AccuRev

AccuRev®

Administrator's Guide

Version 6.2

Revised 19-January-2015

Copyright and Trademarks

This product incorporates technology that may be covered by one or more of the following patents: U.S. Patent Numbers: 7,437,722; 7,614,038; 8,341,590; 8,473,893; 8,548,967.

AccuRev, **AgileCycle**, and **TimeSafe** are registered trademarks of AccuRev, Inc.

AccuBridge, **AccuReplica**, **AccuSync**, **AccuWork**, **Kando**, and **StreamBrowser** are trademarks of AccuRev, Inc.

Table of Contents

Preface ix
Audience ix
Before You Begin ix
Using This Book ix
Typographical Conventions x
Contacting Technical Support x

1. Setting Up AccuRev® 1
Before You Begin 1
Using the Quick Setup Wizard 1
Manual Set-Up Procedures 3
Create an AccuRev Username and Password 3
Create a New Storage Depot 3
Create a Workspace to Store Your Work 3
Import Files into Your Workspace 3
Add the Files in Your Workspace to the Depot 4
Update Your Workspace 4
Next Steps 4

2. The AccuRev Repository 5
Repository Access Permissions 5
READ ME NOW: Assuring the Integrity of the AccuRev Repository 5
Backing Up the Repository 6
Restoring the Repository 8
Archiving Portions of the Repository 9
Moving a Workspace or Reference Tree 9
Moving a Depot 9
Removing a Depot 9
Moving the db and site_slice Directories 9
A Word of Caution on Windows Zip Utilities 10
Storage Layout 11

3. The AccuRev Server 13
Operating-System User Identity of the Server Processes 13
UNIX/Linux Identity 13
Windows Identity 13
AccuRev User Identity of the Server Process 14
Starting the AccuRev Server 15
Running the Server Automatically at Operating System Startup 15
Starting the Server Manually 15
Server Configuration File 16

UNIX/Linux: Controlling the Server's Operating-System User Identity 16
Controlling Login Session Longevity 17
Controlling Multithreading of the AccuRev Server 17
Configuring Access to the Database 17
Server Logging 18
Time zone offset 18
Logging Levels 18
UNIX/Linux: Log File Rotation 18
Controlling Server Log Verbosity 19
Verbose Server Logging 19
Server Watchdog 20
Watchdog Logging 21
Controlling Server Operation 21
UNIX/Linux: 'acserverctl' Utility 21
Windows: 'Services' Control Panel 22
Server-Control Files 22
Open Filehandle Limits and the AccuRev Server 24
Checking the Number of File Descriptors 24
Changing the Per-Process Open File Descriptor Limit 24
AccuRev Server Performance 25
AccuRev Architecture Overview 25
Key Factors Affecting AccuRev Performance 26
Resource Utilization and the Client-Server Architecture 26
AccuRev Server Machine Usage: Software 27
AccuRev Server Machine Networking 27
Using Multiple AccuRev Servers 27
Setting Up Client Machines 28
Workspaces and Servers 28
Specifying a Server By Using the –H Option 29
Configuring the Web User Interface (Web UI) 29

4. System Clock Synchronization 31
Detecting System Clock Discrepancies -- Timewarp 31
AccuRev's Timewarp Detection Scheme 31
Fixing System Clock Discrepancies 32
Automatic, Gradual Convergence of System Clocks 32
Manual Synchronization Tools 33

5. Timestamp Optimization (TSO) 35
Local, Client-Based Timestamp Optimization 35
Managing TSO Behavior 35

6. Archiving of Version Container Files 37
The 'archive' Command 38
Determining Which Versions to Archive 38

Dry Run Capability 38
Archiving the Versions 38
The ‘reclaim’ Command 39
Attempts to Access Archived Versions 40
Using ‘hist’ to Research Previous ‘archive’ Commands 40
Restoring Archived Versions -- The unarchive Command 40

7. Replication of the AccuRev Repository 43

Master and Replica 43
AccuRev Licensing in a Replication Environment 45
Installing and Configuring AccuRev in a Replication Environment 45
Configure an AccuRev Server as a master server 45
Configure a Replica server 45
Establish an AccuRev User Identity for the AccuRev Server Process 46
Synchronize the Site Slice 47
Indicate the Depots to be Replicated 47
Setting Up a Client Machine to Use a Replica Server 47
Using a Replica Server 48
The Update Command 48
Command Interaction in a Replicated Environment 48
Removing Storage Containers on a Replica Server 49
Removing a Replica Server 49
Improving Replica Performance 49
Triggers and Replication 50
Creating New Depots 51
Adding and Removing Depots from a Replica Repository 51
Synchronizing a Replica Manually 51
On-Demand Downloading of a Version’s Storage File 52
Automating Replica Synchronization 52
Synchronization Security 52
The replica_site.xml File 53

8. Moving the AccuRev Server and Repository to Another Machine 55

Procedure for Moving the Repository 55
On the Source Machine 55
On the Destination Machine 55

9. AccuRev Security Overview 57

Users and Groups 57
Usernames and Groupnames 57
User Authentication 57
The “accurev_login” User-Authentication Method 58
The “custom” User-Authentication Method 58
Authenticating a Replica User on the Master 58

The ‘server_auth_trig’ Script 58
Selecting the User-Authentication Method 59
How AccuRev Records the User-Authentication Method 59
Restriction on Access to the “Add User” Command 60
Locks on Streams 60
Access Control List (ACL) Permissions 60
Element-Level Security (EACLs) 60
Features 61
Basic Terms 61
Important Concepts 62
Privileges 63
Inheritance 63
Access Denied 63
Create a superuser to Administer Element Security 64
Auditing 64
Setting Permissions for a Replica Server 64
EACL Usage Scenarios 65
Restricting Access to Commands using Triggers 69
Implementing Secure Sockets Layer (SSL) Encryption 70
Generating a Private Key 70
Obtaining an SSL Certificate 71
Enabling SSL Encryption on the AccuRev Server 73
Managing SSL 74

10.AccuRev Triggers 77

Pre-Operation Triggers 77
Client-Side Triggers 77
Server-Side Triggers 77
Post-Operation Triggers 78
Triggers in a Replication Environment 79
Triggers and Security 79
Preparing to Use an AccuRev-Provided Trigger Script 79
Enabling a Trigger 80
pre-create-trig, pre-keep-trig, pre-promote-trig, server-post-promote-trig 80
server_admin_trig 80
server_preop_trig 80
server_dispatch_post 80
Notes on Triggers in Multiple-Depot Environments 81
Notes on Triggers in Multiple-Platform Environments 81
The Trigger Parameters File 82
Format of the “pre-create-trig” Trigger Parameters File 83
Format of the “pre-keep-trig” Trigger Parameters File 84
Format of the “pre-promote-trig” Trigger Parameters File 86
Format of the “server-post-promote-trig” Trigger Parameters File 87
Format of the “server_preop_trig” Trigger Parameters File 88
Format of the “server_admin_trig” Trigger Parameters File 94

Format of the “server_dispatch_post” Trigger Parameters File....96
Encoding of Element Lists....96
Encoding of Reserved Stream Properties....97
Restricting the Ability to Set Stream and Principal Properties....97
Encoding of Command Comments....98
Trigger Script Contents....99
Trigger Script Exit Status....99
Trigger Script Execution and User Identities....100
‘Administrative Users’ in Trigger Scripts....100
The Trigger Log File....100
Disabling Triggers....101

11.Using Streams to Enforce Process....103

What is a Gated Stream?....103
Staging Streams....104
The server_master_trig Trigger....105
Triggers are Customized....105
Example: A Simple Use Case....105
The barcon Depot....106
Identifying the Need for Gated Streams....106
Setting Up the Trigger....107
Promoting Changes....107
Understanding Gated Stream Status....109
Adding Gated Streams....109
Creating Gated Streams....110
Implementing the server_master_trig Trigger....110
Step 1: Create a Non-expiring Session....110
Step 2: Edit the server_master_trig Trigger....111
Step 3: Create a Batch File for server_master_trig.pl....112
Working with Gated and Staging Streams....112
Hiding Empty Staging Streams....112
Changes to the Backing Stream....112
Troubleshooting....113
Trigger Log....113
Status Icons and Tooltips....113
Overriding Running Status....113

12.The ‘maintain’ Utility....115

Specifying a Database Admin Username and Password....115
‘maintain’ Command Reference....115
Backup/Restore of the AccuRev Repository....119
Removing a Depot from the AccuRev Repository....119
Before You Begin....119
Depot Removal Procedure....119
Reusing a Depot’s Name....120

13.License Management 121
AccuRev Product Licenses 121
Standard/Flexible Licenses 121
How Flexible Licenses Work 122
Enabling License Management 123
License, Configuration, and Options Files 123
Changing the RLM Listener Port Value 124
Accessing the RLM Web Interface 124
Specifying RLM Privileges 125
Configuring Multiple AccuRev Servers 126
Configuration Example 126
Replication Server Licenses 127
Additional RLM Documentation 127
Stopping and Starting the AccuRev and RLM Servers 128
Stopping the Servers 128
Starting the Servers 128

14.The Client-only Package Download Utility 129
Prerequisites 129
About the Client-only Installation Packages 129
Running the Client-only Package Download Utility 129

A. Code Page Support 131

Preface

The *AccuRev® Administrator's Guide* presents topics, procedures, and reference material that will be of interest to the AccuRev administrator.

Audience

This book is intended for the AccuRev administrator.

Before You Begin

This book assumes that you have already performed a full AccuRev installation (Server and Client).

Using This Book

This book assumes you are familiar with your operating system and its commands, as well as basic AccuRev concepts like depots and workspaces.

Chapter	Description
Chapter 1 Setting Up AccuRev®	Describes the initial set-up of AccuRev depots and workspaces using the Quick Setup Wizard and manual procedures.
Chapter 2 The AccuRev Repository	Describes the AccuRev data repository, its components, and procedures like back up and restore.
Chapter 3 The AccuRev Server	Describes the AccuRev Server and related procedures for starting and stopping, logging, and configuring.
Chapter 4 System Clock Synchronization	Describes AccuRev's facilities for detecting system-clock discrepancies, along with other related facilities commonly available on Windows and UNIX/Linux systems.
Chapter 5 Timestamp Optimization (TSO)	Describes AccuRev's Timestamp Optimization (TSO) algorithm, how it is used to identify modified workspace files, and features you can use to manage TSO.
Chapter 6 Archiving of Version Container Files	Describes how to use the archive and reclaim commands to manage container files of versions of AccuRev elements.
Chapter 7 Replication of the AccuRev Repository	Describes how to set up and use master and replica AccuRev data repositories.
Chapter 8 Moving the AccuRev Server and Repository to Another Machine	Describes how to prepare for and move the AccuRev Server and data repository to another machine.
Chapter 9 AccuRev Security Overview	Describes AccuRev security features including users and groups, locks on streams, access control lists (ACLs), and element-level security (EACLs).

Chapter	Description
Chapter 10 AccuRev Triggers	Describes AccuRev how to create and use triggers, which can be used to execute user-defined programs before or after certain AccuRev events.
Chapter 11 Using Streams to Enforce Process	Describes how you can use AccuRev gated streams to help ensure that changes are promoted only after passing externally defined processes like builds, test suites, and code reviews.
Chapter 12 The 'maintain' Utility	Describes how to use the maintain utility to perform tasks associated with AccuRev database administration.
Chapter 13 License Management	Describes the types of licenses supported by the Reprise License Manager (RLM) and how they work.
Chapter 14 Using Client-only Install Packages	Describes how to use the Client-only Package Download Utility to download AccuRev Client-only installation packages from the AccuRev web site to the AccuRev Server.
Appendix A Code Page Support	Describes code pages supported by AccuRev.

Typographical Conventions

This book uses the following typographical conventions:

Convention	Description
`blue sanserif`	Used for sample code or output.
`red monospace`	Used for examples.
bold	Used for command names, and button names in the AccuSync Web user interface
light italic	Used for emphasis, book titles, and for first use of important terms
blue italic	Identifies a hyperlink (to a page or Web URL, for example)

Contacting Technical Support

Micro Focus offers a variety of options to meet your technical support needs as summarized in the following table.

For	Visit
Information about technical support services	*http://supportline.microfocus.com/*
Information about platforms support	*http://supportline.microfocus.com/prodavail.aspx*
Product downloads and installations	*http://supportline.microfocus.com/websync/productupdatessearch.aspx*
Product documentation	*http://supportline.microfocus.com/productdoc.aspx*
SupportLine phone numbers, listed by country	*http://www.microfocus.com/about/contact/support/assistance.aspx*

When you contact Micro Focus technical support, please include the following information:

- The version of AccuRev and any other AccuRev products you are using (AccuSync or GitCentric, for example).
- Your operating system.
- The version of relevant third-party software (if you are using AccuSync, for example, the version of your ITS).
- A brief description of the problem you are experiencing. Be sure to include which AccuRev interface you were using (Web user interface, Java GUI, or CLI), any error messages you received, what you were doing when the error occurred, whether the problem is reproducible, and so on.
- A description of any attempts you have made to resolve the issue.
- A simple assessment of how the issue affects your organization.

1. Setting Up AccuRev®

This chapter describes how to perform initial data setup using either AccuRev's Quick Setup Wizard or equivalent manual procedures. It assumes you are familiar with basic AccuRev concepts such as depots and workspaces as described in the *AccuRev Concepts Manual*.

Before You Begin

Identify a directory that contains files that you want to place under version control and note its pathname (**c:\myfiles\src**, for example). The directory you choose can have subdirectories. The data in these directories is not modified during initial data setup -- AccuRev makes a copy of it.

Note that while you can place files under AccuRev control at any time, you might find it more convenient to do so during the initial data setup.

In addition, consider preparing a list of users for whom you want to create AccuRev accounts.

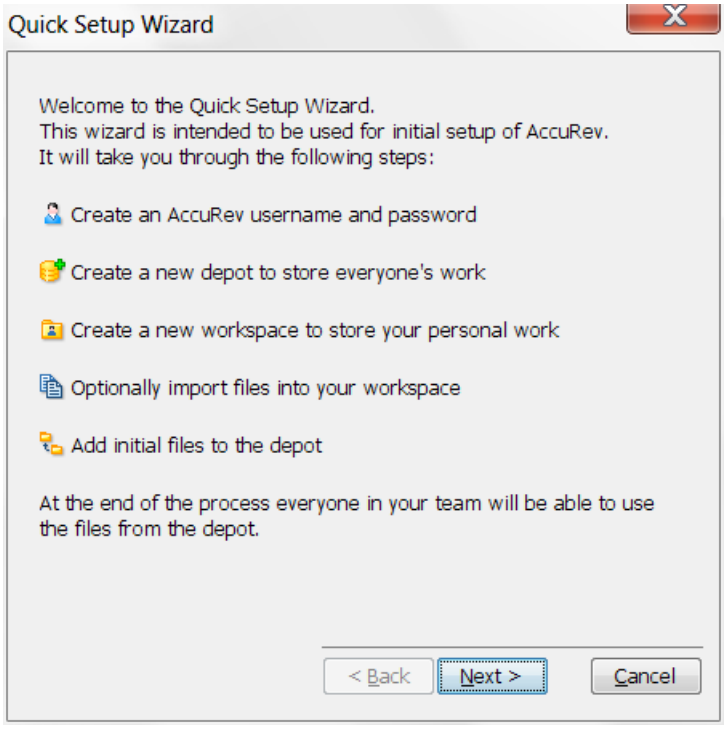
Using the Quick Setup Wizard

This section describes how to use the AccuRev Quick Setup Wizard to quickly get files under AccuRev control.

1. Start the AccuRev GUI by double-clicking the AccuRev shortcut () on your desktop.

   Alternative: In a Command Prompt window, enter **acgui**.

