY_FILE_PATH runtime configuration variable 7-5
AS_CLIENT_ENCRYPT, replaced by AGS_SOCKET_ENCRYPT 4-5
assigning a port to **acuserve** 5-8

B

buffers, V_BUFFERS 4-28

C

-c option
    remote name notation 7-6
    specifying full path and name of configuration files 4-3, 5-12
    specifying server configuration files on command line 4-8
    -unlock argument 5-27
C$COPY library routine, name notation restrictions 7-18
C$DISCONNECT library routine 8-3
C$FILEINFO library routine 7-20
C$PING library routine 8-8
C$RERR library routine 8-6
CACHE_DIRECTORY configuration variable 4-6
CACHE_DIRECTORY_SIZE configuration variable 4-6
cached object files, storing 4-6
cblconfig file 4-3
chgrp command 2-7
chmod command 2-7
    umask values and 2-9
chown command 2-7
client
    allocating indexed block buffers on 4-28
    detecting a dead Windows connection on 8-2
client and server on same machine 4-7
client hang caused by inappropriate permissions 8-24
client load, balancing 5-6
client machine name, confirming 8-22
client passwords 2-10, 3-12
client user name 6-7
    confirming 8-20, 8-22
client-enabled runtime, verifying
    UNIX 2-2, 2-10
    Windows 3-2, 3-12

closing stranded files 5-26
ACP 5-31
command line 5-26
cmd command 5-18
CODE_PREFIX configuration variable
and hard-coded file paths 1-2
enabling applications to use AcuServer 7-3
maximum length of 7-6
remote name notation 7-6
updating 7-8
command line, configuring **acuserve** on 5-17
commands, printing a list of those available 5-18
compressing socket data 4-5
compression, gzip 5-13
-config option
changing configuration of a running **acuserve** 5-17
enabling tracing 4-15, 8-19
configuration file format 4-2
configuration file, specifying for **acuserve** 5-8
configuration variables
remote name notation examples 7-7
runtime 4-3
SECURITY_METHOD 4-28
server 4-8
configuration variables, list of
ACCESS_FILE 4-11, 4-15
ACUSERVER_MASTER_SERVER 4-11
ACUSERVER_PORT 4-12, 4-26, 5-6, 5-13, 8-33
AGS_PING_TIME 4-3, 4-13, 8-2
AGS_SERVER_SOCKET_RESERVE 4-12
AGS_SOCKET_COMPRESS 4-5
AGS_SOCKET_ENCRYPT 4-5, 6-21
APPLY_CODE_PATH 7-5
APPLY_FILE_PATH 7-5
CACHE_DIRECTORY 4-6
CACHE_DIRECTORY_SIZE 4-6
CODE_PREFIX 1-2, 7-3, 7-6, 7-8

configuration variables, list of (continued)
COUNT_STATISTICS 4-13, 5-22
DEAD_CLIENT_TIMEOUT 4-3, 4-13, 8-2
DEFAULT_MAP_FILE 1-10
DEFAULT_PROGRAM 7-6
DEFAULT_TIME_OUT 8-12
DEFAULT_TIMEOUT 4-14, 8-33
DEFAULT_UMASK 4-14
DEFAULT_USER 4-15, 6-7, 6-12, 6-18
ENCRYPTION_SEED 4-15, 6-21
FILE_PREFIX 1-2, 4-12, 7-3, 7-6, 7-8, 7-19, 8-7
FILE_TRACE 4-15, 5-14
FILE_TRACE_FLUSH 4-15, 5-14
FILE_TRACE_TIMESTAMP 4-16
*filename*_DATA_FMT 4-10
*filename*_DATA_FMT 4-16
*filename*_INDEX_FMT 4-10, 4-18
*filename*_LOG 7-6
*filename*_MRC 4-7, 4-22, 7-14
*filename*_VERSION 4-10, 4-19
HOT_KEY 7-6
LOCK_ALL_FILES 4-20, 8-34
LOCKS_PER_FILE 1-6, 4-20, 8-29
LOG_DIR 7-6
LOG_FILE 7-6
MASS_UPDATE 4-29
MAX_ERROR_LINES 4-21
MAX_FILES 1-6, 4-20, 8-29
MAX_LOCKS 1-6, 4-20, 8-29
MAXIMUM_RECORD_SIZE 7-16
MULTIPLE_RECORD_COUNT 4-7, 4-22, 7-14
NO_LOCAL_CACHE 4-22
NT_SECURITY 2-10, 3-11
PASSWORD_ATTEMPTS 4-11, 4-15, 4-22, 4-24, 6-21
PROVIDE_PASSWORD_MESSAGES 4-23
SECURITY_METHOD 2-10, 4-23, 4-31, 6-2, 6-4, 6-18
SERVER_IP 4-25

