The TimeSafe® Configuration Management System

AccuRev Administrator's Guide

Version 3.8

August, 2005

AccuRev Administrator's Guide
August, 2005

Copyright © AccuRev, Inc. 1995–2005
ALL RIGHTS RESERVED

TimeSafe, **AccuRev**, and **StreamBrowser** are registered trademarks of AccuRev, Inc.
Java is a trademark of Sun Microsystems, Inc.
Rational and **Rational Rose** are trademarks of IBM Corporation.

Table of Contents

The AccuRev Repository

The AccuRev Server program manages a data repository, which provides long-term storage for your organization's development data — for example, all versions of all source files. (Among other things, AccuRev is a special-purpose database management system; the files in the repository — thousands of them — are part of this database.) By default, the repository resides in subdirectory **storage** of the AccuRev installation directory. The repository consists of:

- **site_slice** directory: implements a database that contains a user registry, list of depots, list of workspaces, and other repository-wide information.
- **depots** directory: contains a set of subdirectories, each storing an individual depot. A depot subdirectory stores one or both of:
  - A version-controlled directory tree: all the versions of a set of files and directories, along with a database that keeps track of the versions.
  - A database of Dispatch issue records.

When it starts, the Server program determines the location of the **site_slice** directory by looking at the SITE_SLICE_LOC setting in configuration file **acserver.cnf**. This file must reside in the same directory as the Server program (**accurev_server**) itself.

Repository Access Permissions

The user identity of the AccuRev server process — **acserver** (Unix) or **System** (Windows) — must have full access to all the files and directories within the data repository. For maximum security, this should be the *only* user identity with permission to access the repository. The only exception to this might be an **acadmin** AccuRev administrator account, as suggested in *Unix Systems Only: Administrative User Identities* on page 9.

This user identity must also have access to the **bin** directory where the AccuRev executables are stored.

READ ME NOW: Assuring the Integrity of the AccuRev Repository

The integrity of the AccuRev data repository is critically important. If information in the repository is lost or corrupted, your organization's ability to do business may be severely compromised. The integrity of the data repository relies on the integrity of underlying software (the file system, including the device drivers for data storage devices) and underlying hardware (the data storage devices themselves). Certain practices will enhance the safety and reliability of these underlying facilities. We strongly recommend the following:

- Use high-quality disk drives and disk controllers.
- Reduce the impact of a hard-disk failure by using disk mirroring (for example, using a RAID system) or other fault-tolerant disk subsystems.

- Power the AccuRev server machine with an uninterruptible power supply (UPS), with automatic shutdown of the server machine if the UPS is running out of power. This reduces the likelihood of interrupted data transfers to disk.
- Establish a good data-backup regimen, and make sure your backups are reliable by doing test restores on a regular basis. (See *Backing Up the Repository* on page 3.)

This section focuses on one aspect of data integrity: guaranteeing "write" operations to the repository. The AccuRev Server process does not, itself, perform the act of writing data on the disk. Like all application programs, it makes a "write" request to the operating system (Unix, Windows). In turn, the operating system performs a "write" operation to the disk itself. (On some larger systems, there may be additional links in this chain of write operations.)

Operating systems and disk subsystems often use special techniques that boost the performance of write operations, but can compromise data integrity. For example, when an application program makes a write request, the operating system might:

- Acknowledge the request immediately — good, because the application program can then proceed to its next operation.
- Delay actually sending the data to the disk ("write-behind") — bad, because a system failure at this point might result in the data never being stored on the disk.

It is essential that such techniques *not* be used when the AccuRev Server process sends information to the disk containing the AccuRev data repository. The Server always follows each write request with a "synchronize the disk" request. Sometimes, this ensures that data is safely on disk before the Server proceeds to its next task. For example, this is typically the case if the repository is stored on a disk that is local to the machine on which the Server is executing.

But in some situations delayed-write techniques may be used even when the AccuRev Server makes "synchronize the disk" requests. This is typically the case if the repository is located on a network shared file system. In such situations, the Server's "synchronize the disk" requests are effectively ignored, so that successful completion of write operations to the AccuRev repository cannot be guaranteed. (Some disk subsystems implement such a guarantee by having their own battery backup; buffered data is flushed to disk when the power fails.)

In an attempt to avoid such unsafe situations, the AccuRev Server process attempts to determine whether the file system where the repository is stored guarantees the successful completion of write operations. If it decides "no", the Server refuses to use the repository. This determination is not foolproof — both "false positives" and "false negatives" are possible.

There's a workaround in the "false negative" case — where the AccuRev Server process decides that the file system does not guarantee write operations, but *you* know that writes are, in fact, guaranteed. In this case, set environment variable AC_FS_WRITE_GUARANTEED to the value 1 in the environment in which the Server process runs; then restart the Server process.

If you have any question about the safety of your data-storage system, please contact AccuRev Support.

Backing Up the Repository

Note: before you start, consult *A Word of Caution on Windows Zip Utilities* below.

AccuRev supports live backup of the data repository: making copies of the data repository files while the AccuRev Server is running. The **backup** command takes just a few seconds to make checkpoint copies of certain **site_slice** files in a subdirectory named **backup**. It also records a "high water mark" file, **valid_sizes_backup**, in each depot directory, noting the depot's current transaction level. Transactions that are underway at the time the **backup** command executes are not included.

During **backup** command execution, clients can continue to work, but may notice a slight delay: transactions arriving at the AccuRev Server are queued for execution after completion of the **backup** command.

After executing the **backup** command, you can make a complete copy of the repository (the **storage** directory tree), without worrying about synchronization or time-skew. The append-only nature of AccuRev's databases makes this simple scheme possible. No matter when you make the backup copies, you'll be able to restore the repository to its state at the time you executed the **backup** command.

Note: the live-backup scheme relies on the ability to copy files that are currently open to the AccuRev Server process. Your backup utility must be able to copy files that are currently open at the operating system level. If you have any doubts or questions, contact AccuRev support.

Thus, the repository backup procedure is:

1. "Checkpoint" the database portion of the repository:

   ```
   accurev backup mark
   ```

2. If your backup utility cannot copy files that are currently open at the operating system level, stop the **accurev_server** program. (See *Controlling Server Operation* on page 12.)

3. Use standard operating system backup/restore tools to create a backup copy of the entire directory tree below the **storage** directory. This backup can be all-at-once or piecemeal; for example, you can back up the **site_slice** directory and the individual subdirectories within the **depots** directory with separate commands.

   Good candidates for backup/restore tools are **tar** (Unix), **xcopy /s** (Windows), and **zip** (both).

   Note: if your site slice is in a non-standard location (as specified by the SITE_SLICE_LOC setting in the **acserver.cnf** file — see *Server Configuration File* on page 10), or if some depots are in non-standard locations (perhaps moved with the **chslice** command), then your job in backing up the entire repository is more complicated than simply to copy the **storage** directory.

4. If you stopped the **accurev_server** program in Step 2, start it again. (See *Controlling Server Operation* on page 12.)

Restoring the Repository

If you have backed up the repository according to the directions above, you can easily restore the repository to the time at which you executed the **backup** command:

1. Stop the **accurev_server** program. (See *Controlling Server Operation* on page 12.)
2. Restore the backup copies of the **site_slice** and individual depot directories, using the standard backup/restore tools.
3. Go to the **site_slice** directory.
4. Overwrite database files with the checkpoint files in the **backup** subdirectory:

   ```
   cp backup/* .                 (Unix)

   copy backup\*.* .             (Windows)
   ```

5. Reindex the site slice:

   ```
   maintain reindex
   ```

6. Restore and reindex each depot:

   ```
   maintain restore <depot-name>
   maintain reindex <depot-name>
   ```

7. Restart the **accurev_server** program. (See *Controlling Server Operation* on page 12.)

   Note: suppose a particular depot's files were not backed up for several hours after the **backup** command was executed. During that interval, several new versions of file **gizmo.c** were created with the **keep** command. All of those versions will officially be lost when the repository is restored to its state at the time the **backup** command was executed. But you can still retrieve a copy of the last-created lost version of file **gizmo.c** from the backup medium.

Archiving Portions of the Repository

The container files that store the contents of individual file versions can now be move to offline storage, in order to save online storage space for the repository. For details, see *Archiving of Version Container Files* on page 19.

Moving a Workspace or Reference Tree

Note: before you start, consult *A Word of Caution on Windows Zip Utilities* below.

First, move the physical contents of the workspace tree or reference tree with standard operating system tools (e.g. **tar**, **zip**, **xcopy /s**). Then, let AccuRev know about the move:

```
accurev chws -w <workspace-name> -l <new-location>

accurev chref -r <reftree-name> -l <new-location>
```

Moving a Depot

Note: before you start, consult *A Word of Caution on Windows Zip Utilities* below.

First, move the physical contents of the depot with standard operating system tools (e.g. **tar**, **zip**, **xcopy /s**). Then, let AccuRev know about the move with this command:

```
accurev chslice -s <slice-number> -l <new-location>
```

(Use **accurev show depots** to determine the slice number of the depot.)

Removing a Depot

A depot can be removed completely from the repository with the **maintain rmdepot** command. This operation is irreversible! For details, see *Removing a Depot from the AccuRev Repository* on page 50.

A Word of Caution on Windows Zip Utilities

Be careful when using WinZip® or PKZIP® on a Windows machine to perform the tasks described above: backup/restore of the entire repository, or moving a workspace, reference tree, or depot. You may want to use **tar** on a Unix machine to "pack up" a directory tree, and then use the Zip utility on a Windows machine to "unpack" it.

- When moving the entire repository or an individual depot, be sure to disable conversion of line-terminators during the "unpack" step:
  - In WinZip, make sure the option "TAR file smart CR/LF conversion" is not selected (**Options > Configuration > Miscellaneous**).
  - In PKZIP, make sure the "CR/LF conversion" setting is "None -- No conversion" (**Options > Extract**).

  Enabling conversion of line-terminators during the "unpack" step will corrupt the text files in a depot's file storage area (see *File Storage Area* below). The AccuRev Server always expects lines in these text files to be terminated with a single LF character, no matter what kind of machine the server is running on.
- Conversely, when moving a workspace or reference tree, you may wish to enable "TAR file smart CR/LF conversion". The files in a workspace or reference tree are handled directly by text-editors, compilers, testing tools, etc. Many Windows text-editors are incapable of handling text files whose lines are terminated with a single LF character.

Storage Layout

Each AccuRev depot is stored in a separate directory tree under the installation area's **storage** directory. The **storage** directory is a sibling of the executables ("bin") directory. For example, if

AccuRev is installed at **/usr/accurev** and depots named **moe**, **larry**, and **curly** are created, the directory layout would be:

```
/usr/accurev
    bin
    storage
        site_slice
        depots
            moe
            larry
            curly
```

A depot consists of three parts:

Configuration Files

The **mktrig** command creates a one-line configuration file that names the script to be executed when the trigger fires for transactions involving this particular depot. For example, making a trigger of type "pre-keep-trig" creates a configuration file in the depot named **pre-keep-trig**. (This file might contain the pathname **/usr/local/bin/accurev_prekeep.pl**.)

Metadata Area

The metadata area stores information about versions, times, and source file storage locations within the source file storage area of the depot. The metadata is stored in files ending with **.ndb** and **.ndx**.