   If this is the first time you are starting AccuRev, the AccuRev Quick Setup Wizard appears.



If the Quick Setup Wizard does not appear, choose **Help** > **Quick Setup** from the AccuRev menu.

2. Review the quick setup process, then click the **Next** button to start the wizard.

   Note: Names for all AccuRev entities (usernames, depot names, workspace names, and so on) are case-sensitive. Usernames *john* and *JOHN* are considered to be different, for example.

3. Create an AccuRev username.
4. Create a new storage repository (depot).
5. Create a workspace for yourself, and copy data files to the workspace. Make these choices when prompted by the wizard:
   - Select **Yes** when asked "Do you have files that you would like to import into AccuRev?"
   - Select **Yes** when asked "Do you want to create a workspace in a new location …?"
   - When prompted "Where are the files stored …" specify the directory you chose in *Before You Begin* on page 1.
   - Accept AccuRev's suggestion for the location of the new workspace.
6. Add the files in your workspace to the depot by clicking **Promote** in the Promote window.

Manual Set-Up Procedures

Use the procedures in this section to perform you initial data set-up manually.

Create an AccuRev Username and Password

1. Select **Admin** > **Security** from the AccuRev main menu.
2. On the Users tab, click the **Add User** button ( ) to create a new user.
3. Enter a username, but do not enter the optional password.
4. Click **Ok**.
5. Select **Tools** > **Login** from the AccuRev main menu.
6. Enter your newly created username.
7. Click **Ok**.

Create a New Storage Depot

1. Select **File** > **New** > **Depot** from the AccuRev main menu.
2. Give the depot a name (example: **widget**).
3. Click **Ok** (and continue to the next operation).

Create a Workspace to Store Your Work

1. Click **Yes** ("Do you want to create a workspace for the depot?").
2. Click **Next** to select your newly created depot, and the stream that has the same name as the depot.
3. Choose a physical location for the workspace (example: **C:\ws\widget**) and click **Next** to accept the default name for the workspace.
4. Click **Next** to accept defaults for the new workspace.
5. Click **Finish**.

Import Files into Your Workspace

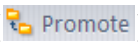
Use operating system commands to copy the directory and files you identified in *Before You Begin* on page 1 into the workspace directory on your file system. For example:
xcopy /E /Y c:\myfiles\src* c:\ws\widget)

Add the Files in Your Workspace to the Depot

If your newly created workspace is not already open in the AccuRev GUI, open it by clicking the ***Open Workspace*** *button on the toolbar (Alternative: Select* **File** > **Open Workspace** *from the AccuRev main menu).*

1. In the File Browser, select the Outgoing Changes mode (

).

   All the files listed in the Details pane should have a status of **(external)**.
2. Select all the files (using Ctrl+a, for example).
3. Click the **Promote** button (

) in the toolbar.

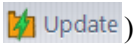
   Alternative: Right-click and choose **Promote** from the context menu.

   Tip: Promote automatically performs a Keep operation on any files that require it.
4. (*optional*) In the Promote dialog box, enter a comment string.

   Tip: Comments that you enter during Promote and other operations become part of the history of those files.
5. Click the **Promote** button.

   The files you just added to your workspace are now available to other users with workspaces that share the same parent stream. They are no longer visible in Outgoing Changes mode because their status has changed from **(external)** *to* **(backed)**. *Click the* ***Workspace Explorer*** *button to change modes; the files you just promoted are displayed here.*

Update Your Workspace

Click the **Update** button (

) to update your workspace with changes promoted by other users with workspaces that share the same parent stream.

Keeping your workspace up-to-date with frequent Update operations greatly improves workspace-search performance.

Next Steps

After completing the initial setup, your users can start using AccuRev as described in *Getting Started with AccuRev*.

2. 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. The repository consists of:

- A database called **accurev**, which contains:
  - A **site** schema, which contains the user registry, list of depots, list of workspaces, and other repository-wide information.
  - A schema for each depot, each of which contains depot-specific metadata and AccuWork issue data.
- The **site_slice** directory, which contains repository-wide AccuWork data, workflow configuration data, server preferences, and triggers
- The **depots** directory, which 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.
  - AccuWork schema, query, and change package configuration data.

When it starts, the AccuRev Server program determines the location of the **site_slice** and **depots** directories by looking at the settings in configuration file **acserver.cnf**. This file must reside in the same directory as the Server program (***accurev_server***) itself. See *Server Configuration File* on page 16 for more information.

Repository Access Permissions

The operating-system user identity of the AccuRev Server process must have full access to all the files and directories within the **storage** directory. For maximum security, this should be the *only* user identity with permission to access the repository.

If you create an **acadmin** AccuRev administrator account, as suggested in *Operating-System User Identity of the Server Processes* on page 13, 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 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 repository relies on the integrity of underlying software (the database software and 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 that you:

- 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 *Restoring the Repository* on page 8.)

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/Linux, 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 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.

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

Backing Up the Repository

You can use the procedures described in this section to back up both master and replica servers. Indeed, AccuRev recommends that you back up the replica servers in your environment on the same schedule you use for backing up your master. If you are backing up a replica server, be sure to also read *Backing up a Replica Server* on page 8.

Overview

Backing up the repository is a two-step process:

1. First, you use the accurev ***backup*** command to make a copy of the AccuRev metadata.

   During ***backup*** command execution, clients can continue to work. Only transactions that are complete at the time the ***backup*** command is invoked are included in the metadata backup. The backup may take a few seconds or a few minutes, depending on the amount of metadata on your system.

2. Next, you use a backup/restore tool to make a complete copy of the file storage area (the **storage** directory tree), without worrying about synchronization or time-skew. When you back up file storage, you can either stop the AccuRev Server or use a backup/restore tool that supports a live-backup scheme. See *Before You Begin* for more information.

   CAUTION: Do not execute the ***backup*** command while you are copying the file storage area. This can place incorrect data into the backup copy of the repository.

With the metadata backup and the copy of the file storage area, you can restore the repository to its state at the time you executed the ***backup*** command.

Before You Begin

- Read the section titled *A Word of Caution on Windows Zip Utilities* on page 11.
- Obtain a backup/restore tool if you don't already have one.

  Unless you plan to stop the AccuRev Server prior to backing up your file storage area, your backup/restore tool must be able to copy files that are currently "open" at the operating system level (that is, files that are in use by the AccuRev Server process). Your backup/restore tool should also have the ability to:

  - Preserve file timestamps.
  - Preserve file ownership and execute permissions.
  - Back up zero-length files. See *Server-Control Files* on page 22 for more information.
- Order matters. If you are using AccuRev's repository replication feature, AccuRev recommends that you back up the replica server first. See *Backing up a Replica Server* on page 8 for more information.

If you have any questions, contact Micro Focus technical support.

Backup Procedure

1. Back up the AccuRev metadata:

   ```
   accurev backup <backup-file-name>
   ```

   By default, the backup is stored in the **storage\site_slice\backup** directory.

   For more information on the ***backup*** command, refer to the CLI documentation.

2. If your backup/restore tool cannot copy files that are currently open at the operating system level, stop the AccuRev Server. (See *Controlling Server Operation* on page 21.)

3. Use your backup/restore tool to create a backup copy of the entire directory tree below the **storage** directory, except for the **db** directory. This backup can be all-at-once or piecemeal; for example, you can back up the **site_slice** directory and the subdirectories within the **depots** directory separately.

   Note: If your site slice is in a non-standard location (as specified by the SITE_SLICE_LOC setting — see *Server Configuration File* on page 16), 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 have scripts or triggers that are crucial to your usage of AccuRev, back up those as well.
5. If you stopped the AccuRev Server in Step 2, start it again. (See *Controlling Server Operation* on page 21.)

Backing up a Replica Server

As mentioned previously, you can use the procedure described in *Backing Up the Repository* to back up both master and replica servers. If you are backing up a replica server, note the following additional considerations:

- Back up all replica servers first, that is, before backing up the master server.
- If you have more than one replica server, you can back them up in any order.
- Start the back-up procedure for the master server as close as practical to the finish of the replica server backup -- a one-hour difference is a good interval, though you might want to adjust that over time. Keeping the interval as small as possible reduces the amount of content that will be required to update the replica servers after you restore the master server.
- On each replica server, make a copy of the **acserver.cnf** from the **bin** directory and save it with the backed-up data.

Out of Shared Memory Error

If you have a large number of depots, you may encounter error messages similar to the following:

```
pg_dump: WARNING:  out of shared memory
pg_dump: attempt to lock table <table name> failed: ERROR:  out of
shared memory
HINT:  You may need to increase max_locks_per_transaction.
```

You can address this issue in the following ways:

- perform a more selective backup using "-p" to back up only *some* depots
- increase "max_shared_locks_per_transaction" as suggested by the error message

To increase "max_shared_locks_per_transaction", edit the value of this variable in postgresql.conf using the following formula:

```
20 x (number_of_depots + 1) / max_connections
```

NOTE: Supporting the backup of more than 1,000 depots may also require that you increase the value of SHMMAX to more than 32MB.

Restoring the Repository

You can use the procedures described in this section to restore both master and replica servers. If you are restoring up a replica server, be sure to also read *Restoring a Replica* on page 9.

Restore Procedure

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. (See *Controlling Server Operation* on page 21.)

2. Restore the backup copy of the **storage** directory, using the backup/restore tool that you used to create it.

   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 16), or if some depots are in non-standard locations (perhaps moved with the ***chslice*** command), then your job in restoring the backup of the entire repository is more complicated than simply to restore the **storage** directory.

3. Restore any backups of scripts or triggers, using the backup/restore tool that you used to create them.

   Steps 2 and 3 will restore the backed-up data to standard locations on an existing AccuRev system. For more complex restore operations, see *Moving the AccuRev Server and Repository to Another Machine* on page 55.

4. Restore the backed-up metadata:

   ```
   maintain restore <backup-file-name> <db-admin>
   ```

   The ***maintain*** command will prompt you for the database admin password.

   For more information on the ***maintain restore*** command, refer to the CLI documentation.

   Note: the AccuRev Database Server must be running for this command to succeed.

5. Restart the AccuRev Server. (See *Controlling Server Operation* on page 21.)

   Note: suppose a particular depot's files were not backed up for several hours after the ***backup*** command was executed. Even if several new versions of file **gizmo.c** were created with the ***keep*** command during that interval, the backed-up AccuRev metadata will have no record of those transactions. But you can still retrieve a copy of any or all of those versions from the backup medium: it's in a container file in the **data** subdirectory of the depot directory.

Restoring a Replica

As mentioned previously, you can use the procedure described in *Restoring the Repository* to restore both master and replica servers. If you are restoring up a replica server, note the following additional considerations:

- Restore the master server first.
- Synchronized the replica servers using the ***replica sync*** command. See *Synchronizing a Replica Manually* on page 51 for more information.
- After restoring the backed-up metadata, refer to the backed-up **acserver.cnf** file to ensure that the MASTER_SERVER, PORT, LOCAL_SERVER, and LOCAL_PORT settings are correct for your replication environment.

Archiving Portions of the Repository

The container files that store the contents of individual file versions can be moved to offline storage, in order to save online storage space for the repository. For details, see *Archiving of Version Container Files* on page 37.

Moving a Workspace or Reference Tree

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

First, make sure that no user or script process is currently using the workspace or reference tree. Move the physical contents of the workspace tree or reference tree with a backup/restore tool (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, make sure that no user or script process is currently using the depot. (To guarantee this, you may wish to stop the AccuRev Server.) Move the physical contents of the depot with a backup/restore tool (e.g. ***tar***, ***zip***, ***xcopy /s***). Then, let AccuRev know about the move:

```
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 119.

Moving the db and site_slice Directories

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

The site_slice and db directories must be under the same parent directory, so if you move one, you must use the other as described in the following procedure.

1. Make sure that no user or script process is using the depot, and stop the AccuRev and database servers:

   ```
   acserverctl dbstop
   ```

2. Move the `site_slice` and `db` directories to a new parent directory location. Make sure that both directories are moved to the same parent directory.
3. Open `acserver.cnf` in a text editor and change the `SITE_SLICE_LOC` definition to point to the new site_slice directory location.
4. `cd` into the db directory and open the `postgresql.conf` file in an editor. Change the directory path that points to where the `site_slice/logs` directory is now located.
5. Restart the database server:

   ```
   acserverctl dbstart
   ```

6. Register the `acserver.cnf` changes with the database:

   ```
   maintain server_properties update
   ```

7. Restart the AccuRev server process:

   ```
   acserverctl start
   ```

8. Verify that the db directory path is correct:

UNIX/Linux: From the command line, enter `ps -ef | grep postgresql` and verify that the db directory path listed after the "`-D`" option is correct.

Windows: In the `services.msc` utility in Windows, right-click on the "`Accurev DB Server`" process, select "`Properties`" and verify that the directory path listed after the "`-D`" option is correct.

A Word of Caution on Windows Zip Utilities

Be careful when using WinZip® or PKZIP® on a Windows machine to move a workspace, reference tree, or depot. You may want to use ***tar*** on a UNIX/Linux 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.
- Zip utilities typically refuse to copy files that are open at the operating system level. Typically, you can work around this limitation by stopping the ***accurev_server*** program, but this defeats AccuRev's "live backup" feature.

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. An example directory layout would be:

```
<ac-install>
    bin
    storage
      db
      depots
            project1
            project2
            project3
      site_slice
```

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**.)

Database

The database stores all the versions of a set of files and directories, as well as AccuWork schema, query, and change package configuration data.

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. 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**.

You can relocate a depot's file storage area onto other disk partitions or even onto remote disks. 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.

3. The AccuRev Server

The AccuRev repository is managed by a single program, the AccuRev Server (***accurev_server***). This program communicates with the AccuRev Database Server. Both of these must be started prior to running any AccuRev client commands. No user should attempt to work directly with the repository, unless it is an emergency. In this case, please contact AccuRev Support Services.

Operating-System User Identity of the Server Processes

Like all processes, the AccuRev Server and AccuRev Database Server processes have an operating-system user identity. It should be a unique user identity, not used by any other program. This helps to ensure that no other user or process has access to the repository.

UNIX/Linux Identity

> CAUTION: The AccuRev Database Server cannot be run as the **root** user. In addition, do not attempt to run the AccuRev Server as **root**. Some user-supplied trigger scripts run under the operating-system identity of the AccuRev Server, which poses a significant security risk. (See *Trigger Script Execution and User Identities* on page 100.)

We suggest that you create an operating-system user named **acserver**, belonging to a group named **acgroup**. (Any similar names will do.) Only the AccuRev Server should run as **acserver**.

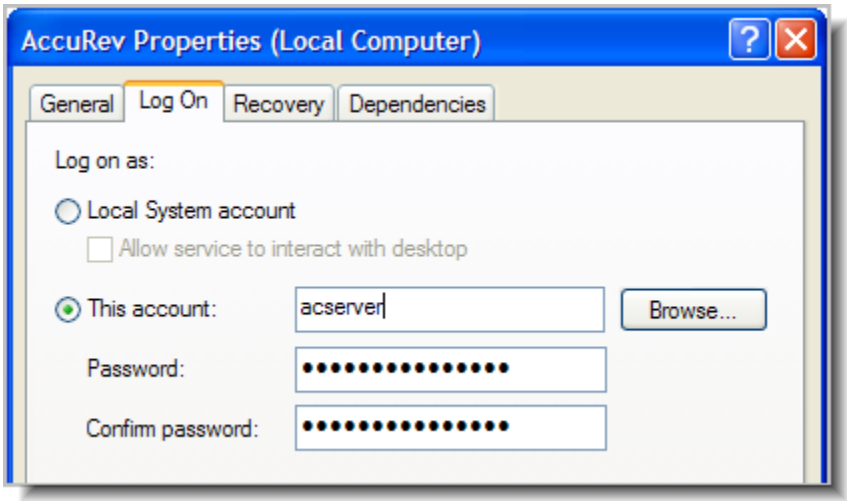
For emergency "manual" access to the repository, you can create another user identity — say, **acadmin** — and place that user in the same group, **acgroup**. You can configure UNIX/Linux-level auditing and place other appropriate controls on this account; this leaves the **acserver** account (and thus, the AccuRev Server process) unencumbered by such controls.

Configure the AccuRev Server to run with the **acserver**/**acgroup** identity by placing these names in the server configuration file, **acserver.cnf**. See *UNIX/Linux: Controlling the Server's Operating-System User Identity* on page 16.

Windows Identity

The AccuRev Server and AccuRev Database Server run as Windows services. By default, these services run as the built-in local user named **System**. This user identity must have access to the AccuRev executables (**bin**) directory and to the repository. See *Repository Access Permissions* on page 5.

You can use the **Services** control panel to configure the services to run under another identity ("account").



AccuRev User Identity of the Server Process

In addition to its user identity at the operating system level, the AccuRev Server process sometimes needs an AccuRev username (principal-name) identity:

- When it executes a server-side trigger script that invokes AccuRev client commands, such as ***annotate*** or ***promote***.
- When it performs a synchronization with the master repository — explicit or implicit ***replica sync*** command. This applies only if the AccuRev Server is managing a replica repository.

If either of these situations applies to the AccuRev Server that you are administering, you must take steps to establish a valid AccuRev username for the AccuRev Server. The AccuRev username need not be special or reserved. Just make sure that any security controls — ACL permissions and/or ***server_admin_trig*** script — are configured to allow that particular AccuRev username to perform the required operations. See *AccuRev Security Overview* on page 57.

> **Note**: for security reasons, we recommend that the operating-system identity of the AccuRev Server process (for example, **acserver**) should *not* also be an AccuRev username.

With the "AccuRev login" user-authentication scheme, a session file establishes the AccuRev user identity of a process. Create a long-lived session file for the AccuRev Server's operating-system identity as follows:

1. Access the command line.
2. Set environment variable ACCUREV_HOME to the home directory of the operating system user that the AccuRev Server runs as (for example, **acserver**).

   If the AccuRev Server is running as the local **System** account on a Windows machine, the home directory is **C:**.
3. Create a long-lived session file for the AccuRev username that the AccuRev Server will use:

```
accurev login -n john
Password: ********
```

Note: if you are creating a session file on a replica server machine, to be used for communicating with the AccuRev Server process on the master server machine, direct the ***login*** command to the master server machine. For example:

```
accurev login -n -H bingo_master:5050 john
Password: ********
```

This session file will be valid indefinitely, thanks to the ***–n*** option.

Starting the AccuRev Server

The following sections describe how to start the AccuRev Server, either automatically at operating system bootstrap time, or manually at a command prompt. (You can also perform a "manual" startup with a UNIX/Linux 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/Linux systems, an "rc" or "init.d" startup script starts the ***accurev_server*** program. The AccuRev installation program does not install the startup script automatically. You must customize and install the sample startup script, named **accurev**, located in the **extras/unix** subdirectory of the AccuRev installation directory. See the **README** file in that subdirectory for complete instructions.

Note: Red Hat Linux will not run an rc.d script at shutdown unless it finds a lock file named `/var/lock/subsys/accurev`. Without this lock file, the "acserverctl stop" comand will not run, and the log file will not record a clean shutdown.

On Windows systems, the AccuRev installation program automatically configures the ***accurev_server.exe*** program as a Windows service. Use the ***Services*** control panel to stop, start, and configure the Server program.

Starting the Server Manually

The AccuRev Server may also be started manually:

- **Windows systems**: If you've changed the startup type of the **AccuRev** service to "Manual", you can start the service from the ***Services*** control panel. Alternatively, run the ***server_start.bat*** script, located in the AccuRev executables (**bin**) directory.
- **UNIX/Linux systems**: Start the Server with the ***acserverctl*** utility:

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

  The Server will run with your operating-system user identity. (Make sure that the server configuration file's USER and GROUP settings are commented out. See *UNIX/Linux: Controlling the Server's Operating-System User Identity* on page 16.)

Server Configuration File

When it starts, the AccuRev 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.

> *Important! After editing this file, you must restart the AccuRev server. If you have edited the values for SITE_SLICE_LOC, MASTER_SERVER, or PORT, you **must** also run the **maintain server_properties update** command before the new values will take effect. See The 'maintain' Utility on page 115 for more information.*

Here are some sample **acserver.cnf** settings:

```
MASTER_SERVER = accurev_server_machine.company.com
PORT = 5050
SITE_SLICE_LOC = /partition0/site_slice
DEPOTS_DEFAULT = C:\Program Files\AccuRev\storage\depots
```

> *Important! The white space surrounding the equals sign (=) in configuration files is mandatory.*

The MASTER_SERVER name should be the fully-qualified name of the server machine, 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 PORT setting contains the port that the AccuRev Server is available on.

The SITE_SLICE_LOC setting points to the directory that the Server uses for storing repository-wide AccuWork data, workflow configuration data, server preferences, and triggers. This directory:

- Must be owned by the operating-system account that the Server runs as (for example, **acserver**).
- Must be physically located on the server machine. The SITE_SLICE_LOC location must not be within a remotely mounted file system (UNIX/Linux) or within a shared directory (Windows) on a remote machine.

The DEPOTS_DEFAULT setting points to the directory that the Server uses for new depots, for storing the files created when a permanent copy of a version is created with the ***keep*** command. Note that if you move an existing depot, this variable is not used — you must tell AccuRev about the new location with ***chslice***.

UNIX/Linux: Controlling the Server's Operating-System User Identity

The following specifications determine the user identity and group membership of the operating system process in which the AccuRev Server runs:

```
USER = <user-name>
GROUP = <group-name>
```

When the AccuRev Server is started automatically (usually at system boot), it changes its user identity and group membership according to the USER and GROUP settings in the **acserver.cnf** file. (See *Controlling Server Operation* on page 21.)

With these settings commented out, the AccuRev Server runs under the identity of the user who started it.

UNIX/Linux: Setting the Server's Home Directory

In addition to having a user identity and group membership, the AccuRev Server has an AccuRev home directory. This directory is used for a variety of purposes — for example, to store a login session file created by a ***server_admin_trig*** trigger script.

By default, the AccuRev home directory is the same as the operating-system home directory, as indicated by the environment variable HOME. It's a best practice to override the HOME value by setting the value of environment variable ACCUREV_HOME. If the AccuRev Server is started automatically at system startup time by a script in the "rc" or "init.d" directory, the most logical place to set the AccuRev home directory is in this startup script:

```
export ACCUREV_HOME=/users/acserver
```

Controlling Login Session Longevity

A successful user login creates a session that by default expires 4 hours (240 minutes) after the last AccuRev command is executed. You can change this behavior by creating or modifying this **acserver.cnf** setting:

```
SESSION_TIMEOUT = <number-of-minutes>
```

On UNIX/Linux systems, a user can control the timeout for an individual session by setting environment variable SESSION_TIMEOUT before logging in. For example, to set a 15-minute timeout interval for a single session:

```
export SESSION_TIMEOUT=15
accurev login derek
```

Non-expiring login sessions

The following setting in **acserver.cnf** causes user sessions never to expire:

```
SESSION_TIMEOUT = 0
```

No matter what the setting, users can create non-expiring sessions with ***login –n***.

Controlling Multithreading of the AccuRev Server

The AccuRev Server is a multi-threaded program, architected to support a maximum of 256 concurrent threads. To conserve system resources, you can specify a lower maximum in the **acserver.cnf** file:

```
MAX_THREADS = 25
```

As it's running, the AccuRev Server may reduce the maximum even further than the specified MAX_THREADS level, depending on the available computing resources.

Configuring Access to the Database

The AccuRev Server (as of Version 5.0) uses the third-party database PostgreSQL. The settings listed below are configured at installation.

```
DB_DRIVER = Postgres
DB_CONNECT = localhost:5075@accurev
DB_USER = acserver
DB_PASS = e8c5ed8f07bfaf44d2f2eebc215e3cf3
DB_HOME = C:\Program Files\AccuRev\postgresql
```

- DB_DRIVER – Identifies the third-party database used by AccuRev. This setting should not be changed.
- DB_CONNECT – Identifies the database server, port, and name used by AccuRev. This setting should only be changed if you change the port that the AccuRev database server uses.
- DB_USER – Identifies the database user AccuRev uses to connect to the third-party database. This setting can be changed using the ***maintain setcnf*** command.
- DB_PASS – Identifies the (encrypted) password for DB_USER. This setting can be changed using the ***maintain setcnf*** command.
- DB_HOME – Identifies the location of the binaries and libraries used by the third-party database. This setting is used by the ***accurev backup*** command, and should not be changed.

Server Logging

The AccuRev Server maintains a log file, **acserver.log**, in subdirectory **logs** of the **site_slice** directory. Each log message includes a timestamp, the AccuRev username that invoked a client command, and the IP address of the client machine.

AccuRev also supports client-side logging. See *Logging* on page 417 of the *AccuRev Online Help Guide* for more information.

Time zone offset

As of AccuRev V5.4, the format of **acserver.log** entries has been expanded to include a time zone offset. For example:

```
2011/12/20 14:33:41.329-05:00 Server locale is C
```

where "-05:00" is the ISO-8601 TZ offset with respect to GMT.

This enables you or AccuRev Support to correlate logs from masters, replicas, and clients spread across different timezones.

Logging Levels

Logging information can be preserved at various levels of detail, as specified in **acserver.cnf**:

```
# log level 2 or 3 is recommended by AccuRev support team
LOG_LEVEL = 2
```

UNIX/Linux: Log File Rotation

On UNIX/Linux server machines, 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.

Controlling Server Log Verbosity

The verbosity of the server logs is controlled by the LOG_LEVEL entry in the **acserver.cnf** file:

```
LOG_LEVEL = 3    #enable the highest (most verbose) level of logging
```

At log level 1, each execution thread produces one line in the log. At log level 2, each execution thread can produce multiple log lines. Log level 3 essentially combines levels 1 and 2. At higher log levels, some of the messages detail the work of server subtasks.

Verbose Server Logging

The Server is a multi-threaded program, so it can handle several client commands concurrently. A typical AccuRev client command causes the AccuRev Server to execute a set of server subtasks. For each client command, the Server's "master thread" creates a new "worker thread" to perform the set of subtasks for that particular command. When the worker thread has performed all the subtasks, it exits. When the LOG_LEVEL is 2 or 3, the log messages indicate many of the details of server subtask execution.

For example, a single ***update*** command can generate a set of log messages like this:

```
2008/02/09 16:24  connection 1076 on 00000E98 cache 0 started
2008/02/09 16:24  1076 mary         *update       00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         cur_wspace    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         ws_type       00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         stream_top    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         check_time    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         update        00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         end           00000E98 1.2.3.101
2008/02/09 16:24  connection 1076 on 00000E98 success 0.563 0 0 0 1.2.3.101 mary
```

These messages may or may not appear on consecutive lines of the log file. If multiple client commands are being executed concurrently by different worker threads, the log messages that the threads generate will be interleaved in the log file.

Let's examine each message in the above example:

```
2008/02/09 16:24  connection 1076 on 00000E98 cache 0 started
```

The first message is generated at the time (`2008/02/09 16:24:20`) a client request is accepted by the Server's master thread. This is `connection 1076` between the client and the server). The master thread creates a new worker thread (worker thread-ID `00000E98`) and hands the request off to it.

```
2008/02/09 16:24:20   1076 mary         *update       00000E98 1.2.3.101
```

This message indicates the user who invoked the command (`mary`), the name of the command, marked with an asterisk (`*update`), and the IP address of the client machine (`1.2.3.101`).

```
2008/02/09 16:24  1076 mary         cur_wspace    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         ws_type       00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         stream_top    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         check_time    00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         update        00000E98 1.2.3.101
2008/02/09 16:24  1076 mary         end           00000E98 1.2.3.101
```

Each time the worker thread begins a particular subtask, it sends a message to the log. In the lines shown above, the client command is implemented through server subtasks **cur_wspace**, **ws_type**,

stream_top, **check_time**, **update**, and **end**. (The last subtask is always named **end**.) Each message also includes the username, thread-ID, and client IP address.

```
... connection 1076 on 00000E98 success 0.563 0 0 0 1.2.3.101 mary
```

The last message is generated by the worker thread after it has completed all subtasks and is about to exit. (If the LOG_LEVEL is 1, this is the *only* message generated for each client command.) In addition to the username, thread-ID, and client IP address data also included in the preceding messages, this message reports summary measures, listed in boldface above:

- ***success*** / ***failure*** (`success` in the example above): The overall result of the attempt to execute the client command.
- ***run time*** (`0.563`): the total time, in seconds, that the worker thread took to process the entire client command.
- ***last check*** (`0`): the time, in seconds, elapsed since last progress update from worker thread. In a success message, this value is 0. In a failure message, this value is non-zero.
- ***time delta*** (`0`): the time difference between the clocks on the client and server machines.
- ***exit status*** (`0`): the exit code for thread: 0 = success, non-zero = error.

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 most robust software systems are occasionally compromised; the AccuRev Server can be brought down by a bad disk sector or an administrator's mistaken command.

The reliability of the AccuRev Server is further enhanced by a companion program, called the "Watchdog", which 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***.

Every 10 seconds, the Watchdog sends a simple command to the Server. If the Watchdog detects that the Server is not responding or is not functioning properly, the Watchdog restarts the Server. If the Watchdog detects five such failures within a three-minute timespan, it doesn't restart the Server; such a situation indicates the need for server reconfiguration or investigation by the AccuRev support team. (If ACCUREV_WATCHDOG_FAST_FAIL_DISABLE is set in the Watchdog's environment, it keeps trying to restart the Server indefinitely.)

For the most part, the functioning of the Watchdog process 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.

Tools that control the execution of the Server and Watchdog are in described in section *Controlling Server Operation* on page 21.

Watchdog Logging

The Watchdog maintains a simple log file, **acwatchdog.log**, in subdirectory **logs** of the **site_slice** directory. On UNIX/Linux server machines, the Watchdog log file is rotated similarly to the Server log file.

Controlling Server Operation

AccuRev includes facilities for controlling the operation of the AccuRev Server, the Watchdog, and the AccuRev Database Server. The user interface varies by platform:

- UNIX/Linux: the ***acserverctl*** command-line utility
- Windows: the ***Services*** control panel

UNIX/Linux: 'acserverctl' Utility

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

The ***acserverctl*** script also controls the AccuRev Database Server. The AccuRev Server process is dependent upon the Database Server process; if it is not running the AccuRev Server will not start.

> Note: by default, ***acserverctl*** assumes 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:

dbstart

Start the Database Server process.

dbstop

Tell the Server, the Watchdog, and the Database Server processes to stop gracefully.

start

Start the Database Server (if it is not running), then start the Server and Watchdog processes.

stop

Tell the Server and Watchdog processes to stop gracefully.

dbstatus

Report whether the Database Server is running or not.

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.

dbrestart

Stop the Server and Watchdog processes, then restart the Database Server, the Server, and the Watchdog processes.

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' Control Panel

If the AccuRev Server and AccuRev Database Server are running as services on a Windows machine, you can control their operation from the ***Services*** control panel. In some versions of Windows, this control panel is in a subdirectory called ***Administrative Tools***.

The context (right-click) menu available for the services includes these commands:

start
stop
pause
resume
restart

For descriptions of these commands, see *UNIX/Linux: 'acserverctl' Utility* above. The ***restart*** command brings down both the Server and the Watchdog, by performing a ***stop*** followed by a ***start***. For the AccuRev Database Server, the ***start***, ***stop***, and ***restart*** commands perform the equivalent of the ***dbstart***, ***dbstop***, and ***dbrestart*** commands in ***acserverctl***.

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.

acserver.command.taskkill

Use this capability only as a last resort. Typically, it is preferable to restart the entire AccuRev Server (which allows in-progress tasks to complete), rather than terminating just one of its worker threads.

To terminate a particular worker thread:

- Go to the **site_slice** directory.
- Determine the ID number of the worker thread by examining the thread-status table (***Tools*** **>** ***Server Tasks*** command).
- Place the thread's ID number (e.g. **42**) in the flag file:

  ```
  echo 42 > tempfile
  mv tempfile acserver.command.taskkill
  ```

  Using the ***mv*** (or ***move***) command instead of the ***echo*** command to create the **taskkill** file prevents a race condition that might cause the server to see **taskkill** as an empty file.

Important! After terminating a thread, restart the AccuRev Server as soon as possible. This minimizes the likelihood that terminating the thread will cause a memory resource leak in the Server process, impairing overall system performance.

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.)

Checking the Number of File Descriptors

To check the number of file descriptors in use at any given time by the AccuRev Server:

- *(UNIX/Linux)* Create a file (**anyname.xml**) containing the following line:

  ```
  <serverInfo/>
  ```

  Then run this AccuRev command:

  ```
  accurev xml -l anyname.xml
  ```

- *(Windows)* Use Microsoft's ***Handle*** utility on the AccuRev Server machine to monitor the open file handles:

  ```
  handle.exe -p accurev_server
  ```

  (See the "Process Utilities" link at ***http://www.sysinternals.com***.)

Changing the Per-Process Open File Descriptor Limit

> **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.

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

Regardless of operating system, 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.

Windows

The default limit is 2048 file descriptors and cannot be changed.

Linux

You can review and modify file descriptor limits at both the system and process level as described in the following procedures.

System-wide Limits.

1. Check the current system-wide limit by issuing:

   ```
   % cat /proc/sys/fs/file-max
   ```

2. If you need to increase the system-wide limit (to 100000, for example), as **root**, issue:

```
# /sbin/sysctl -w fs.file-max=100000
```

Per-Process Limits.

1. Check the per-process limit by issuing (in bash):

```
% ulimit -n
```

2. To increase the per-process limit (to 4096, for example) for the current shell and its sub-processes:

```
% ulimit -n 4096
```

3. To increase the per-process limit (to 4096, for example) *permanently* for user 'acserver':
   - Make sure the following line exists in **/etc/pam.d/login**:

     ```
     session required pam_limits.so
     ```
   - Add the following lines to **/etc/security/limits.conf**:

     ```
     acserver soft nofile 4096
     ```
4. Open a new shell and verify the limit before restarting the AccuRev Server:

```
% ulimit -n

% acserverctl stop

% rm acserver.handle.limit

% acserverctl start
```

AccuRev Server Performance

This section is targeted to administrators planning a new enterprise deployment of AccuRev configuration management software, as well as to those planning the growth of an existing deployment. It will also be useful for administrators whose existing deployment is experiencing lower-than-expected performance.

When evaluating AccuRev performance, it is important to use the actual hardware that you will be using in production, or hardware of the same class.

AccuRev Architecture Overview

AccuRev uses a standard client-server architecture:

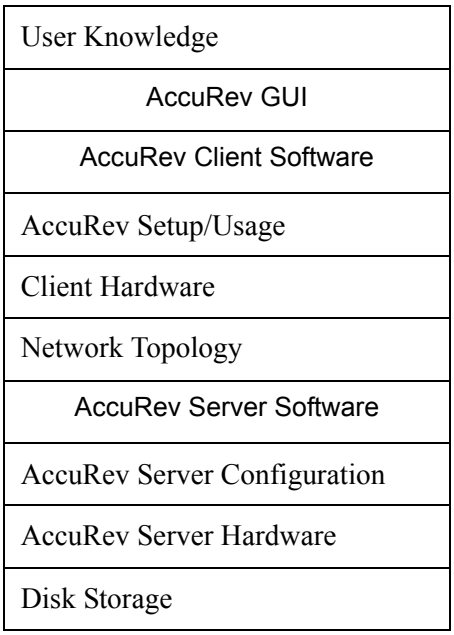
- The **AccuRev Server** software runs on one machine, as a multi-threaded operating system process, named **accurev_server**. The Server manages the AccuRev repository, where all data under AccuRev management is stored.
- **AccuRev Client** software runs on each AccuRev user's machine. The command-line interface is implemented as a single program, **accurev**, which executes a single command and then exits. The graphical user interface is implemented as two programs: the user-visible Java application **acgui** is long-lived; when the user invokes a command through the GUI, ***acgui*** runs the background application **accurevw**, which communicates with the Server and then exits.
- User workspaces, where all development work is performed, are typically stored on users' machines. (But they can be stored on network file servers, or even on the AccuRev Server machine.)

- Each Client submits requests to the Server through a TCP/IP network. AccuRev Clients never communicate directly with each other.

As the above description indicates, the AccuRev Server is the heart of the AccuRev system. The sizing and configuration of the AccuRev Server machine significantly impacts the performance perceived by users. We highly recommend that you first read *The AccuRev Data Repository* of the *AccuRev Concepts Manual*, along with the rest of this manual. We also recommend that you attend training courses offered by AccuRev, Inc.

Key Factors Affecting AccuRev Performance

This diagram illustrates the AccuRev "performance stack". The highlighted components are the ones that are under customer control — each component can influence the user's experience of AccuRev either positively or negatively. In particular, a problem at any level can severely degrade the user's experience of AccuRev performance. We strongly recommend that a top-to-bottom assessment of all of these components be performed prior to deploying AccuRev on the enterprise level.



The following sections discuss the aspects of an AccuRev deployment that can be "performance tuned" through knowledgeable choices.

Resource Utilization and the Client-Server Architecture

The AccuRev Server is a multi-threaded application. Each individual AccuRev operation, such as executing a ***History*** command, uses only one thread, which runs on one CPU. When multiple client command requests arrive at the AccuRev Server at (approximately) the same time, these commands are executed through normal operating system multitasking. On a multiple-CPU machine, the threads are shared across all available CPUs.

AccuRev is architected to minimize client-server interaction and network bandwidth. Operations such as viewing files, editing files, and performing builds do not require any interaction with AccuRev. Thus, even during periods of intensive development, each user performs actions that require interaction with AccuRev relatively rarely. It is only when a user wishes to add a new file to a depot, rename a file, get the history of a file, get the latest version of a file, or similar operations that interaction with AccuRev will occur. Even then, these are short-running operations that put little load on the AccuRev Server software.

Experience indicates that, on average, a user will actually "hit" the AccuRev Server for less than 5 minutes per day. Typically, Server utilization is not uniform throughout the day, but tends to form peaks and valleys. For example, you might find that multiple users ***Keep*** their changes just before the nightly build is started, or that users tend to ***Update*** their workspaces at approximately the same time (e.g. when they arrive at work, or just after lunch).

Customer data has consistently shown that "peak load" rarely exceeds 15%. That is, you are unlikely to experience a situation in which more than 15% of the active AccuRev users have overlapping requests being processed by the AccuRev Server. (You can determine the number of active users by examining the Server log. Or you can simply assume that all the users listed by the command ***accurev show users*** are active.) The most common command request during peak load periods is ***Update***. This also is the AccuRev command that requires that most processor, disk, and network resources.

AccuRev Server Machine Usage: Software

AccuRev uses a data repository located on the AccuRev server machine. Any software running on the AccuRev Server machine that is not specifically supporting the AccuRev installation runs the risk of contending for the resources that AccuRev needs. For instance, if the machine is also acting as a file server for builds or other purposes, then users may find that AccuRev seems slow, even when there are no other AccuRev users currently making requests.

For optimal performance, the server machine should be completely dedicated to AccuRev usage. NFS, Samba, httpd, print queues, and other similar services should all be disabled. Only AccuRev administrators should use the system, and then only to service the AccuRev machine or software.

Reference trees and/or build workspaces should be located on client (user) machines or on dedicated build machines, not on the AccuRev Server machine. Software builds should not use the AccuRev Server machine's processor.

AccuRev Server Machine Networking

We strongly recommended that all build machines be located on the same subnet as the AccuRev Server machine, and that this subnet not include any other machines. The build subnet should use gigabit Ethernet. It is important that all of the equipment in the subnet be locked down to gigabit speeds and not be allowed to autonegotiate. Autonegotiation typically causes the subnet to operate, eventually, at the lowest supported speed.

AccuRev's network bandwidth usage is similar to that of ***ftp***. For instance, a ***keep*** command on 100 files needs to send a list of 100 files to the server, in addition to the contents of those files. AccuRev is transaction-based; it often works on groups of files in a single transaction. Thus, the full list of file names and contents is sent to the server in one burst. This minimizes changes in the direction of data transfer, effectively removing the impact of any network latency.

Secure Networking

There are three ways to implement secure communication between two AccuRev sites:

- Most efficient is a dedicated connection between the two sites. Having a dedicated connection guarantees privacy and security. There is no performance penalty on a dedicated line.
- Next most efficient is to tunnel via a secure socket (SSH). The encryption has very little impact on bandwidth. Most SSH software offers compression, which can actually increase the effective bandwidth of the connection.
- Least efficient is VPN. Depending on the implementation, VPN overhead can consume up to 20% of the available bandwidth.

Using Multiple AccuRev Servers

The simplest way to use AccuRev is to have a single repository, managed by a single AccuRev Server. The server process runs on a particular machine, and listens for client requests on a particular IP port. But it's possible for an organization to have multiple servers running concurrently on different machines. Each server manages a separate repository, located on the server's own machine; the servers and their repositories are completely independent (and unaware) of each other.

There are no special considerations, and no special procedures, for setting up multiple machines to run AccuRev Servers. You can install a server on as many hosts as desired (subject to license restrictions)

within a local area or wide area network. The fact that AccuRev is installed on machines that are networked together is irrelevant; each installation is independent of all the others.

Setting Up Client Machines

Installing AccuRev on a machine creates a client configuration file, **acclient.cnf**, in the AccuRev executables directory. (That's subdirectory **bin** of the location designated as the "Install Folder" or "Install Directory" during installation.) The configuration file specifies the network location of the IP port on which an instance of the AccuRev Server is listening for client requests. For example:

```
SERVERS = jupiter:4079
```

If more than one server has already been set up, you'll need to specify one of them during the client installation. Check with a system administrator, or look at file **acserver.cnf** in the AccuRev executables directory on the machine where the server is installed, For example, the server machine's **acserver.cnf** file might contain:

```
MASTER_SERVER = jupiter
PORT = 4079
...
```

You can modify the client configuration file at any time to identify additional AccuRev servers. For example, in a network where AccuRev servers are running on machines **jupiter**, **venus**, and **pluto**, each client machine's **acclient.cnf** file might look like this:

```
SERVERS = jupiter:4079
SERVERS = venus:5050
SERVERS = pluto:6678
```

Each server's IP port must be listed on a separate line (even though the keyword is "SERVERS", not "SERVER"). The file must not contain any empty lines; be sure to check the end of the file.

The AccuRev GUI allows you to configure the contents of the **acclient.cnf** file via the ***Tools > Login*** dialog. See *Using the Login Dialog* on page 6 for more information.

In what order should the servers be listed? For users of the AccuRev CLI (***accurev***), the order is important: by default, each AccuRev CLI command is directed to the server listed on the *first* line of the client configuration file. This implies that if you want to change the "active server" in the CLI, you need to rearrange the lines in your machine's **acclient.cnf** file. This is quite doable, but cumbersome; the sections below describe a better way for CLI users to work with multiple AccuRev servers. For users of the AccuRev GUI, it doesn't matter. The GUI's ***Tools > Login*** command revises the **acclient.cnf** file, moving the line for the new active server to the beginning of the file.

Tip: The ACCUREV_SERVER environment variable influences which AccuRev Server the AccuRev GUI interacts with at startup time. See *ENV_VARS* on page 264 of the *AccuRev CLI User's Guide* for more information.

Workspaces and Servers

Each AccuRev workspace is associated with a particular AccuRev server: the workspace is attached to a particular stream, which belongs to a particular depot, which is managed by a particular server. When you execute a ***mkws*** command to create a new workspace, the command is directed to the server that is listed first in the machine's **acclient.cnf** file. (You must use the ***–s*** option to specify a backing stream for the new workspace; this stream must belong to one of the depots managed by that particular server.) To access

multiple workspaces on multiple servers, create a workspace configuration file as described in *Working with Multiple Repositories* on page 4 of the *CLI User's Guide*.

> **Note**: It is more precise to describe the workspace's association as being with a particular AccuRev *repository*, rather than with a particular AccuRev *server*. This workspace-to-repository association is permanent: AccuRev has no concept of associating an existing workspace with a different repository (or even with a different depot in the same repository). On the other hand, many of the details can change: you can rename a workspace; you can move a workspace to a different location in the file system or to a different machine; you can move a repository to a different machine.

Specifying a Server By Using the –H Option

For most commands in the ***accurev*** CLI program, you can specify the AccuRev server/repository to target on the command line, using the ***–H*** option:

```
accurev show -H pluto:6678 users
```

The ***–H*** option follows the command name — in this example, ***show*** — not the program name, ***accurev***. The hostname/port-number argument to this option has the same form as in the **acclient.cnf** file.

This mechanism bypasses the **acclient.cnf** file, though the file must still exist. It does *not* override a specification in the **wspaces** file.

Configuring the Web User Interface (Web UI)

The optional Web UI can be used to access AccuRev functionality through a Web UI server. However, you can also access the Web UI from the AccuRev Java client to provide access to AccuWork features. Configuring the Web UI on the AccuRev server is also required if you use the AccuRev integration with Eclipse.

To configure the AccuRev Web UI for use with:

- AccuWork
- AccuRev integrations

use the following procedure:

1. Install the Web UI server as described in the AccuRev *Installation and Release Notes*, and make note of the host and port where it is installed.
2. On the AccuRev server, create a file named `settings.xml` in the directory ***`ac_install`***`/storage/site_slice/dispatch/config`
3. This file should contain the following entry:

   ```
   <settings>
   <webui url="http://yourwebhost:yourwebport/accurev"/>
   </settings>
   ```

   where you must substitute valid values for your site for *`yourwebhost:yourwebport`*.

   This entry allows the AccuRev server to kick off a web browser and have it point to the correct web server.
4. Optionally, you can also add the following line within the `<settings>` tags if you wish to enforce that all of your users use the AccuRev Web UI client instead of the traditional Java client when working with AccuWork issues:

   ```
   <useWebForIssues value="true"/>
   ```

Note that this setting is case-sensitive: make sure that `useWebForIssues` is capitalized exactly as shown here.

This setting will override the preferences.xml files for individual users, and the “Enable Issue Preview” and “Open Issues In Web Only” options in their AccuRev Preferences dialog will be disabled.

4. System Clock Synchronization

Time plays a fundamental role in AccuRev's architecture and in its day-to-day operations. Some examples: each transaction is recorded in the database at a particular time; a snapshot reconstructs the state of a stream at an arbitrary time; the ***stat*** command and the AccuRev GUI use timestamps to optimize the lookup of modified files within a workspace.

AccuRev is a networked product: programs execute on one or more server machines and multiple client machines. In a perfect world, the system clocks on all these machines would always be perfectly synchronized. The data items on the (master) server machine (say, versions created by ***keep*** commands) and corresponding data items on a client machine (the files that were kept) would have timestamps that are consistent with each other.

Software systems do exist that synchronize all the machines in a network to within milliseconds. If your organization uses such a system, then you don't need to read any further in this chapter!

Most software development organizations don't have — and don't need — synchronization at the millisecond level. AccuRev defines a 5-second tolerance as "good enough for software configuration management". This chapter describes AccuRev's own facilities for detecting system-clock discrepancies, along with other related facilities commonly available on Windows and UNIX/Linux systems.

Detecting System Clock Discrepancies -- Timewarp

A timewarp (clock skew) occurs when the discrepancy between the system clocks on a client machine and a server machine exceeds the allowable tolerance of 5 seconds. A timewarp can also occur in a replication environment, when the clocks on the master server and a replica server differ by more than 5 seconds. (See *Replication of the AccuRev Repository* on page 43.)

Timewarp problems typically occur during initial system setup and during time zone adjustments. For example, the change from Eastern Standard Time to Eastern Daylight Time can cause a timewarp on a machine that is not configured correctly to handle the time zone adjustment.

For most AccuRev operations, a timewarp check is performed when the client contacts the server, or when a replica server forwards an AccuRev command to the master server.

AccuRev's Timewarp Detection Scheme

> **Note**: In previous releases, AccuRev sometimes performed small system-clock adjustments automatically. Now, AccuRev never adjusts a machine's system clock; it only reports discrepancies.

Each time a client program contacts the server program — or a replica AccuRev Server contacts the master AccuRev Server — AccuRev compares the system clocks on the two machines. If the discrepancy is less than or equal to 5 seconds, AccuRev proceeds to execute the user's command.

If there's a timewarp exceeding 5 seconds between a client machine and a server machine *and* the user's command specifies the ***–t*** option with a time specification (not a transaction number), AccuRev uses the (case-insensitive) value of variable AC_SYNC in the user's environment to determine how to proceed:

- If AC_SYNC is unset or has the value **ERROR**, an error occurs and a message like this appears:

```
client_time:    2008/07/14 10:54:54 Eastern Standard Time (1216047294)
server_time:    2008/07/14 10:49:56 EDT (1216046996)
timewarp:       298 seconds
The time on this machine is more than 5 seconds different than the
time on the server. Please fix this and try again.
You may have a problem with your system clock.
You can force the time on your system to match the server time
using the accurev synctime command.
```

- If AC_SYNC has the value **WARN**, no error occurs, but a warning that includes the timewarp details is displayed. Command execution proceeds.
- If AC_SYNC has the value **IGNORE**, no error occurs and no warning is displayed. Command execution proceeds.

Notes on timewarps when a client interacting with a replica server:

- A client command that "reads" information from the repository is handled by the replica AccuRev Server; only a timewarp between the client and replica server machines is relevant.
- A client command that "writes" information to the repository is handled by the master AccuRev Server; only a timewarp between the client and master server machines is relevant.
- If there's a timewarp exceeding 5 seconds between a replica server machine and the master server machine *and* a client of the replica server invokes the command ***accurev replica sync***, a warning message is recorded in the AccuRev Server log on the replica server machine.

Fixing System Clock Discrepancies

The sections below describe schemes for dealing with discrepancies between system clocks in the AccuRev client-server environment. We begin with the more desirable scheme: automatic, smooth clock adjustment. Then we describe the less desirable scheme: manual, sudden clock adjustment.

Automatic, Gradual Convergence of System Clocks

An optimal scheme for synchronizing machines' system clocks has these attributes:

- All machines in the network participate in the scheme, so the entire network is kept synchronized.
- Each machine's system clock is adjusted automatically (perhaps requiring some initial installation or configuration task).
- System clock adjustments can be made smoothly: for example, a discrepancy of 10 seconds can be gradually eliminated over the span of a few minutes by a minor speed-up or slow-down of a machine's clock. Presumably, such adjustments are imperceptible to human users and won't cause any "surprises" in time-sensitive applications.

Synchronization systems fitting this description are typically based on the standard Network Time Protocol (NTP) or its variant, the Simple Network Time Protocol (SNTP). For example, the Windows Time Service is available on recent versions of Windows. It provides a complete solution if all machines in your network are running Windows. For a more general, multi-platform solution, see the Network Time Protocol (http://www.ntp.org).

AccuRev-Related Guidelines

Here are guidelines for using a gradual-convergence system in an AccuRev network:

- Configure the system so that a single machine in the network acts as the time source that other machines synchronize with.
- Ideally, have all AccuRev machines participate in the synchronization system.
- If this isn't possible, make sure that the AccuRev server machine participates in the synchronization system. (AccuRev itself will take care of synchronizing its client machines to the server machine; see the next section.)

The purpose of these guidelines is to ensure that no AccuRev client machine gets into a situation of synchronizing itself with two different, and possibly conflicting, machines: the AccuRev server machine and a separate time-source machine.

Manual Synchronization Tools

A less desirable scheme for keeping system clocks synchronized is to occasionally type clock-adjustment commands manually on one or more of the machines. This method can be improved a bit by using scripts and scheduling tools such as ***cron*** (UNIX/Linux) and ***at*** (Windows). Only the **root** user (UNIX/Linux) or a user with administrator privileges (Windows) can set the system clock manually.

Setting the System Clock on the AccuRev Server Machine

On a UNIX/Linux machine, the ***date*** command changes the system clock. What time should you set the clock to? In many cases, you can use ***rsh*** or ***telnet*** to determine the time on another "time source" machine.

On a Windows machine, use the ***net time*** command to synchronize with a specified "time source" machine, or with the domain controller machine. To set the clock to a particular time, use the ***date*** command in a Command Prompt window, or double-click the digital clock in the Windows taskbar (lower-right corner of the screen).

Setting the System Clock on AccuRev Client Machines

The ***accurev synctime*** command changes a client machine's system clock to match the clock on the AccuRev server machine. The GUI command is ***Tools > Synchronize Time***.

5. Timestamp Optimization (TSO)

In various situations, AccuRev searches some or all of your workspace tree to determine which files should have a status of **(modified)** and to identify external files. This chapter describes the *Timestamp Optimization* (TSO) algorithm, how AccuRev uses it during workspace searches, and how to disable or override it.

Local, Client-Based Timestamp Optimization

Local, client-based TSO uses cached server information that is stored on the client. This approach to timestamp optimization ensures that all modified files in the workspace are correctly identified (even if they have an older timestamp and have been copied into the workspace from an external directory), while ensuring good performance.

> **Note:** Local, client-based TSO requires a one-time, full scan of the workspace. Because of this, the first **accurev stat** or **accurev update** command, whether executed from the AccuRev CLI or GUI, that your users perform on any *existing* workspace may take up to several minutes to complete. After this one-time scan has been completed on the workspace, these commands will behave with performance that is equivalent to or better than prior releases. Consider making your users aware of the initial one-time scan requirement for existing workspaces. (This does not apply to newly created workspaces.)

Local cache of element data

With local, client-based TSO, AccuRev maintains a cache database for each workspace that can be used to detect modified and external files. The information about each file in the local cache allows AccuRev to quickly determine an element's modified or external state based on file sizes and timestamps without needing to compute the hashcode or going to the server.

The advantage of the local, client-based TSO approach is that timestamps for each individual file are compared to the cache, rather than comparing them to a single workspace scan threshold, which is set to the timestamp of the oldest modified file. This not only provides more valid results (by catching all modified files, even those with older time stamps that have been copied into the workspace from an external directory), it can be more efficient in situations such as a workspace that has a large number of **(backed)** files that have later time stamps than a single modified file.

The cache is updated by AccuRev commands that affect the file status or content on disk such as **update**, **keep**, **add**, and **pop**.

Managing TSO Behavior

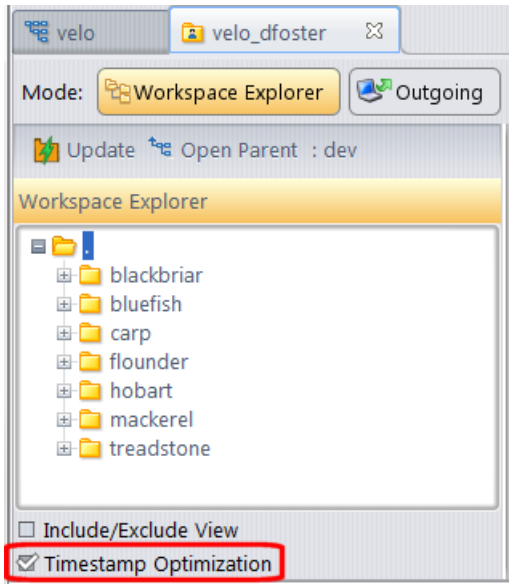
This section describes how to:

- Disable TSO for a workspace
- Override default TSO behavior

Disabling TSO for a Workspace

If you want, you can disable TSO for an individual workspace. Note that this can have an adverse effect on performance when running commands that scan the entire workspace, like **stat** and **update**.

1. In the AccuRev GUI, display the File Browser for the workspace for which you wish to disable TSO.
2. In either the Explorer mode or Outgoing Changes mode, clear the **Timestamp Optimization** checkbox.



The next time you perform a search on this workspace, AccuRev will not use any TSO algorithm.

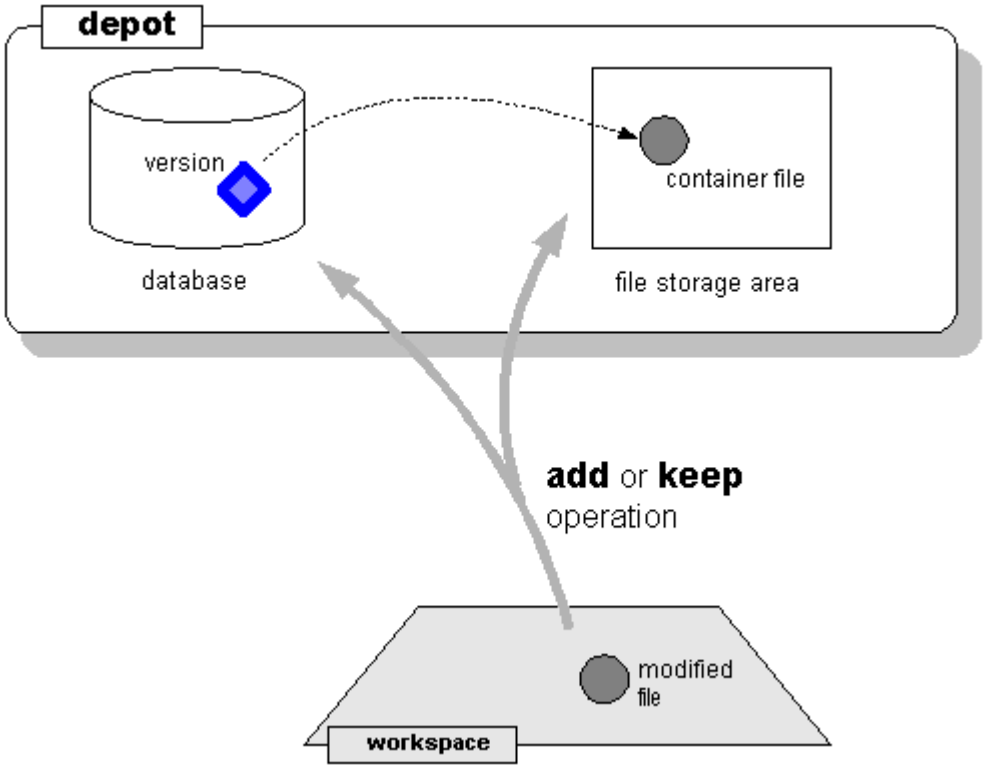
Overriding Default TSO Behavior

AccuRev CLI users can override the default TSO behavior by using the **-O** option with the **stat** command (or commands which call the **stat** command -- **anchor -n, co -n, files,** and **update**).

6. 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 database.



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.

The AccuRev CLI commands ***archive*** and ***unarchive*** shift container files back and forth between the normal and archived states. Before using ***unarchive***, the administrator must 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 [ -E <element-type(s)> ] [ -i ] [ -p <depot> ]
   [ -a | -I <stream-category(s)> ] [ -s <stream> ]
   [ -t <transaction-range> ] [ -c <comment> ] [ -R ]
   [ -Fx ] { -l <list-file> | <element-list> }
```

Determining Which Versions to Archive

archive determines the set of versions to archive as follows:

- Start with a particular set of file elements, which you specify as command-line arguments in the ***<element-list>***, or in a list-file (plain-text or XML format). You can include directories in this list; in this case, use the ***–R*** option to include the recursive contents of those directories.
- Optionally, take the subset of versions whose element type matches the specification made with ***–E***. (Note that different versions of an element can have different element types.)
- Optionally, take the subset of versions that were created in a particular streams (***–s***, for example, your current workspace stream). You can also archive versions from all streams in the depot (***–a***).
- Optionally, take the subset of versions created in a specific transaction, or range of transactions:

  –t <number>*single transaction*
  –t <number>–<number>*range of transactions*
- In addition to the multiple subsettings of versions described above, you can use the ***–I*** option to *include* versions in the set, based on where in the stream hierarchy they are referenced. For example, you can include versions that were not originally included in the set because they are referenced by one or more snapshots.

If you don't use the ***–a*** or ***–I*** option, ***archive*** refuses to archive any version that is currently visible in any stream or snapshot. Specified versions that are already archived are silently ignored.

Dry Run Capability

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 "dry run" before actually archiving any versions, to avoid any surprises.

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:

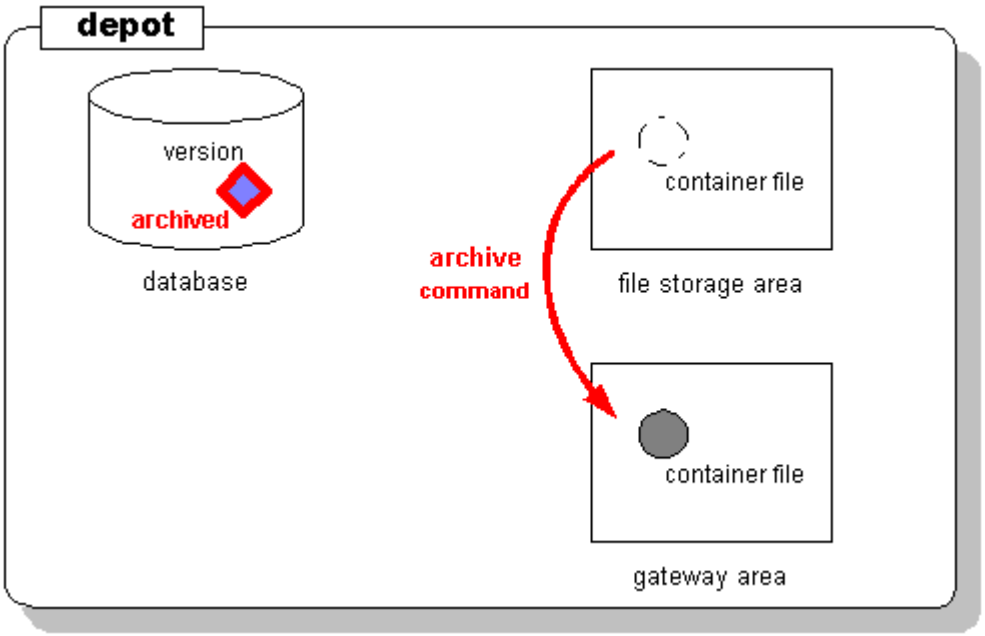
```
.../storage/depots/gizmo/data/25/07.sto
```

... 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 database.

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



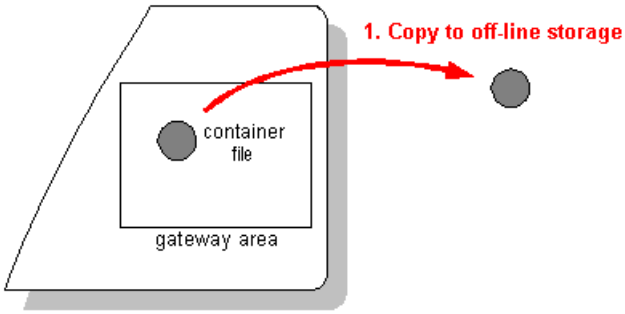
The changes made by this command are recorded in the database as a transaction of type **archive**. You can use the ***–c*** option to specify a comment string to be stored in this transaction. You can search for particular comment strings when using the ***hist*** command to locate previous **archive** transactions. See *Using ‘hist’ to Research Previous ‘archive’ Commands* on page 40.