configuration variables, list of (continued)
*server*_MAP_FILE 1-10
SERVER_NAME 4-25
*server*_PASSWORD 6-21
*server_port*_PASSWORD 6-21
SORT_DIR 7-6
TEXT 4-27, 6-21
USE_LOCAL_SERVER 4-7, 4-20, 8-34
USE_SYSTEM_RESTRICTIONS 4-24, 4-28
USER 6-6, 6-7, 8-18, 8-20
USERNAME 6-6, 6-7, 6-11, 8-18, 8-20
V_BASENAME_TRANSLATION 4-10, 4-28
V_BUFFER_DATA 4-10, 4-28
V_BUFFERS 4-10, 4-28
V_INDEX_BLOCK_PERCENT 4-10, 4-29
V_READ_AHEAD 4-10, 4-29
V_SEG_SIZE 4-10, 4-30
V_STRIP_DOT_EXTENSION 4-10, 4-30
V_VERSION 1-10, 4-10, 4-19, 4-30
WINNT_LOGON_DOMAIN 4-24
WINNT-EVENTLOG-DOMAIN 4-31, 8-5
WINNT-LOGON-DOMAIN 4-31
XFD_DIRECTORY 7-6
confirming network services 2-12, 3-14
connection attempts, limiting 4-23
connection refused 6-18, 8-22
connection times out 8-11
connection validation logic 6-18
Control + C, application terminated by 5-26
COUNT_STATISTICS configuration variable 4-13, 5-22
CPING-CONN-REFUSED 8-9
CPING-NO-CLIENT 8-9
CPING-OK 8-9
CPING-PARAM-ERROR 8-9
CPING-SOCKET-ERROR 8-10
CPING-VERSION-ERROR 8-10
creating an access file 6-10

D

-d option
    turning on debugger 8-14
    using with -start to track memory usage 5-12, 5-22
daemons, distributing client requests among 4-11
data access speed, improving 4-20
dead client detection mechanism 8-2
DEAD_CLIENT_TIMEOUT configuration variable 4-3, 4-13, 8-2
debugging 8-14
    ping messages and 4-4
DEFAULT_MAP_FILE configuration variable 1-10
DEFAULT_PROGRAM configuration variable
    remote name notation 7-6
DEFAULT_TIME_OUT configuration variable
    using to confirm that **acuserve** is running 8-12
DEFAULT_TIMEOUT configuration variable 4-14, 8-33
DEFAULT_UMASK configuration variable 4-14
DEFAULT_USER configuration variable 4-15, 6-7, 6-12
    when used 6-18
dependencies, specifying for **acuserve**
    ACP 5-9
    command line 5-7
detection mechanism for terminated connections 8-2
diagnosing common system problems 8-10
disconnecting from the server 8-3
disk space, increasing 4-29
displaying the value of a variable 5-17

E

-e option
    directing error output 8-14
        command line 5-12
        used with -start 8-4
    error messages and 8-10
    remote name notation 7-6

encryption 1-7, 6-21
encryption module, initializing 4-15
ENCRYPTION_SEED configuration variable 4-15, 6-21
End of log message 4-22
environment variables
    CODE_PREFIX 1-2
    FILE_PREFIX 1-2
    server_PASSWORD 6-21
    server_port_PASSWORD 6-21
    USER 2-12
    USERNAME 2-12
error codes 8-6
error file
    compressing 5-13
    maximum number of lines in 4-21
    saving information about OPENs, READs, AND WRITEs in 4-15
error log 5-12
error logging 8-4
error output, location of for **acuserve** 5-8
error trace buffer, flushing to disk after each WRITE 4-15
error-file, placing file trace and error output in 8-14
errors
    AcuServer 8-7, 8-8
    client runtime 8-8
escape sequences, defined 4-17
event log, failure to output 8-5
event logging 8-5
exiting the access file manager 6-18
exiting the configure mode 5-18