The metadata area must be physically located on the machine where **accurev_server** is running. This guarantees the integrity of physical disk writes. Moving the metadata area to a remote file system compromises data integrity and is not supported by AccuRev, Inc.

File Storage Area

Whenever a user creates a new real version of a file with the **keep** command, the AccuRev Server copies the file from the user's workspace to the depot's file storage area. The newly created storage file is permanently associated with the real version-ID in the workspace stream (e.g. 25/13), and also with subsequently created virtual version-IDs in higher-level streams (7/4, 3/9, 1/4).

Storage files are located in subdirectory tree **data** within the depot directory. The files may be in compressed or uncompressed form. Compressed files may correspond to more than one real version. Conceptually, storage files are numbered sequentially starting with 1 and going up to 2**64. (That should be enough.) Within the **data** directory, they're arranged in a hierarchy for faster access. For example, storage file #123456 would be stored as **data/12/34/56.sto**. Recovery information for the storage file is stored in a like-named **.rrf** file (e.g. **data/12/34/56.rrf**).

You can relocate a depot's file storage area onto other disk partitions or even onto remote disks. The cautions about storing data locally do not apply to files in the data directories. However, exercise extreme caution when relocating storage in this area. Make sure you have first done a full backup and have shut down the **accurev_server** program.

The AccuRev Server

The AccuRev data repository is managed by a single program, the AccuRev Server (**accurev_server**). This program must be started prior to running any AccuRev client commands. The server program should be the only process that directly manipulates the AccuRev repository. No person should attempt to work directly with the repository, unless it is an emergency.

User Identity of the Server Process

The AccuRev Server process, named **accurev_server**, has a user identity at the operating system level. This process should run with a special user identity, to help ensure that no other user or process modifies the data repository:

- Unix: create a user named **acserver**.

  You may be tempted to simply let the AccuRev Server process run as **root**. But we strongly recommend against this, as it would open a large security hole. The Server can run user-supplied trigger scripts (see *AccuRev Triggers* on page 40). In general, having user-supplied scripts run as the **root** user is very dangerous!

- Windows: don't create a new user; the AccuRev Server process runs as the built-in local user named **System**.

This user identity must have access to the AccuRev executables (**bin**) directory and to the data repository (see *Repository Access Permissions* on page 1).

> Note: **acserver** and **System** are user accounts at the operating system level. You don't need to — and should not — create a principal-name (AccuRev user name) "acserver" or "System". On the other hand, the AccuRev Server might need to take on an AccuRev user identity when it executes a server-side trigger script. For more on this topic, see *Trigger Script Execution and User Identities* on page 45.

Unix Systems Only: Administrative User Identities

You can control the user identity under which the Server runs with the server configuration file. See *Server Configuration File* on page 10.

Not even Unix administrators should run as **acserver** unless it is an emergency. If you want to have an AccuRev administrator account, we recommend creating a separate account named **acadmin**. Place both **acserver** and **acadmin** in a group named **acgroup**, and record these names in the server configuration file:

```
USER = acserver
GROUP = acgroup
```

With this setup, the **acadmin** user will be able to access the repository. You can configure Unix-level auditing and place other appropriate controls on this account; this leaves the **acserver** account (and thus, the AccuRev Server process) unencumbered by such controls.

Starting the AccuRev Server

The following sections describe how to start the AccuRev Server program, either automatically at operating system bootstrap time, or manually at a command prompt. (You can also perform a "manual" startup with a Unix shell script or a Windows batch file.)

Running the Server Automatically at Operating System Startup

Typically, the Server program is started automatically when the operating system boots on the server machine. On Unix systems, an "rc" or "init.d" startup script starts the **accurev_server** program. The AccuRev installation program does not install the startup script automatically, however. You must customize and install the sample startup script located in the **extras/unix_init** subdirectory of the AccuRev installation directory. See the **README** file for complete instructions.

On Windows NT/2000/XP systems, the AccuRev installation program automatically configures the **accurev_server.exe** program as a Windows service. Use the standard **Services** applet on the Windows Control Panel to control the Server program.

Starting the Server Manually

The AccuRev Server must be started manually in the following environments:

On Windows 95/98/ME systems

Start the Server with the **AccuRev Demo Server** shortcut on the desktop. Alternatively, run the **server_start.bat** script, located in the AccuRev executables (**bin**) directory.

On Windows NT/2000/XP systems

If you've changed the startup type of the AccuRev service to "Manual", you can start the service from the **Services** applet. Alternatively, run the **server_start.bat** script, located in the AccuRev executables (**bin**) directory.

On Unix systems

Start the Server with the **acserverctl** utility:

```
<AccuRev-executables-dir>/acserverctl start
```

Server Configuration File

When it starts, the Server program reads configuration file **acserver.cnf**, located in the AccuRev executables directory. This configuration file is generated during installation, but can be edited manually thereafter.

Here is a sample **acserver.cnf**:

```
MASTER_SERVER = accurev_server_machine.company.com
SITE_SLICE_LOC = /partition0/site_slice
```

IMPORTANT NOTE: the white space surrounding the equals sign (=) in configuration files is mandatory.

The name of the server machine should be the fully-qualified server name, including a domain name and Internet extension. Using just the server name may work in most situations, but fully-qualified is preferred. Alternatively, you can use the IP address of the server machine.

The SITE_SLICE_LOC setting points to the directory that the server uses for storing information about the site such as the user registry, depot information, etc. This directory should be owned by the **acserver** account (Unix) or the System account (Windows) and must be physically located on the server machine. The SITE_SLICE_LOC location must not be within a remotely mounted file system (Unix) or within a shared directory (Windows) on a remote machine.

Unix Systems Only: Controlling the Server's User Identity

Note: this section applies to Unix machines only. On Windows machines, the user identity of the AccuRev Server is always the local **System** account.

If you perform a server installation as the **root** user, the following specifications are stored in the **acserver.cnf** file:

```
USER = nobody
GROUP = nobody
```

When the Server starts automatically at system bootstrap time, it reads these specifications and assumes the identity: user **nobody** in group **nobody**.

You can edit these settings to change the AccuRev Server's user/group identity. This change takes effect the next time the Server is started by the **root** user — either automatically at system bootstrap time or manually, using the **acserverctl** utility. (See *Controlling Server Operation* below.)

When the Server is started manually, it runs under the identity of the user who started it.

Server Watchdog

The AccuRev Server is designed for high reliability, to ensure the integrity of the repository and its availability to AccuRev users. But even the robust software systems are occasionally compromised; the AccuRev Server can be brought down by a bad disk sector or an administrator's mistaken command.

Starting with Version 3.0.3, the reliability of the AccuRev Server has been increased even further: a companion program to the Server, termed the Watchdog, runs on the same machine. The sole function of the Watchdog is to monitor the Server and restart it in the event of a failure. The effect of the Watchdog on Server performance is insignificant.

Note: both the Server and Watchdog show up in the operating system's process table with the same name: **accurev_server**.

For the most part, the Watchdog is "transparent", making administration simple:

- The Watchdog process starts automatically when the Server process is started (typically, at operating system bootstrap time).
- The administrative commands for stopping the Server process cause both the Watchdog and Server to stop. These commands have been reworked to terminate the Watchdog directly; before it exits, the Watchdog terminates the Server.

Starting with AccuRev Version 3.0.3, there are new tools for controlling the execution of the Server — and now, the Watchdog, too. See *Controlling Server Operation* below.

Server Logging

The AccuRev Server maintains a log file in subdirectory **logs** of the **site_slice** directory. As of Version 3.0.3:

- The name of the log file is **acserver.log**. (In previous releases, the log file was named **accurev_server.log** and was located in the **site_slice** directory itself.)
- Each log message includes a timestamp, a client-server connection ID, the user's principal-name, the name of the server subtask being performed, the ID of the thread performing the server subtask, and the IP address of the client machine. For example:

  ```
  2003/05/09 12:30:36   1002 jjp cur_wspace    0x803 127.0.0.1
  ```

- Log file rotation" keeps the log file from growing too large. Periodically, the AccuRev Server timestamps the current log file and moves it to subdirectory **logs** of the **site_slice** directory. For example, the log file might be renamed **acserver-2002-01-23-04-47-29.log**. The Server then creates a new **acserver.log** file.

  The log file is rotated weekly; it is also rotated whenever the AccuRev Server is restarted.

Watchdog Logging

The Watchdog also maintains a simple log file, **acwatchdog.log**, in the **logs** directory. The Watchdog log file is rotated in the same way as the Server log file.

Controlling Server Operation

Starting with Version 3.0.3, AccuRev includes new facilities for controlling the operation of the AccuRev Server and the new Watchdog. The user interface varies by platform:

- Unix: a special command-line utility
- Windows: the standard **Services** console

Unix: 'acserverctl' Utility

If the AccuRev Server is running on a Unix machine, you can control its operation with the **acserverctl** program. This is a Bourne-shell script, located in the AccuRev **bin** directory. (It is based on the control script for the Apache Web server.)

Note: as of Version 3.2, **acserverctl** assumes, by default, that AccuRev is installed at **/opt/accurev**. If this is not the case, you must run **acserverctl** in an environment where ACCUREV_BIN is set to the pathname of the AccuRev **bin** directory. For example:

```
env ACCUREV_BIN=/var/accurev/bin acserverctl ...
```

acserverctl provides a set of non-interactive commands. The format of each command is:

```
acserverctl <command-name>
```

(Omitting <*command-name*> is equivalent to executing **acserverctl help**.) The commands are:

start

Start the Server and Watchdog processes.

stop

Tell the Server and Watchdog processes to stop gracefully.

status

Report whether the Server is running or not.

pause

Tell the Server to stop accepting new requests from AccuRev clients.

resume

Tell the Server to start accepting new requests from AccuRev clients again.

restart

Tell the Server process to stop gracefully; this allows the Watchdog to restart it. If the Watchdog is not running, a **start** or **hardrestart** is performed.

kill

Forcibly stop the Server and Watchdog processes. This is accomplished by sending a TERM signal to each process. The script gets the process-IDs from files **acserver.pid** and **acwatchdog.pid**, located in the **site_slice** directory. These files are written automatically when the processes are started.

hardrestart

Perform a **kill**, followed by a **start**.

help

Display an **acserverctl** command summary.

The various “tell a process” capabilities are implemented through server-control files. (See *Server-Control Files* below.)

Windows: 'Services' Console

If the AccuRev Server is running on a Windows machine, you can control its operation from the standard Windows **Services** console. The **Services** console is located in the Windows **Control Panel**; in some versions of Windows, it's in a subfolder called **Administrative Tools**.

The context (right-click) menu of the AccuRev service includes these commands:

start
stop
pause
resume
restart

For descriptions of these commands, see *Unix: 'acserverctl' Utility* above. On Windows, the **restart** command brings down both the Server and the Watchdog, by performing a **stop** followed by a **start**.

Server-Control Files

On all platforms, the AccuRev Server and Watchdog processes check, once per second, for the existence of several "server-control files" in the **site_slice** directory. The existence of the server-control file causes the process to perform a particular action. In most cases, the contents of the file are irrelevant; a zero-length file will do.

acserver.command.pause

(used by the **pause** server-control command) Tells the Server to stop accepting new requests from AccuRev clients. The Server completes transactions that are already in progress and logs its "paused" status to the log file. Then, it continues to run, but the only thing it does is monitor the **acserver.command.pause** file. When this server-control file is removed, the Server resumes normal operation.