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.

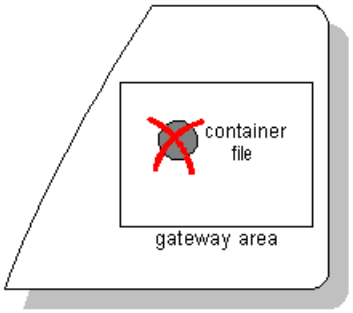
   

   WARNING! AccuRev has no way of tracking which tool you use for this purpose, or what off-line storage medium you copy the files to. It’s up to you to maintain good records of these activities!

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

Within the database, there is a complete record of all version-archiving activity for each 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 the same format as ***archive***. That is, you specify the versions to be restored in exactly the same way as you originally archived them (with one exception — see below).

For example, to archive all non-active versions of GIF image files in stream **gizmo_maint_4.3**:

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

Later, you can restore all those versions:

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

Exception: you can use the ***–a*** option to unarchive *all* the versions currently in the **archive_gateway/in** area. In this case, the ***unarchive*** command syntax doesn't mimic the ***archive*** command syntax exactly:

```
accurev unarchive -s gizmo_maint_4.3 -a
```

This set of restored versions might have been archived in a single step or in multiple steps.

7. Replication of the AccuRev Repository

This chapter describes how to set up and use AccuRev's repository replication feature. One server machine stores the "master" copy of the AccuRev data repository; any number of additional server machines can store "replicas" of the repository. Each replica contains some or all of the repository's depots. Users can send commands to the AccuRev Server software running on any of these machines.

Master and Replica

One host in the network acts as the AccuRev server machine: it runs the AccuRev Server process and contains the AccuRev repository. In a replication scenario, this original host (or more precisely, this instance of the AccuRev Server process) 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.

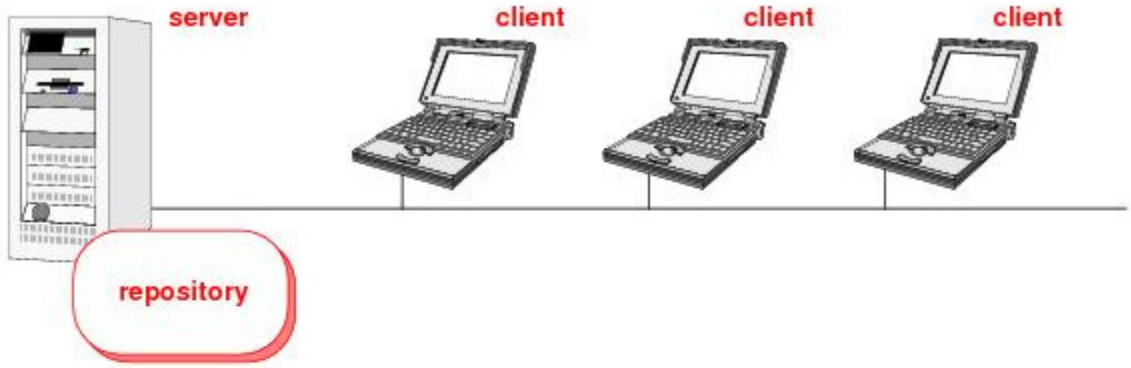
We use the terms master repository and replica repository to distinguish the multiple repositories in a replication scenario. The master repository is always complete and up-to-date; all transactions (operations that change the repository) are handled by the master server and are logged in the master 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 database into an exact copy (temporarily, at least) of the master repository database. Synchronization also occurs automatically whenever a transaction is initiated by a client using that replica server.

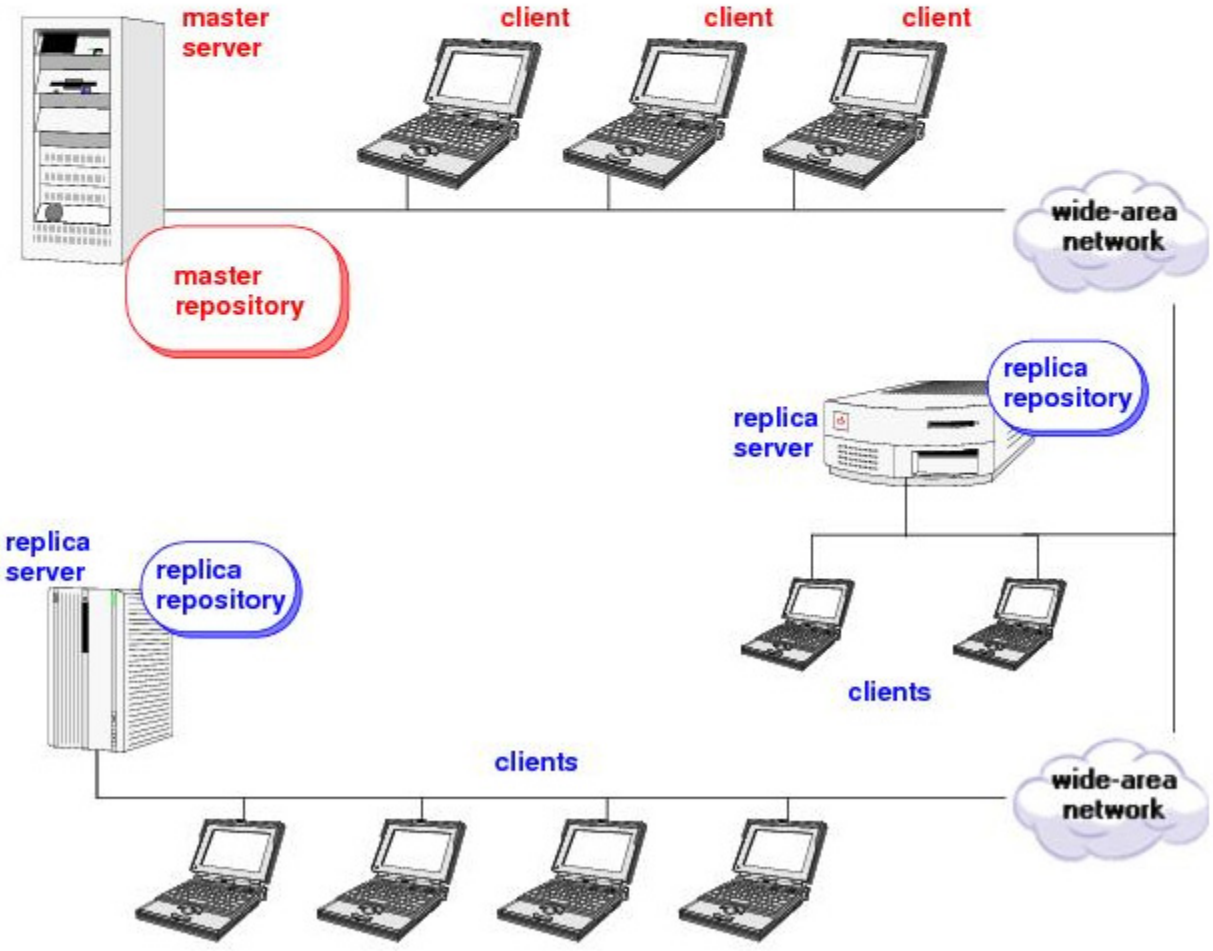
A replica repository can contain a selected subset of the depots in the master repository. If the master repository contains 10 depots, one replica repository might be configured to contain 4 of the depots, another replica repository might be configured to contain 7 of them, and a third replica repository might be configured to contain all 10 depots.

For more details on day-to-day operations involving master and replica repositories, see the sections starting with *Using a Replica Server* on page 48. First, we address licensing issues and describe the replica setup process.

Before Replication



After Replication



AccuRev Licensing in a Replication Environment

With standard/flexible licensing, AccuRev replica servers require no special license administration: license checking for the replica servers takes place on the master server automatically.

See *Replication Server Licenses* on page 127 of the *License Management* chapter for information about calculating numbers of license required by replica servers.

Installing and Configuring AccuRev in a Replication Environment

To install AccuRev software for replication, see the *AccuRev Installation and Release Notes*.

Once AccuRev replication software has been installed, use the procedures below to correctly configure your replica and master server(s).

Caution: Enabling replication poses a potential security risk. Before proceeding, be sure to read *Synchronization Security* on page 53.

Configure an AccuRev Server as a master server

1. Log in to the master and stop the AccuRev Server as described in *Starting and stopping AccuRev* in the *AccuRev Installation and Release Notes*.
2. Edit the **acserver.cnf** file, located in the AccuRev **bin** directory. Add the following line:

   ```
   REPLICATION_ENABLED = true
   ```
3. If you would like the master server to authenticate logins on the replica, ensure that the authentication methods on the master server and the replica server are the same, then add the following line to the **acserver.cnf** file:

   ```
   MASTER_AUTHENTICATES_LOGIN = true
   ```
4. Note the MASTER_SERVER and PORT settings in the **acserver.cnf** file. You'll need these settings when configuring the replica server below.
5. Restart the AccuRev Server process.
6. Create a new AccuRev username (***replica-user***) that will be used as the user identity for requests from a replica server.

Configure a Replica server

1. On the replica server, stop the AccuRev Server process.
2. Edit the **acserver.cnf** file, which is located in the AccuRev **bin** directory.
   - Change the keyword MASTER_SERVER to LOCAL_SERVER, and change the keyword PORT to LOCAL_PORT. But don't change the value of either setting.
   - Add new MASTER_SERVER and PORT settings, using the values of these settings on the master server. (These are the settings you noted in Step 4 above..)

After these edits, the four lines might look like this:

```
MASTER_SERVER = masthost
PORT = 5050
 ...
LOCAL_SERVER = replhost
LOCAL_PORT = 5050
```

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

Important! After editing acserver.cnf, you must restart the AccuRev server. If you have edited the values for SITE_SLICE_LOC, MASTER_SERVER, or PORT, you ***must*** *also run the* ***maintain server_properties update*** *command before the new values will take effect. See The 'maintain' Utility on page 115 for more information.*

Establish an AccuRev User Identity for the AccuRev Server Process

These steps ensure that the replica server process has an AccuRev username (***replica-user***) with enough rights to access all the files in the master repository. AccuRev ACL permissions control access to depots and streams for specified AccuRev users and groups.

Caution: As of AccuRev 5.2, your site may have also implemented the optional element-level security (EACLs) feature. If you are using element security (EACLs) to control access to your data, you *must* ensure that permissions are correctly set for ***replica-user***. If they are not correctly set, the replica may not be able to fetch any data from the master server, or it may be possible for a user to log into the replica and access elements that the replica is not supposed to have access to. See *Setting Permissions for a Replica Server* on page 64 for more information.

Be sure to use the same ***replica-user*** username on both the replica and the master.

1. Create an operating-system user ***replica-user***, and then log in as that user.
2. Configure the AccuRev server:
   - **UNIX/Linux:** Edit the server configuration file so that ***replica-user*** is the identity of the replica AccuRev Server. See *Operating-System User Identity of the Server Processes* on page 13.
   - **Windows:** Reconfigure the AccuRev Server service to run as ***replica-user***, instead of as **LocalSystem**. In the Control Panel's Services program: open the Properties window for the AccuRev service, go to the Log On tab, select "This account", and enter the actual value of ***replica-user*** and its Windows password.
3. Next, establish ***replica-user***'s AccuRev-level credentials:
   - Open a command-shell or C-prompt window.
   - Set environment variable ACCUREV_HOME to ***replica-user***'s home directory. (**Note:** Make sure that the path to the home directory does not include any spaces.)
   - Create a "permanent" session file for user ***replica-user***, for accessing the AccuRev Server on the master server from the client machine replica:

     ```
     accurev login -n -H <mast-server-host>:<mast-server-port> replica-user
     Password: ********
     ```

     The **–n** option makes this session file valid indefinitely.

4. Continue to set up replication as described in the following sections.

Synchronize the Site Slice

Perform these steps on the replica server:

1. Start the AccuRev Server and AccuRev Database Server.
2. Run the ***accurev synctime*** command to ensure that the replica server time is synchronized with the master server time.
3. Run the ***accurev replica sync*** command to copy site-specific data from the master server to the replica server. In particular, this command makes the AccuRev Server on the replica aware of all the depots on the master server.

Indicate the Depots to be Replicated

The AccuRev repository on the replica server now has an up-to-date site slice, but the repository doesn't yet contain detailed data on any depots.

1. Log in to the replica server. See *Authenticating a Replica User on the Master* on page 58 if you would like to authenticate all replica server logins on the master server.

   ```
   accurev login repl_user
   Password: ********
   ```

   Note: If ***repl_user*** on the replica server and the replica user defined on the master server bear the same name, there are now two session files for **repl_user**, one for accessing the master server and the other for accessing the replica server.

2. List all the depots in the master repository, by executing this command on the replica server:

   ```
   accurev show -fix depots
   ```

   In the XML-format output, the depots that exist in the master repository, but are not replicated on the replica server, are listed with this attribute:

   ```
   ReplStatus = "missing"
   ```

3. For each depot that is to be replicated on the replica server, execute a ***mkreplica*** command. For example, if depots named **widget**, **gadget**, and **cust_support** are to be replicated:

   ```
   > accurev mkreplica -p widget
   Created replica of depot 'widget'.
   Synchronizing ...
   Done.
   > accurev mkreplica -p gadget
       ...
   > accurev mkreplica -p cust_support
       ...
   ```

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, the 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 and then running ***accurev synctime*** to synchronize the server times.

Tip: The ACCUREV_SERVER environment variable influences which AccuRev Server the AccuRev GUI interacts with at startup time. See *ENV_VARS* on page 264 of the *AccuRev CLI User's Guide* for more information.

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:

- Configuration management commands that *read* data from the repository — such as ***files***, ***diff***, and ***cat*** — use the replica repository.
- Configuration management commands that *write* data to the repository — such as ***keep***, ***promote***, and ***merge*** — use the master repository. After the master repository has been modified, the local replica repository is automatically brought up to date. For details, see *Synchronizing a Replica Manually* on page 51, which describes how you can bring the local replica repository up to date when you are *not* writing data to the repository.
- All AccuWork issue management operations are handled by the master server. Thus, replication does not improve AccuWork performance.

The Update Command

The ***update*** operation works as follows when you execute it on a client that uses a replica server:

1. An implicit ***replica sync*** command is performed, copying database transactions from the master repository to the replica repository. This brings the replica database completely up to date.
2. A ***stat*** operation is performed on the replica server, to determine the state of the workspace stream and its backing stream.
3. Data files representing new versions of elements are copied from the file storage area in the master repository to the file storage area in the replica repository.
4. Data files are copied from the replica repository to your workspace tree. In addition to the files retrieved from the master repository in the preceding step, this can include files that have already been "downloaded" to the replica repository through other users' commands.
5. On both the replica server and the master server, the transaction level of the workspace is set to the most recent transaction (or to the transaction specified with ***update –t***).

Command Interaction in a Replicated Environment

If a particular replica server is executing a ***mkreplica*** command, a request to perform a ***replica sync*** on that server might fail. Be careful to run ***mkreplica*** at a time when it is unlikely that any user or script is invoking ***replica sync***.

If a ***replica sync*** command does fail for this reason, first make sure that no ***mkreplica*** command is executing, then invoke ***replica sync*** again.

Similarly, if a ***replica sync*** command is already in progress, in rare situations a client command sent to the same replica server might not complete correctly, causing an error message to be displayed. The change is made correctly on the master server in such situations. The user who issued the command can perform an ***update*** command to have the change reflected in the workspace.