F

-f option
    and FID 5-27
    and -info 5-20
    and -install 5-7

-f option (continued)
and -kill 5-15
running **acuserve** in foreground 5-11, 5-13
FAQs (frequently asked questions) 8-25
FAT file system 1-7, 6-2
FID, field in status report 5-20
file access failures 2-9, 8-6
file errors, ambiguous 8-10
file extensions, preventing removal of 4-30
FILE parameter, server history 5-22
file permissions 4-14
on Windows systems 3-10
file systems on Windows NT/2000 6-2
file trace information, not losing with unexpected termination of AcuServer 4-15
FILE TRACE, turning on for debugging 8-14
file tracing, server 5-14
file type, in status report 5-20
FILE_PREFIX configuration variable
enabling applications to use AcuServer 7-3
hard-coded file paths 1-2
length restrictions 7-6
maximum length of 7-6
not used in library functions 7-19
remote name notation 7-6
server load balancing 4-12
syntax 8-7
updating 7-8
FILE_TRACE configuration variable 4-15, 5-14
FILE_TRACE_FLUSH configuration variable 4-15, 5-14
FILE_TRACE_TIMESTAMP configuration variable 4-16
*filename*_DATA_FMT configuration variable 4-10, 4-16
*filename*_INDEX_FMT configuration variable 4-10, 4-18
*filename*_LOG configuration variable, remote name notation 7-6
*filename*_MRC configuration variable
and MULTIPLE_RECORD_COUNT 4-22
and multiple-record mode 7-14
defined 4-7

*filename*_VERSION configuration variable 4-10, 4-19
filenames, including full path information in 4-28
files
    getting list of open 5-19
    locking 4-20
    opening in multiple-record mode 4-7
firewall restrictions 4-26
foreground, running **acuserve** in 5-13
format
    timestamp 4-16
    Vision file 4-16

G

-g option, compressing files 5-13
GET command 5-17
getting started
    UNIX 2-2
    Windows 3-2
group access accounts 6-5
gzip compression 4-5, 5-13

H

HELP command 5-18
host name 2-11
    of client system 8-21
    setting up 3-12
        UNIX 2-11
        Windows 2000/2003/XP/Vista 2-11
        Windows NT 2-11
    verifying on Windows 2000/2003/XP 2-11
HOT_KEY configuration variable, remote name notation 7-6

I

-i option, remote name notation 7-7
impersonation method of security 8-24
improving performance. *See* performance, improving
-info option, generating AcuServer system status report 5-20
installing AcuServer 3-5
    UNIX 2-2, 2-5
    Windows 3-2
instances, maximum **acuserve** allowed 1-8
international character handling 1-10
IP address, specifying 4-25

K

-k option, remote name notation 7-7
-kill option
    and ACUCOBOL-GT Web Runtime 5-16
    application terminated by 5-26
    terminating **acuserve** process 5-15

L

-l option
    -start argument 5-13
    using with debugger 8-14
license file
    defined for UNIX 2-5
    defined for Windows 3-5
    runtime 1-8
licensing 1-8
LIST command 5-18
load-balancing 4-11
    re-registering servers 5-32
local caching, preventing 4-22
local files, accessing with AcuServer 4-7

Local Username field 4-15
LOCK_ALL_FILES configuration variable 4-20, 8-34
locking files 4-20
LOCKS_PER_FILE configuration variable
    defined 4-20
    set high enough but run out of locks 8-29
    tuning record locking support 1-6
LOG_DIR configuration variable, remote name notation 7-6
LOG_FILE configuration variable, remote name notation 7-6
logging errors 8-4
logging events 8-5
LOGON mode of security 4-24
LOGON security 6-19
    passwords 6-19
ls -1, using to access a directory 8-15

M

machine failures 8-2
    DEAD_CLIENT_TIMEOUT and 4-13
    DEFAULT_TIMEOUT and 4-14
maintaining server name files 7-8
man printf, UNIX escape sequence definitions 4-17
map file, for international character handling 1-10
MASS_UPDATE configuration variable 4-29
master server 4-11
MAX_ERROR_LINES configuration variable 4-21
MAX_FILES configuration variable
    defined 4-20
    set high enough but run out of locks 8-29
    tune record locking support 1-6
MAX_LOCKS configuration variable
    defined 4-20
    set high enough but run out of locks 8-29
    tune record locking support 1-6
maximum size, setting for Vision file segments 4-30