This server-control file is *not* removed when a new Server process starts up. If the file exists, the Server starts up in the paused state.

acserver.command.shutdown

(used by the **stop** and **restart** server-control commands) Tells the Server to "finish up quickly" and exit. The Server immediately stops accepting new requests from AccuRev clients. It continues to work on transactions that are already in progress, but it aborts any transactions that are not completed within 10 seconds. Then, the Server exits.

If the Watchdog is running, it detects the Server's shutdown and starts up a new Server immediately. Thus, this server-control file typically causes a Server restart. In any event, this file is automatically removed whenever a new Server process starts up.

If 10 seconds is not the appropriate interval for "finishing up quickly", place another integer (such as 120) in the **acserver.command.shutdown** file. The Server exits when there are no more transactions to work on, or when the timeout interval has passed, whichever comes first.

acwatchdog.command.shutdown

(used by the **stop** server-control command) Tells the Watchdog to exit cleanly. When the Server detects that the Watchdog has exited, it exits also, just as if it had found an **acserver.command.shutdown** file (see above). In this case, however, there is no longer a Watchdog process, so no restart of the Server takes place.

This server-control file is automatically removed when a new Server process starts up.

Open Filehandle Limits and the AccuRev Server

The AccuRev Server is designed to handle multiple client commands concurrently: any number of requests that "read" data, along with one command that "writes" data. Accomplishing such concurrency typically requires that the AccuRev Server have many files open at the same time. Each operating system imposes limits on how many files can be open simultaneously. There may be an "open file descriptor" limit for each user process, or an overall limit for all user processes, or both. If the AccuRev Server hits the open file descriptor limit, additional client requests will be queued until file descriptors become available. (No client command is cancelled, and no data is lost. Hitting the open file descriptor limit just slows AccuRev Server performance.)

The table below indicates the default open file descriptor limits for the various AccuRev-supported operating systems. Following the table are instructions for increasing these limits.

Operating System	Default Limit on Open Filehandles	Command to Change Limit on Open Filehandles
Windows NT, Windows 2000, Windows XP Pro	2048 per system	none
Windows 98, Windows ME, Windows XP Home	2048 per system	none
HP-UX 11	360 per process	no command; must rebuild Unix kernel (see below)
Solaris	64 per process	no command; must reconfigure Unix kernel (see below)
Red Hat Linux 6.2	about 3500 per system	/sbin/sysctl –w fs.file-max=10000
Red Hat Linux 7.2	about 28,000 per system	/sbin/sysctl –w fs.file-max=30000
Power PC Linux 2.4	1024 per process	
FreeBSD 4.2	1064 per process	/sbin/sysctl –w kern.maxfiles=30000 /sbin/sysctl –w kern.maxfilesperproc=10000
FreeBSD 4.5	about 3600 per system	/sbin/sysctl –w kern.maxfiles=30000 /sbin/sysctl –w kern.maxfilesperproc=10000
NetBSD 1.5.2	1024 per process	/sbin/sysctl –w kern.maxfiles=30000 /sbin/sysctl –w kern.maxfilesperproc=10000

Operating System	Default Limit on Open Filehandles	Command to Change Limit on Open Filehandles
AIX	2000 per process	
IRIX 6.2	1024 per process	
Tru64	4096 per process	

Note: If you are performing a pre-purchase evaluation of AccuRev in an environment with a limited number of users and a limited amount of data, there is no need to make any changes. The default limits will be more than adequate.

Changing the Per-Process Open File Descriptor Limit

The procedure for increasing a process's maximum number of open files varies from operating system to operating system.

Note: in all cases, be sure to remove file **acserver.handle.limit**, located in the AccuRev **site_slice** directory, before restarting the AccuRev Server or rebooting the operating system. This file caches the current value of the open-files limit.

Linux

You must be the **root** user to perform the following procedure.

1. Change the overall limit on the number of open file descriptors each process can have (e.g. to 10,000):

   ```
   > /sbin/sysctl -w fs.file-max=10000
   ```

   The number you specify is stored in file **/proc/sys/fs/file-max**.

2. Add this line to file **/etc/pam.d/login**:

   ```
   session   required   /lib/security/pam_limits.so
   ```

3. Change the capabilities of the Bash shell command **ulimit**, by creating or editing the "nofile" (number of open files) lines in file **/etc/security/limits.conf**. Example:

   ```
   *    soft    nofile    1024
   *    hard    nofile    10000
   ```

   These lines specify that:

   - By default, a Bash shell (and its subprocesses) can have as many as 1024 open file descriptors.
   - A Bash shell can execute the command **ulimit –n *number***, with 65535 as the maximum value of ***number***. This enables that particular shell (and its subprocesses) to have up to ***number*** open files. (The person executing the **ulimit** command doesn't need to know what

the "hard limit" specified in **/etc/security/limits.conf** is — he can just enter the command as **ulimit –n unlimited** to get the maximum value.)

4. Test your change, by entering a **ulimit** command in a Bash shell, setting the limit somewhere in between the **soft** and **hard** specifications you made in Step 3. For example:

```
> /bin/bash

$ ulimit -n
1024

$ ulimit -n 5000

$ ulimit -n
5000
```

5. Restart the AccuRev Server process from a Bash shell:

```
> <AccuRev-bin-directory>/acserverctl stop     (stop the AccuRev Server)

> /bin/bash                                    (start a Bash shell)

$ <AccuRev-bin-directory>/acserverctl start    (restart the AccuRev Server)
```

Solaris

You must be the **root** user to perform the following procedure.

1. Change the overall limit on the number of open file descriptors each process can have (e.g. to 5,000), by adding or changing this line in file **/etc/system**:

```
set rlim_fd_max=5000
```

2. Reboot the operating system.

HP-UX

You must be the **root** user to perform the following procedure.

1. Enter this command to start the System Administration Manager utility:

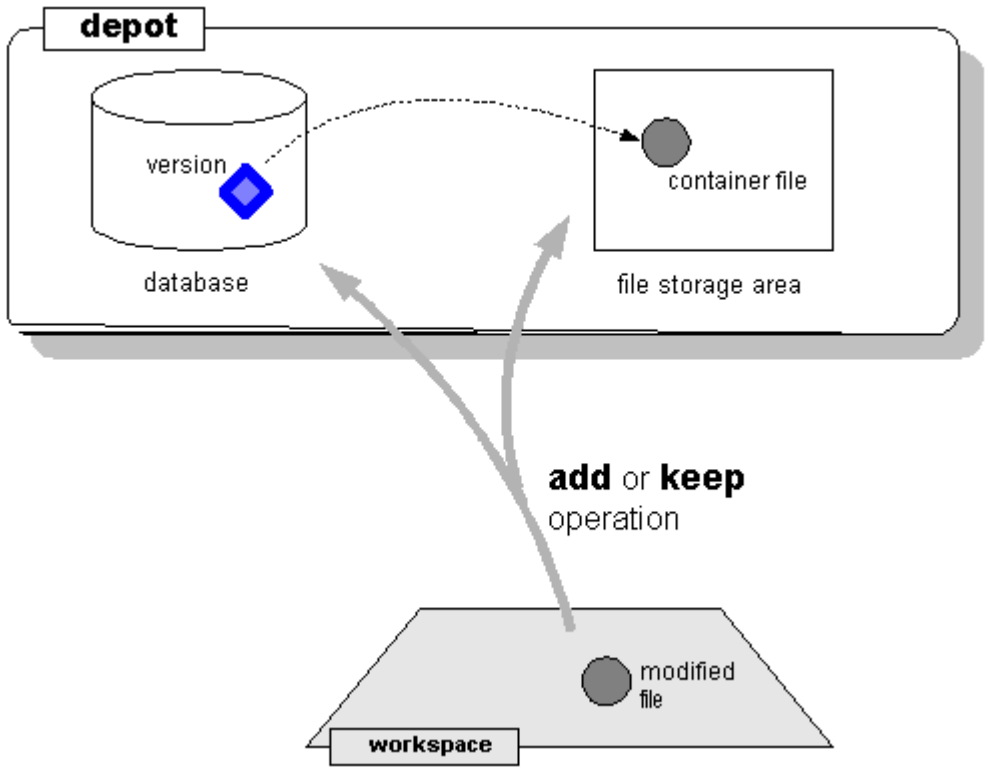
```
> sam
```

2. Select **Kernel Configuration**, then **Configurable Parameters**.
3. Select the **maxusers** parameter.
4. Increase the value of this parameter, for example to 128.
5. Invoke the **Actions > Process New Kernel** command, to create a new HP-UX kernel.
6. Exit the System Administration Manager utility.
7. Reboot the operating system.

Archiving of Version Container Files

Users execute a **keep** command to preserve the current contents of a version-controlled file ("file element") in an AccuRev depot. Similarly, users execute an **add** command to place a file under version control. The **add** and **keep** commands:

- copy the current contents of the file to a container file, located in the depot's file storage area.
- create an associated version object in the depot's database.



In accordance with the TimeSafe principle, the version object can never be deleted from the database or modified in any way. The corresponding container file is always accounted for, and can be in either of these states:

- **normal** — the container file is located in the depot's file storage area (the **data** subdirectory of the depot directory). AccuRev commands, such as **update**, **cat**, and **diff**, can access the contents of the version.
- **archived** — the container file has been moved to a gateway area outside the depot's file storage area. AccuRev commands cannot access the contents of an archived version. After container files have been moved to the gateway area, an administrator can use standard operating system or third-party tools to transfer the container files to off-line storage: tape, CD-ROM, removable disk drive, Web-accessible storage, etc.

Note: as of Version 3.5.5, only binary files can be in the archived state. Text files are always in the normal state.

The AccuRev CLI commands **archive** and **unarchive** shift container files back and forth between the normal and archived states. Before using **unarchive**, the administrator would transfer the appropriate container files from off-line storage back to the gateway area. Then, invoking **unarchive** moves the container files back into the depot's **data** directory.

The 'archive' Command

The command **accurev archive** processes one or more versions of file elements, shifting the versions' container files from **normal** status to **archived** status. The command has this format:

```
accurev archive [ -i ] [ -s <stream> ] [ -t <transaction-range> ]
   [ -c <comment> ] [ -R ] [ -E <elem-type> ] <element-list>
```

Determining Which Versions to Archive

archive determines the set of versions to archive as follows:

- It focuses on a particular stream. If you don't specify a stream with **–s <*stream*>**, it uses your current workspace's stream.
- It narrows the scope to a particular set of file elements, which you specify as command-line arguments in the ***<element-list>***. You can include directories in this list; in this case, use the **–R** option to include the recursive contents of those directories.
- By default, all eligible versions of the specified elements in the specified stream are archived. You can use **–t** to limit the set to the versions of those elements created in a specified transaction, or range of transactions:

  –t <*number*> *single transaction*
  –t <*number*>–<*number*> *range of transactions*

- You can also limit the set of versions to those of a particular element type, using the **–E** option.

  Note: as of Version 3.5.5, you *must* specify the **binary** element type: **–E binary**.