Removing Storage Containers on a Replica Server

Use the ***replica archive*** command to remove unused container files from the depot's **storage** directory on a replica server.

The ***replica archive*** command activates the **server_preop_trig** trigger with a **replica_archive** command. The command lists each element for which the storage container was removed. For example:

```
Removing storage for element /file1.txt (2/5)
Removing storage for element /file1.txt (2/6)
```

For more information on this command, see the *replica* command in the *AccuRev CLI User's Guide*, or enter ***accurev help replica*** in the CLI.

Removing a Replica Server

Since the master repository is always complete and up-to-date, removing a replica server is very simple:

1. Inform any users on the replica server of the change. Every user must log out and edit the **acclient.cnf** file (located in the AccuRev **bin** directory) to point to another server.
2. On the replica server machine:
   - Stop the AccuRev Server and AccuRev Database Server.
   - Uninstall AccuRev by running the uninstall program.
   - Access the AccuRev installation directory and remove any remaining files.

Improving Replica Performance

The techniques described in this section can greatly improve the user experience with AccuRev replicas.

Performance of a Newly Created Replica

Use the following procedure to download version files to a replica when it is first created, instead of when the replica's first user performs an ***Update***.

Perform these steps on the master server:

1. Stop the AccuRev Server and AccuRev Database Server.
2. For each depot to be replicated, use a ZIP utility (or ***tar***, or some other tool) to copy the **data** subdirectory of the depot's slice. Example:

```
cd /opt/accurev/storage/depots
zip -r /tmp/oscar_data.zip oscar/data
zip -r /tmp/felix_data.zip felix/data
```

3. Restart the AccuRev Server.

Perform these steps on the replica server:

4. Replicate the depots with the command ***accurev mkreplica***. Example:

```
accurev mkreplica -p oscar
accurev mkreplica -p felix
```

5. Stop the AccuRev Server and AccuRev Database Server.
6. For each replicated depot, move the slice's **data** subdirectory aside. Example:

```
cd /var/applications/accurev/storage/depots
mv oscar/data  oscar/data.ORIG
mv felix/data  felix/data.ORIG
```

7. For each replicated depot, recreate the slice's **data** subdirectory, using the copy created in Step 2. Example:

```
unzip oscar_data.zip
unzip felix_data.zip
```

8. Restart the AccuRev Server.

Performance of Heavily Used Replicas

In a distributed computing environment, many new versions might be created in a replica at Site A, while no one is working at Site B. The first person to update his workspace at Site B pays the penalty of waiting for all those new versions to be transferred from the master server to Site B.

To avoid this situation, create a "dummy" workspace at Site B for each heavily used stream, and create a script that updates all these workspaces. Then, set up a ***cron*** job (UNIX/Linux) or a Scheduled Task (Windows) to run the script on a regular basis. Ideally, the script would run at least once a day — before the start of Site B's work day, but after the end of the work day of Site A (and any other sites). For example, this ***crontab*** line runs script **update_replica_workspaces** at 4:30AM every day:

```
30 4 * * * /usr/local/bin/update_replica_workspaces
```

Running the script at additional times during the work day can further improve replication performance from the users' viewpoint — if there is sufficient file-transfer bandwidth.

Enabling Data Compression

AccuRev 5.5 supports data compression on connections between master and replica servers. Enabling data compression can improve performance on low-bandwidth networks, especially networks whose bandwidth is slower than 10Mbps.

To enable data compression, add the following line to the **acserver.cnf** file on each replica server:

```
COMPRESSION_ENABLED=TRUE
```

Note: If your installation makes use of hardware compression, enabling data compression on the AccuRev replica server might decrease performance.

Backing Up and Restoring Replica Servers

AccuRev recommends that you back up the replica servers in your environment on the same schedule you use for backing up your master. See *Backing Up the Repository* on page 6 for more information.

Triggers and Replication

A user's invocation of an AccuRev command on a client machine may cause a client-side trigger and/or a server-side trigger to fire. A client-side trigger fires on the user's client machine — no ambiguity there. But in a replication environment, there are two AccuRev Servers to consider: the one on the replica server machine and the one on the master server machine. Where does a server-side trigger fire?

- A ***rmreplica*** command fires a ***server_admin_trig*** trigger on the replica server. The AccuRev user identity ("principal-name") of the user who invoked the command is passed to the trigger script.
- A ***replica sync*** command fires a ***server_admin_trig*** trigger on the master server. The AccuRev user identity of the AccuRev Server process on the replica server is passed to the trigger script — not the identity of the user who invoked the command. The command-name is passed to the trigger script as **replica_sync**.
- For all other commands that write to the repository, a server-side trigger fires on the master server. The AccuRev user identity ("principal-name") of the user who invoked the command is passed to the trigger script.
- For commands that do not write to the repository, a server-side trigger fires on the replica server.

Creating New Depots

New depots can be created only in the master repository, not in a replica repository. If a client using a replica repository issues a ***mkdepot*** command, an error occurs:

```
Cannot create a new depot on the replica server
```

After creating a new depot in the master repository, you can include it in a replica repository with this sequence of commands, issued on a client that uses the replica server:

```
accurev replica sync
accurev mkreplica -p <depot-name>
```

Adding and Removing Depots from a Replica Repository

After you have set up a replica repository, you can use the commands ***mkreplica*** and ***rmreplica*** to change which depots are included in the replica repository. These commands are described in the *AccuRev CLI User's Guide*, at *mkreplica* on page 168, and *rmreplica* on page 220.

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 transfers data from the master repository site slice to the replica repository site slice. It also transfers database transactions from the master repository to the replica repository — but only for the depots that are included in the local replica. It does not transfer the corresponding storage files for ***keep*** transactions. See *On-Demand Downloading of a Version's Storage File* on page 52.

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 48.

> 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 synchronizes the master repository with a replica repository during:

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

Storage files for versions are downloaded from the master repository to the local replica repository during an ***update***. (See *The Update Command* on page 48.) The storage file for an individual version is downloaded when a client using a replica server explicitly references that version. Examples:

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

Both these commands cause the storage file for version **talon_dvt/12** of file **mywork.c** to be downloaded to the local replica repository 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 program 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 identities.

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

The replica_site.xml File

Each replica repository's site slice directory contains an XML-format file, **replica_site.xml**. This file contains information about the depots that are replicated in that repository. The ***mkreplica*** and ***rmreplica*** commands maintain the contents of this file.

8. 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 not an absolute restriction — see *READ ME NOW: Assuring the Integrity of the AccuRev Repository* on page 5.) The repository consists of multiple slices: the site slice contains information that pertains to the entire repository, and each depot has its own slice.

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:

- Install AccuRev on the new machine.
- Move your data repository to the new machine.

The procedure described below is safe to use even if the new machine has a different byte order than the old machine.

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/Linux 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 21.

On the Source Machine

1. Execute the ***accurev show slices*** and ***accurev show depots*** commands, and save the output for reference in the following steps.
2. Perform a backup of the AccuRev repository, as described in *Backing Up the Repository* on page 6.
3. Make the backed-up files accessible to the destination machine.

On the Destination Machine

1. Request a new license key for the destination machine by filling out the form at *http://www.accurev.com/license-request.shtml*. Save this key file to a location on the destination machine.

2. Perform a full AccuRev Server installation on this machine as described in the *AccuRev Installation and Release Notes*. The installation directory pathname need not be the same as on the source machine.
3. Stop the AccuRev Server, as described in *Controlling Server Operation* on page 21.
4. Perform a restore of the AccuRev repository you backed up on the source machine, as described in *Restoring the Repository* on page 8.

   Note: Do *not* copy over the **/storage/db** directory on the destination machine. This directory should not have been included in the backup of the source machine repository, and you do not want to overwrite the **/storage/db** directory on the destination machine.

   Check that the file ownership and permissions are correct; they may need to be altered if the username used to run the AccuRev server is different on the source and destination machines, or if your backup/restore tool does not correctly preserve these settings.
5. Update the server properties:

   ```
   maintain server_properties update
   ```

   See *The 'maintain' Utility* on page 115 for more information.
6. Change the locations of the slices to their locations on this machine using the output from the ***accurev show slices*** and ***accurev show depots*** commands on the source machine. To do so, run the following command for each slice:

   ```
   maintain chslice <slice-number> <new-location>
   ```
7. Start the AccuRev Server.

9. AccuRev Security Overview

This chapter presents an overview of AccuRev's security-related features. We discuss and compare the following topics, and provide pointers to more detailed documentation.

- *Users and Groups*
- *User Authentication*
- *Locks on Streams*
- *Access Control List (ACL) Permissions*
- *Element-Level Security (EACLs)*
- *Restricting Access to Commands using Triggers*
- *Implementing Secure Sockets Layer (SSL) Encryption*

Users and Groups

Each AccuRev user must have an AccuRev username (also called a "principal-name"). AccuRev maintains its own user registry, which is separate from the registry maintained by your machine's operating system (or the network).

Optionally, you can create AccuRev user groups, and add users to the groups as "members". Starting in Version 4.5, groups can be nested within one another; that is, a group can be a member of another group. AccuRev groups are independent of operating system groups.

Groups are used by the ACL facility to grant or deny access to a particular resource for an entire set of users. They are used by the Locks facility to specify a set of users to which a stream lock does or does not apply. (See *Locks on Streams* on page 60 and *Access Control List (ACL) Permissions* on page 60.)

Usernames and Groupnames

Each AccuRev group has a user-defined name. Usernames and group names share a "namespace" in AccuRev. This means that a user and a group cannot have the same name.

User Authentication

With a few exceptions, a user cannot execute AccuRev commands unless he is authenticated as a registered AccuRev user. For authentication purposes, each registered AccuRev user has a username/password pair recorded in the database. A user's password can be empty.

AccuRev automatically distinguishes two categories of users — those who have empty passwords, and those who have non-empty passwords. The keywords **authuser** and **anyuser**, respectively, are used by the ACL facility to designate these categories. See *secinfo* on page 223 of the *AccuRev CLI User's Guide* for more information.

The “accurev_login” User-Authentication Method

AccuRev supports authenticating a user through an explicit login to the AccuRev Server. Using the GUI or the CLI, the user invokes the ***Login*** command and enters his password (possibly empty). This launches a user session, which is typically of limited duration (a few hours). When the session expires, the user must login again to continue using AccuRev.

The “custom” User-Authentication Method

AccuRev also supports script-based authentication of AccuRev users. This **custom** method works as follows:

1. The user invokes the ***login*** command through the AccuRev GUI or CLI client, and types a password.
2. The client encrypts the password and transmits the username/password combination to the AccuRev Server.
3. The AccuRev Server verifies that the username exists in AccuRev’s user registry. It does not check the password, however.
4. The AccuRev Server invokes a script named ***server_auth_trig***, passing it the username and password combination. The exit status of this script determines the success/failure of the ***Login*** command. See *The ‘server_auth_trig’ Script* on page 58 for details.

This feature provides tremendous flexibility. For example, an authentication script could use an external user database, accessed through an LDAP interface, to perform AccuRev user authentication.

Authenticating a Replica User on the Master

Prior to V4.7.3, any authentication scheme on the master was only as trustworthy as the security of all the replica servers. The changes to replica authentication in AccuRev 4.7.3 allow sites using replication to deploy replicas to sites they may not trust to do user authentication, by allowing user authentication to be performed on the master server only.

Use the MASTER_AUTHENTICATES_LOGIN parameter in the **acserver.cnf** file on the master server:

- If this parameter is set to **true**, user authentication is performed on the master server only. The master and replica servers must have the same authentication method. Only the *accurev_login* or *custom* authentication methods are allowed. The replica server must be able to contact the master server to log in; if the connection fails, the login fails.

  To change the authentication method, change and restart the master server first, then change and restart all replicas.
- If this parameter is set to **false**, user authentication can be performed on the replica, as before.

After changing the value of the parameter, restart the master server.

As a result of these changes, the XML input to the **server_auth_trig** script has changed to add **server** and **port** elements, which should contain the hostname and port of the master server.

The ‘server_auth_trig’ Script

The ***server_auth_trig*** user-authentication script, used by the **custom** user-authentication method, is similar to the ***server_admin_trig*** administrative trigger script. A sample script, implemented in Perl, is included in the AccuRev software distribution, in the **examples** subdirectory.

To deploy a user-authentication script, place an executable file in subdirectory **triggers** of the **site_slice** directory:

- UNIX/Linux: the file must be named **server_auth_trig** or **server_auth_trig.pl**
- Windows: the file must be named **server_auth_trig.bat** or **server_auth_trig.exe**

The script is called when a user invokes the AccuRev ***Login*** command. It executes in a subprocess of the AccuRev Server. The script's standard input is a simple XML document, with the structure shown in this example:

```
<triggerInput>
   <hook>server_auth_trig</hook>
   <command>login</command>
   <ip>192.168.6.186</ip>
   <username>derek</username>
   <password>MyP@ssw0rd</password>
   <server>myserver</server>
   <port>5050</port>
</triggerInput>
```

(No password encryption is necessary, because this XML document is passed to the subprocess through an operating system pipe, not through a file.) The script's standard output is appended to file **triggers.log**, located in the **logs** subdirectory of the **site_slice** directory. No output is required, however; only the script's exit status is significant:

- If exit status is zero, the user's Login command succeeds.
- If exit status is nonzero, the user's Login command fails.

The user's ***Login*** command also fails if the script does not execute properly — for example, if a runtime error occurs or the script file is not executable.

Selecting the User-Authentication Method

When you install AccuRev on the machine where the AccuRev Server process will run, the AccuRev Installation Wizard automatically sets the authentication method. Thereafter, you can switch methods with the ***authmethod*** command. Example:

```
accurev authmethod accurev_login
```

The switch takes place immediately. This might cause user confusion; it is not a command to run casually.

How AccuRev Records the User-Authentication Method

The current user-authentication method is stored in file **preferences.xml** in the **site_slice** directory. (Don't confuse this with individual users' **preferences.xml** files.) For example:

```
<preferences>
   <authmethod>accurev_login</authmethod>
</preferences>
```

As an alternative to the ***authmethod*** command, you can change the user-authentication method by editing this file, then stopping and restarting the AccuRev Server process.

Restriction on Access to the “Add User” Command

User authentication is performed by the add-new-user command (GUI: ***Add User***, CLI: ***mkuser***). To prevent a chicken-and-egg problem, user authentication is bypassed if the AccuRev user registry is currently empty. After the first AccuRev user has been created, only an authenticated user can add users to the AccuRev user registry.

Locks on Streams

Each stream can have one or more locks on it. A lock prevents a certain set of users from using the stream to create new versions of elements. That is, it prevents execution of the ***Promote*** command — either promoting from the designated stream, or promoting to the designated stream, or promoting in either direction.

A lock can be absolute, applying to all users. Alternatively, you can specify that a lock applies to a particular AccuRev user, or to a particular AccuRev group. Conversely, you can specify that a lock applies to everyone *except* a particular AccuRev user, or to everyone *except* a particular AccuRev group.

Locks can also prevent reconfiguration of the contents of a stream with the include/exclude facility.

For more on locks, see the ***lock*** reference page in the *AccuRev CLI User’s Guide*.

Access Control List (ACL) Permissions

In addition to (or instead of) locks, each stream can have one or more permissions on it. Whereas a lock controls the ability to create new versions (through the ***Promote*** command), a permission is more general: in addition to controlling ***Promote***, it controls the ability to read data from the stream, using such commands as ***Annotate***, ***Diff***, and ***Open***. A permission also controls workspace-specific commands, such as ***Update*** and ***Populate***.

Unlike locks, which always apply to individual streams, ACL permissions can be configured to apply to entire stream subhierarchies.

You can create an ACL permission that applies to an entire depot. This provides a way of controlling access to all of a depot’s file system data, in all streams. It also provides a way to control access to a depot’s AccuWork issues.