MAXIMUM_RECORD_SIZE configuration variable, multiple-record mode and 7-16
memory
    getting information about 5-22
    multiple-record mode 7-16
    saving 4-28
    tracking usage of 5-12
MEMORY parameter, server history 5-22
messages
    changing in server configuration file 4-27
    controlling text for runtime 4-27
    sending event log on Windows servers 4-31
MPE/iX, running AcuServer in 2-4
MULTIPLE_RECORD_COUNT configuration variable
    and multiple-record mode 7-14
    defined 4-22
    *filename*_MRC 4-7
multiple-record mode
    and programming 7-14
    defined 1-6
    *filename*_MRC 4-7
    limitations 7-15
    memory issues 7-16
    MULTIPLE_RECORD_COUNT 4-22
    race conditions 7-16
multi-record mode. *See* multiple-record mode

N

-n option
    -info argument 5-21
    -kill argument 5-16
    overriding ACUSERVER_PORT 4-26
    -remove argument 5-9
    specifying **acuserve** service number with 5-7
    -start argument 5-13
    -unlock argument 5-27

name aliases
    and remote name notation 7-6
    defined 7-7
    updating 7-8
    using to improve performance 7-3
named pipe connection failure 8-24
named pipe security 6-18
    passwords 6-12, 6-19
NAMED-PIPE security option 4-25, 6-6
native system security 4-24, 6-4
network maintenance, simplifying 7-8
NETWORK parameter, server history 5-22
network performance, debugging 8-8
network services, confirming 2-12, 3-14
NO_LOCAL_CACHE configuration variable 4-22
NT_SECURITY configuration variable 2-10, 3-11
NTFS (NT file system) 1-7, 6-2

O

-o option, remote name notation 7-7
open files, getting list of 5-19
opening an access file 6-10
overriding file permissions 3-10
overriding the server access file 6-2

P

-p option, -unlock argument 5-27
password error, but user not prompted for password 8-24
password field in access record 6-12
password invalid, connection refused 8-23
password messages, sending to client on initial connection 4-23
password validation attempts, limiting 4-23
PASSWORD_ATTEMPTS configuration variable
    and LOGON value of SECURITY_METHOD 4-24

PASSWORD_ATTEMPTS configuration variable (continued)
defined 4-22
password verification cycle 6-21
passwords 6-19
Acu_Client_Password 2-10, 6-20
client 2-10, 3-12
using to deny access to AcuServer 6-5
performance
impacts on
AGS_PING_TIME 8-3
encryption 1-7, 6-21
LOCKS_PER_FILE and MAX_FILES 4-21
share names 7-4
timestamps 4-16
improving 4-22
balancing client load 5-6
indexed block buffers 4-28
local caching 4-6, 4-22
locked files 4-20
multiple-record mode 7-14
on UNIX 4-12
UNIX 4-12
Vision file segment size 4-30
Vision indexed file data blocks 4-28
Vision read-ahead logic 4-29
network 8-8
optimizing 1-8
permissions
access denied 8-13
acuserve services 6-3
file, on Windows systems 3-10
inappropriate cause client/server hang 8-24
named pipe 4-25
read and write on Windows systems 3-11
UNIX on AcuServer files 6-3
verifying on Windows NT/2000/2003 8-16
verifying UNIX 8-15

PID, field in status report 5-20
ping 2-12
    using to confirm network services 3-14
ping messages and the debugger 4-4
port number
    assigning to **acuserve** 5-8, 5-13
    specifying to access **acuserve** 4-26
primary server 4-11
printf escape sequences 4-17
privileges, user, on Windows systems 3-9
process ID, in status report 5-20
PROVIDE_PASSWORD_MESSAGES configuration variable 4-23

Q

-query option 5-10
QUIT command 5-18

R

-r option, remote name notation 7-7
race conditions and multiple-record mode 7-16
read permissions
    setting 4-14
    setting on Windows systems 3-11
record locking support 1-6
-register server command 5-32
registering servers 5-32
RELATIVE parameter, server history 5-24
remote files, accessing with name aliases 7-7
remote name notation
    accessing files with 4-7
    configuration variable examples 7-7
    referencing remote files with 7-2
    specifying files with 7-6
remote search path, adding 7-4