The **archive** command refuses to archive any version that is currently visible in any stream or snapshot.

Note: as of Version 3.5.5, **archive** also refuses to process versions in text format. Only binary-format versions will be archived.

Using the **–i** option (in addition to the other options described above) generates an XML-format listing of the desired versions, but does not perform any actual archiving work. It is highly recommended that you do this before actually archiving any versions.

Archiving the Versions

After determining which versions to process, the **archive** command moves a version's container file from a "normal" location under the **data** directory:

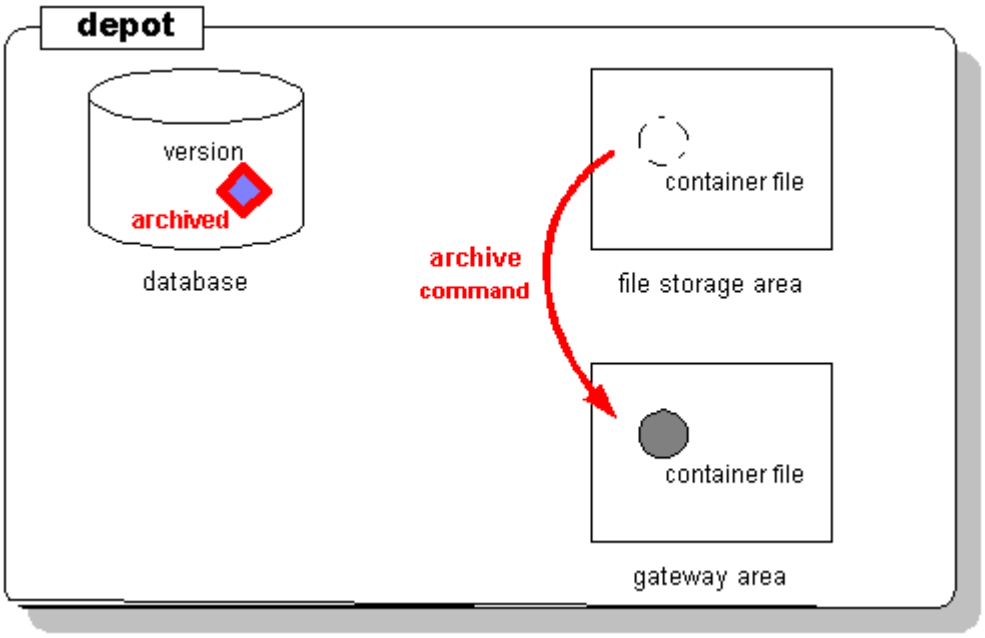
```
.../storage/depots/gizmo/data/25/07.sto
```

... to a corresponding “archived” location in the **archive_gateway/out** area:

```
.../storage/depots/gizmo/archive_gateway/out/data/25/07.sto
```

archive also marks the version as “archived” in the depot database.

Subsequent attempts by AccuRev commands to retrieve the contents of the archived version will fail.



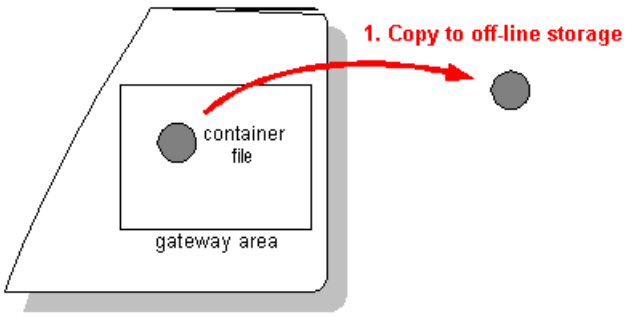
The changes made by this command are recorded in the depot database as a transaction of type **archive**. You can use the **–c** option to specify a comment string to be stored in this transaction. You can search for particular comment strings when using the **hist** command to locate previous **archive** transactions. See *Using ‘hist’ to Research Previous ‘archive’ Commands* on page 22.

The ‘reclaim’ Command

The **archive** command merely moves container files from one location (the depot’s **data** area) to another location (the depot’s **archive_gateway** area). To reduce the amount of disk space consumed by the archived versions, you must:

1. Copy the files from the **archive_gateway** directory tree to off-line storage. You can use operating system commands (**copy**, **xcopy**, **cp**, **tar**) and/or third-party data-backup utilities to accomplish this.

   Be sure to use a tool that preserves the source data’s directory hierarchy in the copied data.

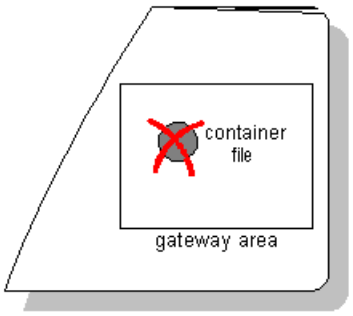


WARNING! AccuRev has no way of tracking which tool you use for this purpose, or what off-line storage medium you copy the files to. It's up to you to maintain good records of these activities!

2. Delete the files from the **archive_gateway** directory tree, using the **reclaim** command:

   ```
   accurev reclaim [ -p <depot> ]
      -t <archive-transaction>
   ```

   You must specify a single transaction of type **archive**, created by previous **archive** command(s).



2. Reclaim disk space

Attempts to Access Archived Versions

The **archive** command affects depot storage only. It has no immediate effect on any workspace. But you might subsequently enter an AccuRev command that attempts to access a version that has been archived. For example, if version **gizmo_int/8** of file **floor_layout.gif** has been archived, then this command fails:

```
accurev cat -v gizmo_int/8 floor_layout.gif > old_layout.gif
```

In such cases, a message is sent to **stderr** and the command's exit status is 1.

Using 'hist' to Research Previous 'archive' Commands

Each depot's database contains a complete record of all version-archiving activity for that depot. Execution of the **archive** command is recorded as a transaction of kind **archive**. You can use the **hist** command to locate all such transactions:

```
accurev hist -a -k archive
```

You can also select just those **archive** transactions that were created with a particular comment string:

```
accurev hist -a -k archive -c "stadium images"
```

In a **reclaim** command, you must indicate the storage space to be reclaimed by specifying the number of an **archive** transaction.

Restoring Archived Versions — The 'unarchive' Command

After you have **archive**d some versions and **reclaim**ed the disk space:

- the versions' container files are no longer in the depot's **data** area.
- copies of the container files are no longer in the depot's **archive_gateway/out** area (since you've transferred them to off-line storage).

If you decide you need to restore some or all of the archived versions, you must first copy the container files from off-line storage back to the **archive_gateway** area. You must place the files under **archive_gateway/in**, at the same relative pathname as they were originally placed under **archive_gateway/out**. For example, if the **archive** command places a container file at:

```
.../storage/depots/gizmo/archive_gateway/out/data/25/07.sto
```

... you must restore the file from off-line storage to this pathname:

```
.../storage/depots/gizmo/archive_gateway/in/data/25/07.sto
```

After placing all the container files in the **archive_gateway/in** area, you can execute the **unarchive** command. This command has exactly the same format as **archive** — that is, you specify the versions to be restored in exactly the same way as you originally archived them. For example:

Archive all non-active versions of GIF image files in stream **gizmo_maint_4.3**:

```
accurev archive -s gizmo_maint_4.3 -E binary *.gif
```

Restore all those versions:

```
accurev unarchive -s gizmo_maint_4.3 *.gif
```

Replication of the AccuRev Repository

This chapter describes how to set up and use AccuRev's repository replication feature, introduced in Version 3.7. One server machine stores the "master" copy of the AccuRev data repository; any number of additional server machines can store "replicas" of the repository. Users can send commands to the AccuRev Server software running on any of these machines.

> Note: use of the repository replication feature requires purchase of the *AccuRev Replication Server* product from AccuRev, Inc.

Master and Replica

As described in the *AccuRev Administrator's Guide*, one host in the network acts as the AccuRev server machine: it runs the AccuRev Server process and contains the AccuRev repository in its local disk storage. In a replication scenario, this original host is termed the master server.

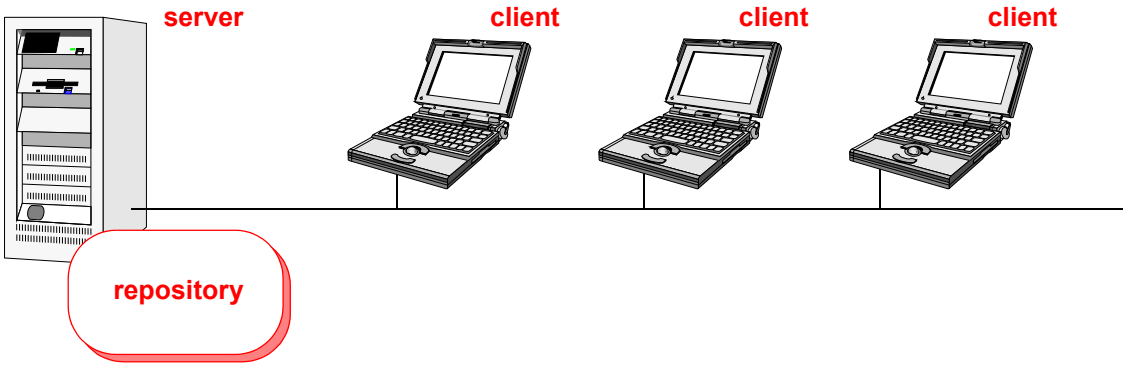
One or more additional hosts in the network can act as replica servers. Each such host runs its own instance of the AccuRev Server process; likewise, each such host has its own copy of the AccuRev repository. The diagram below shows the servers in a replication scenario, along with various client machines.

We use the terms master repository and replica repository to distinguish the multiple repositories in a replication scenario. The master repository is always complete and up-to-date; all transactions (operations that change the repository) are handled by the master server and are logged in the master repository.