-remove option 5-9
RENAME library routine 7-20
RENAME-STATUS 7-21
reports
    generating using multiple-record mode 7-14
    system status 5-19
        opening from command line 5-20
RUNLENGTH compression 4-5
runtime configuration file 4-4
    including contents of in error output 8-14
runtime configuration variables 4-3
runtime messages, controlling text of 4-27

S

same machine for client and server 4-7
search paths, adding remote 7-4
secondary servers 4-11
security
    impersonation method 8-24
    system 6-2
    user account control 3-3, 3-4
    using native system 6-4
    via license files
        UNIX 2-5
        Windows 3-5
    Windows 2008 6-2
    Windows vs. AcuServer 3-9
security method, specifying 4-23
SECURITY_METHOD configuration variable
    and LOGON 4-31
    defined 4-23
    named pipe 6-18
    UNIX 2-10, 4-28
    using native security 6-2, 6-4
segment escape sequences 4-16
SEQUENTIAL parameter, server history 5-25

server
    allocating indexed block buffers on 4-28
    closing all open files at once 5-27
    collecting statistics for 4-13
    confirming it is active 8-11
    disconnecting from 8-3
    on same machine as client 4-7
    specifying for **acuserve** 5-6
server access file
    access records 6-6
    default name 6-4
    ownership and permissions 2-7
server configuration file 4-8
    ownership and permissions
        security 2-7
        UNIX 2-7
        Windows 3-9
    specifying at startup 5-12
server configuration variables 4-8
    getting settings of 5-13
    listed 4-8
server count exceeded 1-8
server hang caused by inappropriate permissions 8-24
server history 5-22
server history parameters, listed 5-22
server name file 7-8
server statistics 4-13
SERVER_IP configuration variable 4-25
*server*_MAP_FILE configuration variable 1-10
SERVER_NAME configuration variable 4-25
*server*_PASSWORD configuration variable 6-21
*server_port*_PASSWORD configuration variable 6-21
Service Properties dialog 5-8
SET command 5-18
SET ENVIRONMENT and AGS_SOCKET_COMPRESS 4-5
SET ENVIRONMENT command 4-8
shared directories and file security 6-3

slave servers 4-11
socket data, compressing 4-5
sockets, reserving on UNIX 4-12
software distribution media contents
    UNIX 2-5
    Windows 3-4
software installation
    client 2-10, 3-12
    UNIX server 2-6
    Windows NT/2000 servers 3-6
SORT_DIR configuration variable, and AcuServer 7-6
specifying a configuration file for **acuserve** 5-8
specifying a port number for **acuserve** 5-9
-start option
    at system startup 5-15
    command line 5-11
    status codes 8-5
    using with **acuserve** 5-7
starting access file manager 5-5
starting **acuserve** 5-11
    ACP 5-16
    command line 5-11
-statistics option, used with -d option 5-12
status codes 8-5
STATUS-CODE 7-20
stopping **acuserve** 5-11
    ACP 5-16
    command line 5-15
stranded files, closing 5-26
syslog 8-5
system load balancing 1-8
system performance 1-7
system problems, diagnosing 8-10
system security
    AcuServer 6-3
    introduction 1-7
    overview 6-2

system status report 5-19
ACP 5-21
command line 5-20

T

-t option
and -f option, to display tracing and transactions to screen 5-13
file tracing and transaction logging 5-14
FILE_TRACE 4-15
termination of **acuserve**, unexpected 5-8, 5-12
TEXT configuration variable 4-27, 6-21
text, controlling for runtime messages 4-27
TIME parameter, server history 5-23
timestamp format 4-16
tracing
enabling with the -config option 8-19
specifying for **acuserve** 5-8
transactions, managing 4-19
translating international characters 1-10
troubleshooting 8-10
access denied 6-12
acuserve won't start, UNIX 6-4
AcuServer won't start 3-10, 6-5
ambiguous file error 8-10
client and server hang 8-24
connection refused 6-12, 6-18, 8-22
connection times out 8-11
file access denied 8-12
invalid password 8-23
permissions and NAMED PIPE value 4-25
unexpected termination of **acuserve** 5-12
unexpected user name 8-19
UNIX permissions 8-15
Windows NT/2000/2003 permissions 8-16
tuning record locking support 1-6
TY, field in status report 5-20