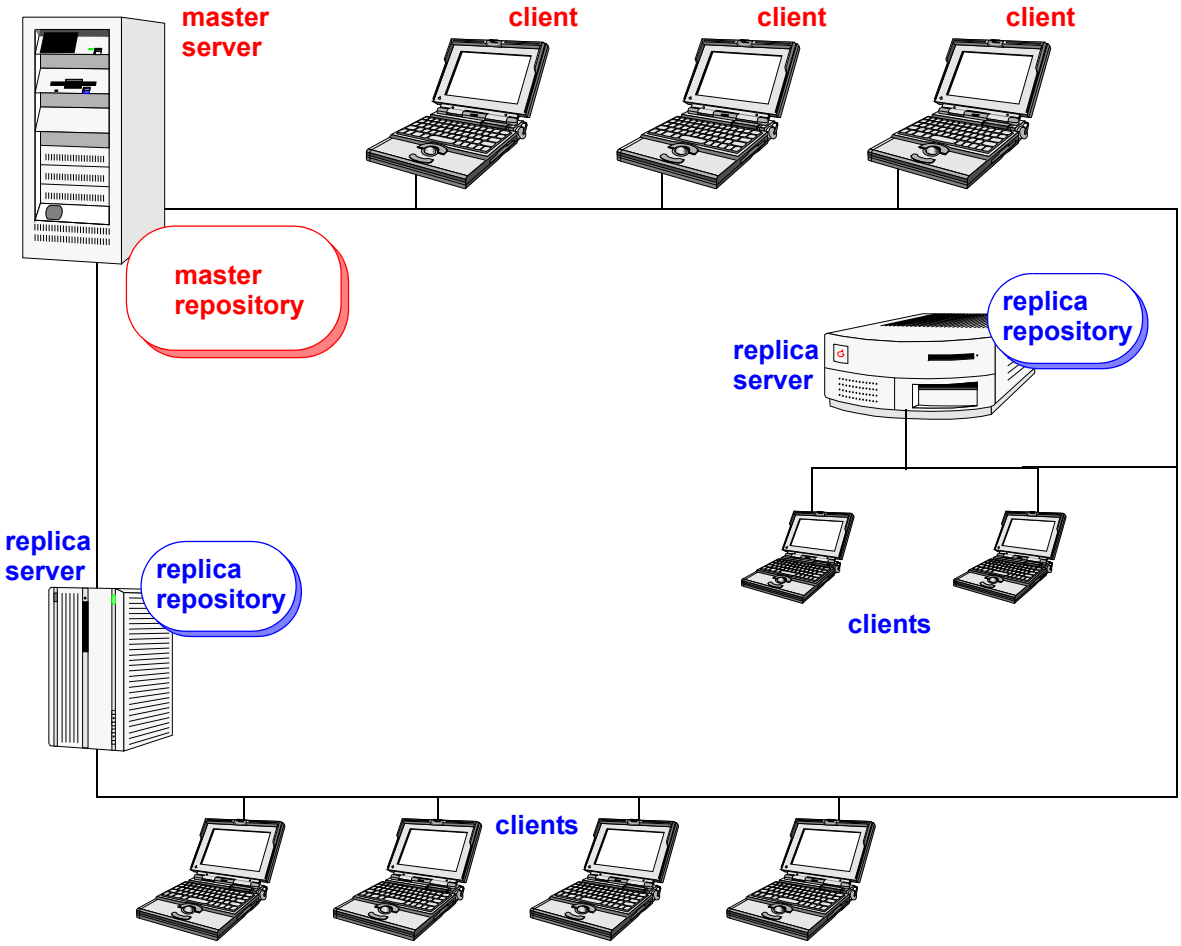
By contrast, a replica repository can become out of date during day-to-day usage: it can be missing recent transactions initiated by clients using other replica servers or the master server. You can issue a simple synchronization command to download such missing transactions from the master repository to the replica repository. This makes the replica repository into an exact copy (temporarily, at least) of the master repository. Synchronization also occurs automatically whenever a transaction is initiated by a client using that replica server.

For more details on day-to-day operations involving master and replica repositories, see the sections starting with *Using a Replica Server* on page 31. First, we address licensing issues and describe the replica setup process.

Before Replication



After Replication



AccuRev Licensing in a Replication Environment

A standard AccuRev license (Professional Edition or Enterprise Edition) enables your organization to maintain a single data repository, managed by a single AccuRev Server process running on a particular machine. This license also enables a certain number of users to access the repository with AccuRev client software.

The most obvious (and easiest) way to add replication to this environment is to have the existing server machine become the master server. Let's say that you want to replicate the repository at two remote sites — one with 10 client users, the other with 25 client users. In this case, your organization needs to purchase:

- two copies of AccuRev Replication Sever
- 10 individual Remote User Licenses
- one Remote Site License (for the 25 users at the second remote site)

Installation Procedure: Assumptions

The procedures in the following sections make the following assumptions about your AccuRev installation:

- The original AccuRev server machine is named **masthost**. If one or more replicas have already been created, then this machine is already the master server.
- The machine to be made into a replica server is named **replhost**.

Restriction: **masthost** and **replhost** must both have "big-endian" hardware architectures, or must both be "little-endian". Little-endian architectures include Intel and AMD, running either Windows or Linux software. Big-endian architectures include Sparc (Sun) and PA-RISC (Hewlett-Packard), and PowerPC (AIX)

Making a Copy of the Master Repository

Establishing a replica is essentially a data-copy operation: you must copy the AccuRev repository data from **masthost** to **replhost**. Most likely, your organization already has a repository-backup procedure in place; if so, you might be able to use this procedure to copy the data. Here's the general procedure:

1. Determine the location of the data storage within the AccuRev repository on **masthost**. Typically, all the data is in one place: under subdirectory **storage** of the AccuRev installation directory. But it's possible that the site slice and depot storage locations have been separated. It's also possible that individual depots are stored outside the default depot storage area. So:
    - Look in configuration file **acserver.cnf**, in the AccuRev **bin** directory on **masthost**. The SITE_SLICE_LOC and DEPOTS_DEFAULT settings indicate where the site slice is stored and where all the depots *might* be stored.

- Use the command **accurev show slices** to display the actual locations of the depot storage directories ("slices"). Note which depot slices, if any, are not stored under the DEPOTS_DEFAULT directory.

2. Stop the AccuRev Server process on **masthost**.
3. Copy all the repository data — including the site slice directory tree and all the depot directory trees — to a data-transfer medium. A CD-ROM or DVD may be an appropriate choice here. Be sure that the new host, **replhost**, will be able to accept the data-transfer medium.
4. Make a copy of **masthost**'s server configuration file: **acserver.cnf** in the AccuRev **bin** directory. This is a small text file, so the copy might easily take the form of an email attachment.
5. Edit the original **acserver.cnf** file (not the copy), adding the following line:

   ```
   REPLICATION_ENABLED = true
   ```

   CAUTION: enabling replication poses a potential security risk. Before proceeding, be sure to read *Synchronization Security* on page 33.
6. Restart the AccuRev Server process on **masthost**.

Procedure for Setting Up the Replica Server

The following sections detail the steps for setting up **replhost** as an AccuRev replica server, using the repository data from **masthost**.

Install AccuRev on the Replica Server

7. Obtain from AccuRev Customer Support a **keys.txt** file containing a license to run the AccuRev Server software on **replhost**. Most specifications, including the number of users, should match the license on **masthost**. The host name will differ — and the port number might differ, too.
8. Install the AccuRev software on **replhost**. During installation, choose both the Custom and Full options, so that you explicitly specify the storage location for the repository ("Customize: Choose a Folder for AccuRev Server Data Storage").
9. Be sure the storage location you choose has enough room for the replica repository to grow as development proceeds. Remember that the replica repository periodically becomes an exact copy of the master replica — containing *all* development changes, not just the changes handled by the local replica server. In the following steps, we'll assume that the repository location is **/opt/accurev/storage** — adjust for your installation and operating system.

   For the license key file, specify a non-existent pathname. The installation wizard displays an "Unable to find the specified license key file" error message, but allows you to continue. In Step 12 below, you'll install the **keys.txt** file that you obtained in Step 7.

   At the conclusion of the installation procedure, choose *not* to start the AccuRev Server software.

Copy the Repository Data / Install the License Key File

10. Make sure the AccuRev Server on **replhost** is not running.

11. Copy the data from the data-transfer medium to disk storage on host **replhost**. The details depend on how the data is organized on the data-transfer medium, and how you want it stored on **replhost**. The most important thing is to overwrite the new site slice directory (**/opt/accurev/storage/site_slice**) with the copy of the site slice directory on the data-transfer medium. In the simplest case, you can simply overwrite the new **/opt/accurev/storage** directory with the **storage** directory on the data-transfer medium.

    Restriction: The site slice must be placed on disk storage that is local (that is, not networked) to **replhost**. Depot slices can be placed on networked data storage.

12. Copy the **keys.txt** file you obtained in Step 7 to the site slice directory (**/opt/accurev/storage/site_slice**).

Revise the Server Configuration File

13. Examine the server configuration file, **acserver.cnf**, in the AccuRev **bin** directory on **replhost**, taking note of the MASTER_SERVER and PORT settings. These settings were established during AccuRev installation (Step 8).

14. Move this file aside — for example, renaming it to **acserver.cnf.orig**.

15. Create a new **acserver.cnf**, using the copy of **masthost**'s server configuration file you created in Step 4. For example, the contents of this file might be:

```
MASTER_SERVER = masthost
PORT = 5050
SITE_SLICE_LOC = C:\Program Files\accurev\storage\site_slice
DEPOTS_DEFAULT = C:\Program Files\accurev\storage\depots
LOG_LEVEL = 2
```

16. Do not change the MASTER_SERVER or PORT settings. These settings enable AccuRev software on **replhost** to communicate with the AccuRev Server on **masthost**, the master server.

```
MASTER_SERVER = masthost
PORT = 5050
```

17. Change the SITE_SLICE_LOC setting to accord with where you placed the copy of the site slice in Step 11:

```
SITE_SLICE_LOC = /opt/accurev/storage/site_slice
```

    Note: The location of the site slice need not be the same on **replhost** as on **masthost**. And if you implement replication on a single machine (that is, **masthost** and **replhost** are the same machine), the two site-slice locations *must* be different.

18. Change the DEPOTS_DEFAULT setting to an appropriate pathname on **replhost**. For example:

```
DEPOTS_DEFAULT = /opt/accurev/storage/depots
```

This will be the default location for the storage directories (slices) of newly created depots. It need not be the same as the location where you placed the copies of existing depot slices in Step 11.

Note: The location of the default depot storage directory need not be the same on **replhost** as on **masthost**. And if you implement replication on a single machine (that is, **masthost** and **replhost** are the same machine), the two depot storage locations *must* be different.

19. Add two new settings, LOCAL_SERVER and LOCAL_PORT, which point to the AccuRev Server process running on **replhost**:

```
LOCAL_SERVER = replhost
LOCAL_PORT = 5050
```

These settings must match the MASTER_SERVER and PORT settings you noted in Step 13 and preserved in Step 14.

20. Save the revised **acserver.cnf** configuration file. According to the examples in the steps above, the file now contains

```
MASTER_SERVER = masthost
PORT = 5050
LOCAL_SERVER = replhost
LOCAL_PORT = 5050
SITE_SLICE_LOC = /opt/accurev/storage/site_slice
DEPOTS_DEFAULT = /opt/accurev/storage/depots
LOG_LEVEL = 2
```

The order of these settings is irrelevant. We suggest you place the SITE_SLICE_LOC and DEPOTS_DEFAULT settings after the LOCAL_SERVER and LOCAL_PORT settings, as a reminder that they indicate the locations in the local replica repository, not in the master repository.

There is no relationship between the LOCAL_PORT and PORT numbers. They can be the same or different.

Start the AccuRev Server Process on the Replica Server

21. Make sure that the **replhost** operating system process in which the AccuRev Server will run has an AccuRev username ("principal-name") identity with enough rights to access all the files in the **masthost** repository. AccuRev ACLs control access to depots and streams for specified AccuRev users and groups.

    - Unix: use environment variable ACCUREV_PRINCIPAL to establish the AccuRev principal-name of the operating system process. For example, in a Unix Bourne shell:

      ```
      ACCUREV_PRINCIPAL=acadmin; export ACCUREV_PRINCIPAL
      ```

    - Windows: the AccuRev Server runs as a Windows service; reconfigure it to run as the desired AccuRev principal-name, instead of LocalSystem. In the Control Panel's Services applet: open the Properties window for the AccuRev service, go to the Log On tab, select "This account", and enter the AccuRev principal-name and Windows password.

On either Unix or Windows, make sure that the AccuRev-level password of the desired principal-name is set. (This password is independent of OS passwords.) The AccuRev-level password is stored in file **authn**, in the **.accurev** subdirectory of the OS home directory.

22. Start the AccuRev Server process on **replhost**.

Host **replhost** is now ready to use as a replica server, with host **masthost** as the corresponding master server.

Change the Depot Slice Settings on the Replica Server

In Step 11, you copied the depot slices (depot storage directories) from the data-transfer medium to disk storage on **replhost**. For example, a depot that was stored on **masthost** at:

```
C:\Program Files\accurev\storage\depots\talon_depot
```

... might have been placed on **replhost** at:

```
/opt/accurev/storage/depots/talon_depot
```

For each depot that "lands" at a different pathname on **replhost** than it has on **masthost**, you must run the **chslice** command to record the different pathname.

23. Set up a client machine to use **replhost** as its AccuRev server. (See *Setting Up a Client Machine to Use a Replica Server* below for details.)

24. Using the **show slices** and **chslice** commands, change the slice pathname for some or all of the replicated depots. Example:

```
> accurev show slices
Slice#  Location
  ...
14      C:/Program Files/accurev/storage/depots/talon_depot
  ...

> accurev chslice -s 14 -l /opt/accurev/storage/depots/talon_depot
Changed location of slice '14' .
```

25. Use a final **show slices** command to verify that the AccuRev Server process on **replhost** has the correct pathname for all replicated depots.

Setting Up a Client Machine to Use a Replica Server

A machine on which the AccuRev client software is installed can use any server — either a replica server or the master server. As always, SERVERS setting in the client configuration file — **acclient.cnf** in the AccuRev **bin** directory — specifies which AccuRev Server process is to be sent client command requests. Examples:

```
SERVERS = replhost:5050          (use replica server)

SERVERS = masthost:5050          (use master server)
```

You can switch a client back and forth among multiple replica servers (and possibly the master server, too). It's as simple as editing the client's **acclient.cnf** file.

Using a Replica Server

When your client machine is set up to use a replica server, you can issue all AccuRev commands in the usual way. In general:

- Commands that *read* data from the repository — such as **files**, **diff**, and **cat** — use the replica repository.
- Commands that *write* data to the repository — such as **keep**, **promote**, and **merge** — use the master replica. After the master replica has been modified, the local replica is automatically brought up to date. For details, see *Synchronizing a Replica Manually* on page 32, which describes how you can bring the local replica up to date when you are *not* writing data to the repository.

The Update Command

The **update** operation is somewhat of a special case. Currently, an **update** executed on a replica server works as follows:

- A **stat** operation is performed on the replica server, to determine the state of the workspace stream and its backing stream.
- Data files representing new versions of elements are copied from the file storage area in the master repository to your workspace tree.
- Database transactions are copied from the master repository to the replica repository, bringing the replica database completely up to date.
- Storage files corresponding to those database transactions are not copied to the replica repository. See *On-Demand Downloading of a Version's Storage File* on page 32.
- The transaction level of the workspace is set to the most recent transaction (or to the transaction specified with **update –t**).

Usage Notes: March, 2005

The following notes apply to the AccuRev 3.7 replication capability:

- The repository replication scheme does not propagate newly created depots from **masthost** to **replhost**. Use the following procedure to propagate a newly created depot to each replica server machine.

  1. Stop the AccuRev Server process on **replhost**.

  2. Copy the following data from **masthost** to the corresponding location on **replhost**: all the files under the **storage\site_slice** directory, along with the directories under **storage\depots** that represent the newly created depots.

  3. Start the AccuRev Server process on **replhost**.

4. Use the **chslice** command on replhost to record the correct pathname of each new depot slice directory under **storage\depots**.

- Suppose a client machine is using the master server, and you update a workspace using that machine. If you then switch the client machine to using a replica server, you must perform a **replica sync** command before updating the workspace again.
- Do not create new depots in a replicated repository. Neither automatic nor manual synchronization propagates newly created depots to existing replica(s).
- All Dispatch operations are handled by the master server. Thus, replication does not improve Dispatch performance in this release.
- Harmless "double unlocking" messages are sent to the **acserver.log** file on the replica server after write operations.
- Communications errors between the master server and the replica server may not be clearly reported to the client.

Synchronizing a Replica Manually

During the course of development, your local replica repository typically becomes out-of-date with respect to the master repository. This occurs when other users send commands to other replica servers or directly to the master server. In both such cases, new transactions are entered in the master repository, but are not entered in the your local replica repository.

At any time, you can enter this CLI command to bring your local replica repository up to date:

```
accurev replica sync
```

This command transfers database transactions from the master repository to the replica repository. It does not transfer the corresponding storage files for **keep** transactions. See *On-Demand Downloading of a Version's Storage File* below.

A **replica sync** command is performed automatically on the local replica after each operation that is initiated by a client of the local replica, and that makes a change to the repository. See *Using a Replica Server* on page 31.

Note: you never need to synchronize directly with other replicas; synchronizing with the master is sufficient to bring your replica up to date.

On-Demand Downloading of a Version's Storage File

As a performance optimization, AccuRev copies database transactions only — not storage files that hold the contents of **keep** versions — when it transfers data from the master repository to a replica repository. Such transfers take place:

- ... during a **replica sync** command.
- ... during the automatic replica synchronization that follows an operation, invoked by a client using a replica server, that modifies the repository.

- ... during an **update** of a workspace. (**update** copies files to the workspace tree on the client machine, but not to the repository's file storage area on the replica server.)

Storage files can be quite large — particularly images and video clips — and many of them may never need to be accessed by users. For example, suppose a user creates intermediate versions 5,6,7, and 8 of an element, before finally creating and promoting version 9. Chances are no one will ever need to access the contents of versions 5–8 again. (But the master repository does preserve those versions' storage files, just in case!)

The storage file for a version is downloaded from the master repository to the local replica repository when a client using that replica explicitly references that version. Examples:

```
accurev cat -v talon_dvt/12 foo.c
accurev diff -v talon_dvt/12 foo.c
```

Both these commands cause the storage file for version **talon_dvt/12** of file **foo.c** to be downloaded to the local replica before the command itself is executed.

Automating Replica Synchronization

If a workgroup is much less active than other workgroups, its local replica repository can "fall behind" the master repository significantly. This can also occur if the workgroup uses the local replica repository mostly as a reference — for frequent read operations, but infrequent write operations. Falling behind in this way does no harm, but it can be bothersome. When some user finally does perform a write operation — keeping a new version of a file, or changing the location of a workspace — the local replica repository automatically "catches up", which might involve downloading tens or hundreds of transactions.

To prevent the local replica repository from falling too far behind, we recommend that you use operating system tools to perform an **accurev replica sync** command automatically, at regular intervals — say, every 15 minutes. On a Windows machine, use the Scheduled Tasks applet in the Control Panel. On a Unix/Linux host, set up a **cron** job to execute this command.

Synchronization Security

> Note: this section describes a security risk that exists only for organizations using the *AccuRev Replication Server* product. This risk does not apply to organizations that use the standard AccuRev software, without the replication option.

The repository synchronization scheme poses a potential security risk: the **acserver.cnf** server configuration file on an AccuRev server machine can name *any* master server machine in a MASTER_SERVER setting. And by default, the targeted master server will comply with *any* synchronization request — even an **accurev replica sync** command executed on a completely unrelated client machine.

We strongly recommend using the **server_admin_trig** trigger on the master server machine to implement an authentication scheme, so that the master server will send repository data over the

wire only to valid requestors. The following Perl code might be added to the sample **server_admin_trig** script included in the **examples** subdirectory of the AccuRev distribution:

```
if ($command eq "replica_sync") {
  if ($principal ne "rep01_acadmin" and $principal ne "rep02_acadmin") {
    print TIO "Repository synchronization disallowed:\n";
    print TIO "Authentication by the server_admin_trig script failed.\n";
    close TIO;
    exit(1);
  }
}
```

This code allows users **rep01_acadmin** and **rep02_acadmin** to perform repository synchronization, rejecting requests from all other user identifies.

Note: a **server_admin_trig** script identifies the command as **replica_sync**, even though the actual CLI command is **replica sync**.

Moving the AccuRev Server and Repository to Another Machine

The AccuRev data repository should be physically located on the machine that runs the AccuRev Server process. (This is firm dictate, but not an absolute restriction — see *READ ME NOW: Assuring the Integrity of the AccuRev Repository* on page 1.) The repository consists of multiple slices: the site slice contains information that pertains to the entire repository, and each depot has its own slice. Each slice contains a database consisting of multiple files.

From time to time, you may want (or need) to have the AccuRev server process run on a different machine. To accomplish this, you must:

- Perform a "full" (client and server) installation of AccuRev on the new machine.
- Move the data repository to the new machine.

If the new machine has a different byte order than the old machine, you must migrate each slice to use the "opposite-endian" data format. This involves swapping the bytes in each machine-level word. The **maintain migrate** command performs this conversion. With no arguments, it migrates the site slice. With an argument, it migrates a depot slice. The results of the migration are stored in a new subdirectory named **swapped** within the site-slice or depot directory.

This procedure is safe: the original slice data is never modified, and there is no harm in running the **maintain migrate** multiple times on a slice.

Procedure for Moving the Repository

Make sure you perform each of the following steps on the appropriate server machine. We call them:

- The "source" machine — where the AccuRev server is currently running and the data repository is currently located.
- The "destination" machine — the machine to which you want to move the data repository.

  Note: the steps below always show Unix pathname separators (/). When you're executing commands on a Windows machine (either source or destination), be sure to use Windows pathname separators (\ ).

The procedure calls for multiple stops and starts of the AccuRev Server process. For details on how to accomplish this, see *Controlling Server Operation* on page 12.

On the Destination Machine

1. Perform that a "full" (that is, both client and server) installation of AccuRev on this machine. In the steps below, we'll refer to the installation directory on the destination machine as <*dest-install-dir*>.

2. Run this command on the destination machine:

   ```
   accurev hostinfo
   ```

3. Send the output of this command to AccuRev Customer Support, as part of a request for a new license key for the destination machine. When you get the new license key, install it in the **site_slice** directory, according to the supplied instructions.
4. Store an additional copy of the license key outside the AccuRev repository, for use in Step 14.
5. *Do the following only if you're moving the repository to an opposite-endian machine:*

   5a) Copy the following file to a secure location:

   ```
   <dest-install-dir>/storage/site_slice/datadict.ndb
   ```

On the Source Machine

6. Execute the command **accurev show slices** and **accurev show depots**, and save the output for reference in the following steps.
7. Stop the AccuRev Server process. (Reminder: see *Controlling Server Operation* on page 12.)
8. Perform a full backup of the AccuRev repository, as described in *Backing Up the Repository* on page 3.
9. *Do the following only if you're moving the repository to an opposite-endian machine:*

   9a) Migrate the site slice, using the **maintain** program located in the AccuRev **bin** directory:

   ```
   maintain migrate
   ```

   This creates a **swapped** subdirectory under the **site_slice** directory.

   9b) Move all ***.ndb** database files from the **site_slice/swapped** directory up to the **site_slice** directory. This overwrites the original ***.ndb** files in the **site_slice** directory.