U

-u option, -unlock argument 5-27
umask
    defined for UNIX 2-9
    defined for Windows 3-11
    field in access records 6-6, 6-12
unexpected termination of **acuserve** 5-8
unexpected user name on server 8-19
unique file process IDs 5-20
UNIX
    automatic startup of **acuserve** on 5-15
    confirming client user name 8-17, 8-20
    confirming host name 8-21
    host name 2-11, 3-13
    logon restrictions 4-28
UNIX ownerships and permissions 2-7
    AcuServer files 2-7, 6-3
    data files 2-8
    new files 2-8
    verifying user permissions 8-15
UNIX performance, improving 4-12
UNIX security, passwords 6-20
unlock function 8-3
-unlock option
    and PID 5-20
    closing stranded files 5-26
    command-line mode 5-30
    interactive mode 5-27
USE_LOCAL_SERVER configuration variable
    and LOCK_ALL_FILES 4-20, 8-34
    defined 4-7
USE_SYSTEM_RESTRICTIONS configuration variable 4-24, 4-28
user access privileges, AcuServer 3-10
user account control 3-3, 3-4
user accounts, setting up on Windows 3-8

USER environment variable
    access records and 6-6, 6-7
    confirming user name 8-18
    setting up user name 2-12
    unexpected user name and 8-20
user name
    confirming for file access 8-17
    setting up 2-12
user privileges on Windows systems 3-9
USERNAME environment variable
    access records and 6-6, 6-7, 6-11
    confirming user name 8-18
    setting up user name 2-12
    unexpected user name and 8-20

V

-v option, checking runtime version 2-10
V_BASENAME_TRANSLATION configuration variable 4-10, 4-28
V_BUFFER_DATA configuration variable 4-10, 4-28
V_BUFFERS configuration variable 4-10, 4-28
V_INDEX_BLOCK_PERCENT configuration variable 4-10, 4-29
V_READ_AHEAD configuration variable 4-10, 4-29
V_SEG_SIZE configuration variable 4-10, 4-30
V_STRIP_DOT_EXTENSION configuration variable 4-10, 4-30
V_VERSION configuration variable
    and *filename*_VERSION 4-19
    building Vision Version 3 or 4 files 1-10
    defined 4-30
    ignored when files accessed through AcuServer 4-10
validating service requesters 6-18
values, setting for variables 5-18
variables, listing names and values of 5-18
version number, specifying for Vision data files 4-30
-version option, getting **acuserve** version information 5-31
version, displaying **acuserve** release version 5-31

Vision
    including full path information in filenames 4-28
    specifying version number for data files 4-30
Vision file segments, setting maximum size of 4-30
Vision files 1-9
    naming data segments of 4-16
    naming index segments of 4-18
    reducing size of 4-29
VISION parameter, server history 5-24

W

-w option
    displays Opened field on command line 5-20
    wide status report, using to create 5-21
Web Runtime, using with AcuServer 1-8
wildcards in access records 6-5
    which fields can have 6-7
Windows
    AcuServer Control Panel (ACP) 5-4
    confirming client user name 8-18, 8-20
    detecting a dead connection 8-2
    installing **acuserve** using ACP 5-8
    setting up user accounts 3-8
Windows 2000/2003/XP, host name 2-11, 3-12
Windows 2008
    accessing AcuServer utilities 5-4
    security 6-2
Windows event log 8-5
Windows NT
    host name 2-11, 3-13
    security 3-11
Windows NT/2000
    ownerships and permissions 3-9
    permissions on new files 3-11
    restricting file access 3-9
    security 3-9, 4-23

Windows NT/2000 services
    installing and removing 5-5
    removing 3-7
    starting and stopping 5-11
WINNT_LOGON_DOMAIN configuration variable 4-24
WINNT-EVENTLOG-DOMAIN configuration variable 4-31, 8-5
WINNT-LOGON-DOMAIN configuration variable 4-31
write permissions
    setting on Windows systems 3-11
    setting with DEFAULT_UMASK 4-14
WSOCK32.DLL 3-5

X

-x option
    9D errors 8-6
    getting extended error code 8-10
    using to get full file name and path 8-14
XFD, international character translation 1-10
XFD_DIRECTORY configuration variable, remote name notation 7-6

Y

-y option, remote name notation 7-6

Z

ZLIB compression 4-5