   9c) Remove all ***.ndx** index files from the **site_slice** directory. These files will be regenerated on the destination machine.

   9d) For each depot listed in the **show depots** output (see Step 6), migrate the depot's slice:

   ```
   maintain migrate <depot-name>
   ```

   This creates a set of **swapped** subdirectories in the depot slices.

   9e) Move all ***.ndb** files from the ***<depot-name>*/swapped** directories up to the parent ***<depot-name>*** directories. This overwrites the existing ***.ndb** files in the ***<depot-name>*** directories.

   9f) Remove all ***.ndx** index files from the ***<depot-name>*** directories. These files will be regenerated on the destination machine.

10. Copy the entire tree starting at directory **storage** within the AccuRev installation area. Be sure to use a method that preserves file ownership (e.g. **tar –cp**).

On the Destination Machine

11. Stop the AccuRev Server process.
12. Rename ***<dest-install-dir>*/storage** (e.g. to **storage.OLD**).
13. "Paste" (unpack, unzip, tar –x) the copy of the directory tree you made in Step 10, so that the pasted data becomes ***<dest-install-dir>*/storage**.
14. Overwrite file ***<dest-install-dir>*/storage/site_slice/keys.txt** with the copy of the license key that you saved in Step 4.
15. *Do the following only if you're moving the repository to an opposite-endian machine:*

    15a) Restore the copy of ***<dest-install-dir>*/storage/site_slice/datadict.ndb** that you made in Step 5a.

    15b) Reindex the site slice:

    ```
    maintain reindex
    ```

16. If the **storage** directory is located at a different pathname on the source and destination machines:

    16a) Start the AccuRev Server process.

    16b) For each depot, determine the corresponding slice number (see Step 6) and run this command:

    ```
    accurev chslice -s <slice-number>
       -l <dest-install-dir>/storage/depots/<depot-name>
    ```

    16c) Stop the AccuRev Server process.

    Perform the following step only if you're moving the repository to an opposite-endian machine:

    16d) Reindex each depot:

    ```
    maintain reindex <depot-name>
    ```

17. Start the AccuRev Server process.

AccuRev Triggers

A trigger is a "code hook" or callback built into certain AccuRev commands. When a user enters the command, the corresponding trigger "fires"; this causes a user-defined or built-in procedure to be performed just before or after the command executes. Typically, a user-defined procedure is implemented as a script in the Perl scripting language. Sample Perl scripts are available in the **examples** subdirectory of the AccuRev installation directory.

AccuRev supports both pre-operation triggers and post-operation triggers. In addition, there are triggers that integrate AccuRev's configuration management facility with its issue management facility (Dispatch); these triggers have pre- and post-operation components.

Pre-Operation Triggers

The following triggers execute a procedure before the user-requested command executes. Each of these triggers has the ability to cancel execution of the user's command. Some of the triggers fire on the client machine, and others on the server machine. It's possible for a single command (e.g. **keep**) to cause triggers to fire both on the client and on the server.

Client-Side Triggers

The following pre-operation triggers fire on the client machine:

- **pre-create-trig**: fires on the client machine prior to execution of an **add** command.
- **pre-keep-trig**: fires on the client machine prior to execution of a **keep** command.
- **pre-promote-trig**: fires on the client machine prior to execution of a **promote** command.

You enable these triggers on a per-depot basis, using the **mktrig** command. For example, to implement the **pre-keep-trig** trigger in depot **WidgetDepot** with script **check_for_comments**:

```
accurev mktrig -p WidgetDepot pre-keep-trig check_for_comments
```

Server-Side Triggers

The following pre-operation triggers fire on the server machine.

- **server_admin_trig**: fires on the server machine prior to execution of certain commands. This is a repository-wide trigger — it fires no matter what depot, if any, the user's command applies to. As of Version 3.7, the following commands cause **server_admin_trig** to fire:

```
mkuser      rmmember      lock          defcomp
chuser      mkgroup       unlock        repl_sync
chref       mkdepot       chws
chdepot     mktrig        chstream
chslice     rmtrig        mkstream
lsacl       setacl        remove
addmember   write_schema  mkws
```

The **defcomp** command is not user-visible; it's used in the implementation of the include/exclude facility CLI commands **incl**, **excl**, and **incldo**. The **repl_sync** command recognized by the **server_admin_trig** trigger corresponds to the CLI command **replica sync**.

Note: prior to Version 3.5, the services now provided by **server_admin_trig** were provided by a trigger named **server_all_trig**. (In the AccuRev-provided sample trigger script, this includes allowing commands to be executed only by a specified list of AccuRev users.) For backward compatibility, the older trigger is still supported.

- **server_preop_trig**: fires on the server machine prior to execution of certain commands. This is a depot-specific trigger — it fires only for commands that operate on the depot(s) where the trigger has been activated. As of Version 3.5, the following commands cause **server_preop_trig** to fire:

```
add         promote
keep        purge
```

The **server_admin_trig** and **server_preop_trig** triggers are designed to be mutually exclusive — for a given command, at most one of these triggers fires. (For many commands, neither of these triggers fires.)

You don't enable these triggers using the **mktrig** command. Instead, the trigger is enabled when you place an executable script file with the corresponding name in the appropriate directory: See *Using an AccuRev-Provided Trigger Script* on page 42 for details.

Post-Operation Triggers

The following triggers execute a procedure after the user-requested command executes successfully. If the user's command fails, the post-operation trigger does not fire. A post-operation trigger always fires on the server machine.

- **server-post-promote-trig**: fires on the server machine subsequent to execution of a **promote** command. You enable this trigger with a **mktrig** command.
- **server_dispatch_post**: fires on the server machine each time a Dispatch issue record is created or modified. You enable this trigger by placing an executable script file with the corresponding name in the AccuRev executables (**bin**) directory. For more information on this trigger, see *The server_dispatch_post Trigger* on page 4 of the *Dispatch Issue Management Manual*.

Trigger for Transaction-Level Integration between Configuration Management and Issue Management

You can achieve tight coordination of your organization's configuration management and issue management capabilities by enabling one or both of the integrations between AccuRev's configuration management and issue management facilities. The transaction-level integration is enabled by a trigger on a depot-by-depot basis:

```
accurev mktrig -p WidgetDepot pre-promote-trig client_dispatch_promote
```

The “client_dispatch_promote” integration routine is built into the AccuRev software — no scripts are required — and includes both pre-operation and post-operation components:

1. On the client machine, a user invokes the AccuRev **promote** command.
2. The pre-operation part of the trigger fires on the client machine, prompting the user to specify one Dispatch issue record. If this part of the trigger fails (e.g. the user specifies a non-existent issue record), the **promote** command itself is cancelled.
3. The **promote** command completes, and is recorded in the AccuRev repository as a transaction.
4. The post-operation part of the trigger fires on the server machine, updating the issue record that the user specified by adding the number of the **promote** transaction to the **affectedFiles** field.

For more information on using and customizing this transaction-level integration, along with information on the other (change-package-level) integration, see *Integrations Between Configuration Management and Issue Management* on page 49 of the *Dispatch Issue Management Manual*.

Using an AccuRev-Provided Trigger Script

Sample trigger scripts are installed with AccuRev, in the **examples** subdirectory. These sample scripts are implemented in the platform-neutral Perl scripting language. Use the following procedure to install and use one of these scripts:

1. **Install Perl**. There are many source on the Web for Perl. We recommend the ActivePerl distribution from http://www.activestate.com. This distribution includes a conversion utility, **pl2bat**, which makes a Perl script executable under Windows, by embedding the Perl code in a Windows batch file (**.bat**).

   Be sure to install Perl on all appropriate machines. Note that some pre-operation triggers run on the client machine, while others run on the server machine. All post-operation triggers run on the AccuRev server machine.
2. **Get a copy of the sample script**. Copy the sample script from the **examples** subdirectory of the AccuRev installation directory to an AccuRev workspace. Then use the **add** command to place the script under version control.
3. **Prepare the script**. Open the script in a text editor, and customize the script according to the instructions included as comment lines. These instructions also explain how to make the script file executable and how to install the script file in the appropriate directory on the client or server machine.
4. **Enable the trigger**. Use the **mktrig** command to enable use of the script in a particular depot. For example:

   ```
   accurev mktrig -p WidgetDepot pre-keep-trig addheader
   ```

   Certain triggers are enabled simply by placing the script file in the appropriate directory, not by executing a **mktrig** command:

- **server_admin_trig**: Place an executable file in subdirectory **triggers** of the **site_slice** directory. Unix: the file must be named **server_admin_trig** or **server_admin_trig.pl**; Windows: the file must be named **server_admin_trig.bat**. Example:

  ```
  C:\Program Files\AccuRev\storage\site_slice\triggers\server_admin_trig.bat
  ```

  If the directory contains both a **server_admin_trig** executable and an old **server_all_trig** executable, only the **server_all_trig** program is executed.

- **server_preop_trig**: Place an executable file in subdirectory **triggers** of the slice directory of one or more depots. (The command **accurev show slices** displays the locations of slice directories.) Unix: the file must be named **server_preop_trig** or **server_preop_trig.pl**; Windows: the file must be named **server_preop_trig.bat**. Example:

  ```
  /opt/accurev/storage/depots/talon_tests/triggers/server_preop_trig
  ```

- **server_dispatch_post**: This script runs after a Dispatch transaction takes place. For details, see *The server_dispatch_post Trigger* on page 4 of the *Dispatch Issue Management Manual*.

Notes on Triggers in Multiple-Platform Environments

Observe these guidelines in a multiple-platform environment, where the trigger script will be accessed from both Windows machines and Unix machines:

- Don't even try to arrange for exactly the same script file to be accessed by all users, on all platforms. Instead, place scripts to be accessed by Unix users in one directory and equivalent scripts to be accessed by Windows users in another directory.
- Make sure the Windows script has a **.bat** suffix (e.g. **check_for_comments.bat**), and that the Unix script has no suffix (**check_for_comments**).

  Note: on a Windows machine, AccuRev searches for a trigger script file named **check_for_comments** before it searches for a batch file named **check_for_comments.bat**. That's another reason to place Windows and Unix scripts in separate directories.
- Make sure the scripts run correctly on their respective platforms. And remember to revise *all* versions of the script when you revise any one of them!
- In the **mktrig** command, specify the script name without a suffix — e.g.:

  ```
  accurev mktrig -p WidgetDepot pre-keep-trig check_for_comments
  ```

The Trigger Parameters File

When a trigger fires and executes a user-supplied program, AccuRev passes a single argument to the trigger script or program: the pathname of a trigger parameters file containing information about the transaction that is about to be performed (or was just completed).

The layout of the parameters file varies among the trigger types. For example, the parameters file passed to a **pre-keep-trig** trigger script is a simple text file, with this line-by-line layout:

```
trigger name
depot name
stream name
number of lines in "keep" command comment
comment lines (if the previous value is non-zero)
pathname of top-level directory of workspace
principal-name of user who executed command
specs of 1st version to be created: pathname version-ID data-type
specs of 2nd version to be created: pathname version-ID data-type
...
```

The parameters file passed to a **server_admin_trig** or **server_preop_trig** script is in XML format:

```
<triggerInput>
  <hook> ... </hook>
  <command> ... </command>
  <principal> ... </principal>
  <ip> ... </ip>
</triggerInput>
```

The sample set of trigger scripts includes a Perl script for each kind of trigger. The script's comments include a detailed description of the layout of the parameters file for that kind of trigger.

Trigger Script Contents

A trigger script can send email, start a build, update a Web site, or perform many other tasks. In particular, you can run AccuRev commands to get more information. One common use of the **server-post-promote-trig** trigger is to run the **hist** command using the transaction number of the promotion, generating the list of promoted elements for inclusion in an email notification.

The exit status (return value) of a **pre-create-trig**, **pre-keep-trig**, or **pre-promote-trig** script is important:

- A zero exit status indicates success: the AccuRev command (**add**, **keep**, or **promote**) is allowed to proceed.
- A non-zero exit status indicates failure: the AccuRev command is canceled and the depot remains unchanged.

File Handling by Trigger Scripts

A trigger script can overwrite its parameters file (after reading it, presumably). This provides a way for the script to communicate with the AccuRev command or with a "downstream" script:

- A **pre-create-trig** script must overwrite its parameters file with data that indicates the type of each element to be created. Each line must describe one new element:

  ```
  <full-pathname-of-element>  <element-type>
  ```

  The <*element-type*> is a numeric code:

  1 directory
  2 text file
  3 binary file

  See sample trigger script **elem_type.pl** in the **examples** subdirectory of the AccuRev installation directory.

- The parameters file for a **pre-keep-trig** script ends with a series of lines, one per element to be kept:

  ```
  <pathname-of-element>  <version-ID>  <element-type>
  ```

  <*pathname-of-element*> is not a full filesystem pathname, but starts at the workspace's top-level directory (which is included earlier in the parameters file). <*version-ID*> is the new version to be created for that element. <*element-type*> is the numeric code 1, 2, or 3, as described above. Note that different versions of an element can have different types.

  See sample trigger script **addheader.pl** in the **examples** subdirectory of the AccuRev installation directory.

- The parameters file for a **pre-promote-trig** script ends with a series of lines, one per element to be kept:

  ```
  <pathname-of-element>
  ```

  <*pathname-of-element*> is not a full filesystem pathname, but starts at the workspace's top-level directory (which is included earlier in the parameters file).

  A **pre-promote-trig** script can overwrite its parameters file, in order to communicate with a **server-post-promote-trig** script: the *first line* of the overwritten parameters file becomes the value of the **from_client_promote** parameter in the **server-post-promote-trig** script.

  See sample trigger script **client_dispatch_promote_custom.pl** in the **examples/dispatch** subdirectory of the AccuRev installation directory, along with **server_post_promote.pl** in the **examples** subdirectory.

A trigger script can also send data to STDOUT and STDERR. If the command for which the trigger fired was executed in the AccuRev CLI, this data appears in the user's command window. If a GUI command caused the trigger to fire, the script's exit status determines whether the user sees the STDOUT/STDERR data: in the "failure" case (non-zero exit status), the data is displayed in an error-message box; in the "success" (zero exit status) case, the data is discarded.

Trigger Script Execution and User Identities

When a trigger script executes on a client machine, it runs under the identity of the AccuRev user who entered the command. Since the user himself is registered (i.e. has a principal-name) in the

AccuRev user registry, there won't be any authentication problems if the trigger script runs AccuRev commands that access the repository.

When a trigger script executes on the server machine, it runs under the user identity of the AccuRev Server itself. We recommend that the server run as user **acserver** or **System** (see *User Identity of the Server Process* on page 9), a user that should not be in AccuRev's user registry. If the trigger script runs AccuRev commands, it needs to have an AccuRev user identity. To accomplish this, set variable ACCUREV_PRINCIPAL to an appropriate principal-name (AccuRev username) in the environment of the **acserver** or **System** account. Alternatively, set this environment variable in the script itself. For example:

```
$ENV{ACCUREV_PRINCIPAL} = "jjp";
system("accurev hist ...");
```

'Administrative Users' in Trigger Scripts

The sample Perl trigger scripts supplied by AccuRev provide a very simple implementation of the "administrative user" concept: a user is permitted to perform certain operations only if his username is recorded in the **administrator** hash defined in the script:

```
$administrator{"derek"} = 1;
$administrator{"allison"} = 1;
   ...
if ( ! defined($administrator{$principal}) ) {
         print TIO "Execution of '$command' command disallowed:\n";
   ...
```

The Trigger Log File

When a trigger program runs on the AccuRev server machine — for a **server-post-promote-trig**, **server_preop_trig**, or **server_admin_trig** trigger — an invocation line is written to file **trigger.log** in the **logs** subdirectory of the repository's **site_slice** directory:

```
##### [2004/06/28 20:50:42] running: '"C:\Program Files\AccuRev\bin\pst_pro.bat" ...
```

If the script produces console output (STDOUT and/or STDERR), this output is also sent to the **trigger.log** file.

As with other server log files, the **trigger.log** file is "rotated" periodically, to keep active logs from growing too large.

The ‘maintain’ Utility

This document describes AccuRev’s **maintain** utility, an administrative tool for occasional use under the guidance of an AccuRev, Inc. Product Support representative. The **maintain** utility is a non-interactive command-line tool, with a simple command structure. For example:

```
maintain reindex <depot-name>
```

The **maintain** program is located in the AccuRev **bin** directory. If the command-line client program, **accurev**, is on your search path, then so is **maintain**.

Each of the **maintain** commands is described below. Several of these commands are related to the client command **accurev backup mark**, so we begin with a description of that command.

The ‘backup mark’ Command

The command **accurev backup mark** declares a “checkpoint” of the entire AccuRev repository, by:

- Copying all the database files (**.ndb**) and the index file **stream.ndx** from the **site_slice** directory to a subdirectory named **backup**.
- Recording the current state of each depot’s database files in file **valid_sizes_backup** in the depot directory.

This command does not actually copy any files to a backup medium; it just marks a consistent state of the repository files. After executing this command, you can back up the repository files, in any order, and on any schedule, while users continue to use AccuRev. The next section describes the **maintain** commands that restore the repository to its state at the time the **accurev backup mark** command was executed.

The ‘maintain’ Commands Related to ‘backup mark’

At any time after you’ve executed an **accurev backup mark** command and copied the repository files to a backup medium, you can restore the repository to its state at the time the **backup** command was executed. The procedure is documented in section *Restoring the Repository* on page 4 of the *AccuRev Administrator’s Guide*. The **maintain** commands involved are:

restore

```
maintain restore <depot-name>
```

Rolls back all the database files (**.ndb**) in the specified depot to their state at the time of the **accurev backup mark** command. This is a “logical truncate” operation — a file-system truncate is *not* performed on the database files.

reindex

```
maintain reindex [ <depot-name> ]
```

Reads various database files (**.ndb**) and recreates the corresponding index files (**.ndx**). This operation is performed on the database files of the specified depot; if you omit the <*depot-name*> argument, the operation is performed on the repository-wide database files in the **site_slice** directory

Additional 'maintain' Commands

This section describes the remaining **maintain** commands. The **chuptrans**, **timeshift**, and **rmdepot** command are not described below; they are to be used only under close supervision by AccuRev Support staff.

chpasswd

```
maintain chpasswd <user> <new-password>
```

Changes the password stored in the AccuRev repository for an existing principal-name (named AccuRev user). To remove a user's password, use two consecutive double-quote characters as the <*new-password*> parameter:

```
maintain chpasswd derek ""
```

For consistency, the user should invoke the **accurev setlocalpasswd** command to change his "local password" on each client machine that he uses. A user's local password is stored in file **authn** in subdirectory **.accurev** of the user's home directory.

dbcheck

```
maintain dbcheck
```

Checks the consistency of the streams database (**streams.ndb**) in the **site_slice** directory, and checks the consistency of all the database (**.ndb**) files in all depots.

Note: be sure to stop the AccuRev Server process before running this command.

migrate

```
maintain migrate
maintain migrate <depot-name>
```

Converts the specified depot from little-endian to big-endian format, or vice-versa. If you omit the <*depot-name*> argument, it converts the **site_slice** directory. In all cases, the conversion is non-destructive: the data structure itself is not modified or deleted; the converted data is written to a subdirectory named **swapped** within the **site_slice** directory or depot directory. Converting the "endian-ness" consists of reversing the order of the 8-bit bytes in each machine-level word. For a step-by-step conversion procedure, see *Moving the AccuRev Server and Repository to Another Machine* on page 36 of the *AccuRev Administrator's Guide*.

rmdepot

```
maintain rmdepot <depot-name>
```

Removes a depot from the AccuRev repository. All streams, snapshots, and workspace streams are also removed from the repository. (Workspace trees are *not* removed.) For details, see *Removing a Depot from the AccuRev Repository* on page 50.

vercheck

```
maintain vercheck <depot-name>
```

Checks the contains of the specified depot's **data** directory tree, to verify that a storage container file exists for each file version recorded in the depot database.

Removing a Depot from the AccuRev Repository

This section describes a procedure for removing a depot completely from the AccuRev repository. Removing a depot:

- Deletes every version of every file and directory in the depot.
- Deletes the entire history of the depot — all transactions involving the depot and its elements.

Removing a depot does not affect any of the workspaces or reference trees that contain copies of the depot's elements.

Before You Begin

We strongly recommend that you preserve a backup copy of the AccuRev data repository before deleting any depots. See *Backing Up the Repository* on page 3.

Depot Removal Procedure

The following procedure must be performed on the machine where the AccuRev repository resides.

1. Stop the AccuRev Server process:
   - Unix: use the **acserverctl** utility, located in the AccuRev **bin** directory:

     ```
     acserverctl stop
     ```

   - Windows: use the **Services** applet, or enter the command **net stop accurev** in a Command Prompt window.
2. Execute the **maintain** program's remove-depot command. This utility program is located in the AccuRev **bin** directory.

   ```
   maintain rmdepot <depot-name>
   ```

   For safety, the **rmdepot** command goes through two confirmation steps, including having you retype the depot name. The command displays some messages, ending with:

   ```
   Site reindex complete.
   ```

The **rmdepot** command completely removes the depot's records from the repository database in the **site_slice** directory. (There's one exception: the depot's name remains in the database. See Step 5.)

3. Remove the depot directory subtree from the AccuRev **storage** directory tree. For example, if the depot is named **widget** and AccuRev is installed on a Unix machine at **/opt/accurev**, use this command:

   ```
   rm -r /opt/accurev/storage/depots/widget
   ```

   Be careful not to remove any other depot's directory subtree!

4. Restart the AccuRev Server process:

   - Unix: use the **acserverctl** utility, located in the AccuRev **bin** directory:

     ```
     acserverctl start
     ```

   - Windows: use the **Services** applet, or enter the command **net start accurev** in a Command Prompt window.

5. At this point, the depot's name remains in the AccuRev repository database. (It won't appear in depot-name listing, though.) If you wish to reuse the removed depot's name, perform a rename-depot command. For example:

   ```
   > accurev chdepot -p widget deleted_widget
   Unknown stream or ver spec: widget
   Unknown stream or ver spec: widget
   Changed name of depot widget to deleted_widget .
   ```

   At this point, you can create a new depot named **widget**, which will be completely unrelated to the depot you removed.