For more on ACL permissions, see the ***setacl*** reference page in the *AccuRev CLI User’s Guide*.

Element-Level Security (EACLs)

Video Tip	Click *this link* to view an introductory video on EACLs.

Historically, AccuRev has supported a security mechanism for depots and streams using Access Control Lists (ACLs). To address customer needs for more extensive security, AccuRev 5.2 introduced greatly enhanced ACL security control at the element level.

A depot ACL protects all the elements, streams, and issues in the depot. A user who has no access to the depot can still see the stream hierarchy, but none of the files in the streams. They can also not edit the

streams in anyway. A stream ACL prohibits editing streams and viewing content in a stream. It can also be inheritable down the hierarchy. The table below shows what stream, depot and element ACLs restrict:

	Viewing Content	Editing Stream	Viewing and Editing Issues
Stream ACL	Yes	Yes	No
Depot ACL	Yes	Yes	Yes
Element ACL	Yes	No	No

Features

There are three main features of the new element security mechanism:

- The ability to allow or deny access to all versions of an element, no matter what stream they are in, for a specified user or group. If a user is denied access they cannot see or view the element.
- The ability to specify who can modify the access to an element. This is a higher privilege than just being allowed access.
- The ability to view the history of access changes for auditing purposes. This means being able to trace who assigned access to an element, when they did it and what they did.

Basic Terms

To control access to elements (files and directories), AccuRev 5.x implements a new ***eacl*** command (see *eacl* on page 104 of the AccuRev *CLI User's Guide*) to set and modify *Access Control Lists (ACLs)* and *Access Control Entries (ACEs)*. An ACL is a list of security protections that applies to an element. An ACE is an entry in an ACL that defines a *principal* and a *privilege*.

A principal can be:

- a user
- a group
- *all*

A privilege can be:

- *Full* – the ability to see and view the element and to modify its ACL
- *Allow* – the ability to see and view the element, but not modify its ACL
- *Read only* - prevents the user from modifying the element or its ACL during ***add***, ***keep***, ***move***, ***defunct***, and ***revert*** commands.
- *Deny* – the inability to see and view the element or modify its ACL

An ACL contains zero or more ACEs. An element can have only one ACL assigned to it at any point in time. You specify whether to set, add, or remove an ACE (principal and privilege) to an element, and AccuRev takes care of the ACLs automatically. Note that ACLs cannot be created or modified without an element.

Important Concepts

Before jumping into implementation details, it is important to keep the following concepts in mind to avoid confusion and incorrect assumptions:

Element-based security

AccuRev security is based on AccuRev *elements* rather than pathnames, and this makes AccuRev security both more powerful and more complex than typical filesystem security. Don't forget: an AccuRev element can be either a file or a directory, and can also be an element link (elink) or symbolic link (slink).

An element is defined by its element id ("EID"), which never changes. It also has a pathname (defined by its "parent EID") and a name, both of which can change.

A symbolic link (slink) is an element whose contents is a pathname which can point to either AccuRev or non-AccuRev data.

An element link (elink) is an element whose contents is a pointer to another element, which must be in the same workspace. The target element can be a directory element, a file element, another element link, or a symbolic link.

A key concept to master is how ACLs apply to the element versus the element's contents. See *Inheritance* on page 63 to understand how access to *contents* and *namespace* differs,

Likewise, if you use element links (elinks), you must understand how AccuRev element security does and does not apply to the link and to the destination file, otherwise a file that you think is denied to a principal may still be accessible.

In AccuRev, the same element can have multiple pathnames, depending on what streams it is in at different times. While pathnames and filenames can change, the element id (EID) never does, and this is what AccuRev security is based upon. This allows you to secure the contents of an element no matter what its current pathname is.

Remember:

- A simple pathname may refer to more than one element (such as both ends of an elink), and likewise a single element may have more than one pathname (such as one pathname to the actual file, and a different pathname for that file as a target of an elink).
- An element may have different paths in different streams, and may have different paths at different points in time.

ACLs are not TimeSafe®

The third concept to keep in mind when using AccuRev element security is that ACLs are always current: they are NOT TimeSafe®, and they are not affected by what stream they happen to appear in. This means that you cannot recreate ACLs as they appeared at a certain point the same way as you can recreate file versions or stream configurations. (You can, however, view ACL changes through the ***hist*** command.) It also means that the appearance of a snapshot stream can change for a particular user or group, The snapshot doesn't change—that is TimeSafe. But a user's *access* to the contents of that snapshot can change. If your access changes, the next time you perform a ***pop*** command on your workspace, you may see files appear or disappear.

Privileges

The default behavior for an element whose ACL doesn't contain an ACE for a user is to deny access to that user. The *deny* privilege overrides *allow* and *full*. This makes it easy to give access to everyone (all), but then deny a few. If a user has both *allow* and *full,* then *full* privilege is granted.

The *full* privilege means the ability to change which ACL is associated with the element and to modify the ACL as well. A user only needs to have *allow* privilege to show ACLs on an element or to view the ACL. When setting or modifying an ACL for an element, it is the current ACL associated with the element that controls the ability to set or modify the ACL. The table below describes what a user can or cannot do based on privileges:

	View ACL	Modify ACL	View Elements	Modify Elements
Full	Yes	Yes	Yes	Yes
Allow	Yes	No	Yes	Yes
Deny	No	No	No	No
Read only	Yes	No	Yes	No

Inheritance

Element security inheritance differs for content and namespace.

Content is based on a *static inheritance* model. This means that an ACL is set for an element when assigned by the user or when it is created and inherits the ACL from its parent.

Namespace access is based on *dynamic inheritance.* This means that it is computed by the element's pathname when it is accessed.

This means that if you are denied access to a particular element, you cannot see the contents or even the name of the element. However, if you are denied access to directory, but not denied access to a file in that directory, you can still perform certain operations on that file by using its EID. For example, if you know its EID, you can ***cat*** the file. You will not be able to see its name, but you can see its contents.

The user also has the ability to set an ACL recursively down the directory tree, making it easy to assign an ACL to multiple elements. Multiple elements can be specified on the command line as well.

Renaming, moving, or defuncting an element does not change its ACL or access. Only setting or modifying an ACL on the element can change access to the element.

Access Denied

This section describes what it means when access is denied to the content or the name space of an element:

- A user is denied access to the content and namespace of an element if they are denied access to the element itself.
- A user is also denied access to the namespace of an element if they are denied access to any of the directories in the element path.
- If a user is denied access to the namespace, but not the element, the user will be denied access to the content if they try to specify the element name.

- If a user is denied access to the namespace, but not the element, the user will NOT be denied access to the content if they try to specify the element EID.

Create a superuser to Administer Element Security

When you install an AccuRev release that contains element security for the first time, all elements are assigned the `allow` privilege. This provides new and existing customers with the exact same behavior provided by previous versions of AccuRev, but it does not permit existing privileges to be modified: modifying an element privilege (an EACL) requires the `full` privilege.

Only AccuRev users identified as a *superuser* are granted the `full` privilege needed to modify element privileges. The superuser always has full access to all elements and is needed to begin the process of setting up element security, and to fix problems that users may cause, such as denying everyone access to an element.

The AccuRev administrator can set or unset a user as a superuser using the **maintain** program, as follows.

To set a user as a superuser, run this command:

```
maintain su -a <principal_name>
```

To remove superuser status from a user, run the same command with the -r option:

```
maintain su -r <principal_name>
```

Auditing

To provide auditing (or history) capability for element security, AccuRev keeps track of all ACL changes to elements. This doesn't mean that element ACLs are TimeSafe® (and it is important to realize that element ACLs are **not** TimeSafe). The most recent version of an ACL and the most recent ACL assigned to an element is always used for authorization of access. Historical information on changes to ACLs is only used for audit reporting. The ***hist*** command is used to display the history of ACL changes on an elements.

Setting Permissions for a Replica Server

If you are using a replica server in an EACL environment, it is critical that you set EACL permissions correctly not only for the users on the replica, but for the special replica user account (***`replica-user`***) that is used to communicate between the replica and the master server.

For example, assume that you want the users of a replica server to be able to access all of the files underneath folder `offshore_files`, but that you do not wish them to be able to see anything under `hq_files`. If you fail to assign **allow** permission to ***`replica-user`*** for `offshore_files`, the replica may not be able to fetch any files from the master.

More importantly, if you fail to assign **deny** permission to ***`replica-user`*** for `hq_files`, it would be possible for a privileged user to log into the replica and bring the denied files down to the replica. Continuing this example, say that a privileged user from corporate headquarters visits the offshore site where the replica is being used. If the ***`replica-user`*** is not correctly configured, the privileged user could log into the replica server and inadvertently bring down `hq-files` elements by doing an **update**. If the ***`replica-user`*** is configured with **deny** access to these files, the server will not send them to the replica.

EACL Usage Scenarios

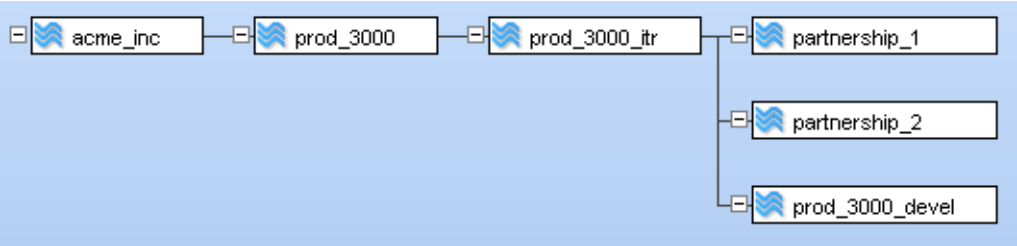
The following examples illustrate two basic approaches to setting up element ACLs:

- **less restrictive**—By default, all users have `allow` access to all elements, unless explicitly denied.
- **more restrictive**—By default, all users are denied access to all elements, unless explicitly allowed.

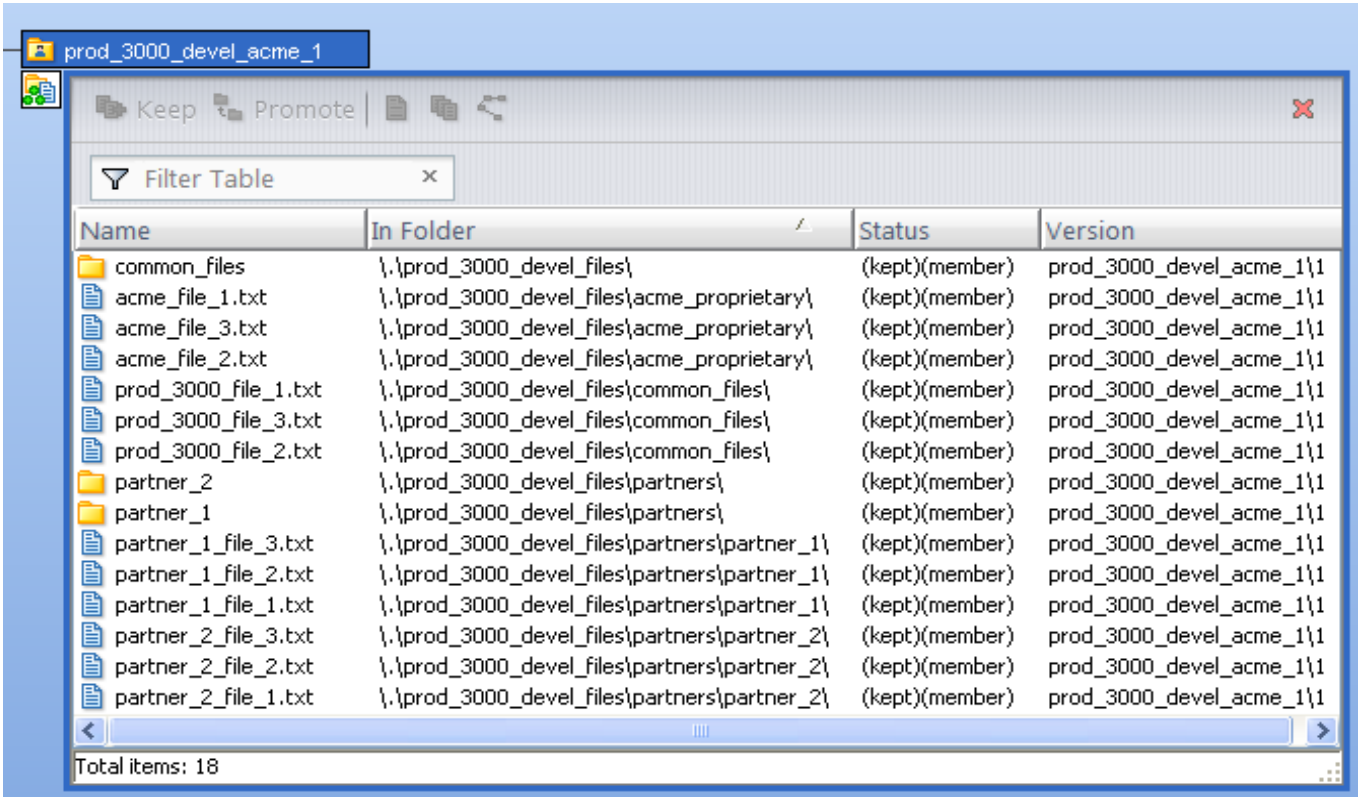
Both examples make use of the same environment: Assume that your company ("Acme Agile, Inc.") has a new product ("Product3000"), and you will be collaborating with two partners to develop the code ("Partnership_1" and "Partnership_2"). For simplicity, these examples illustrate settings ACLs for specific users. In a real-life scenario, you would typically assign many users to various groups, and then set the element ACLs by group.

Set Up

The AccuRev administrator initially set up a basic stream structure:



You, Acme employee user "acme_1", create a workspace off of stream `prod_3000_devel` and set up the initial files and folders for the project:



Since you are a trusted user, the AccuRev administrator grants you "superuser" status so that you will be able to assign element ACLs as needed.

To make you a superuser, the AccuRev administrator logs in to the AccuRev server machine and uses the AccuRev ***maintain*** command. **Note:** The administrator must first shut down the AccuRev server before using ***maintain***.

```
> maintain su -a acme_1

AccuRev 6.0 (2013/05/20)
Copyright (c) 1995-2013 AccuRev Inc. All rights reserved
Changed user 'acme_1' to a super user

>
```

The administrator restarts the AccuRev server, and you log in to the AccuRev CLI as `acme_1` and start assigning EACLS as shown below.

Less Restrictive Scenario

Your starting point is the system default for first time AccuRev installations: everybody has ***allow*** access to all files and namespaces, and you will explicitly change privileges to certain elements for certain users to either **full** or **deny**.

Remember:

- ***deny*** means that the user cannot even see the element.
- ***allow*** means that the user can see and work with an element, but not change its ACL
- ***full*** means that the user not only can see and work with an element, but can also set the ACL on that element.
- ***deny*** overrides ***allow*** and ***full,*** and if there is no explicit ***allow,*** a user is denied.

In this case, you want to grant the user from Partnership_1 ("`part_1`") ***allow*** privileges on the `partner_1` files, but ***deny*** that user any privileges on the `partner_2` files or the `acme_proprietary` files. Likewise, the user from Partnership_2 ("`part_2`") should have ***allow*** privileges to the `partner_2` files, but ***deny*** privilege to the `partner_1` and `acme_proprietary` files.

Note: It is conceivable that partners might want ***full*** access to their own files so that they could set specific ACLs for their own users. However, the ***full*** privilege is very powerful and should not be granted unless absolutely necessary. A partner with ***full*** access could deny you access to files on your own system, or inadvertently ***deny*** access to everybody, both of which would require superuser intervention.

Both partners should retain ***allow*** access to the Acme `common_files` folder and its contents.

Finally, you (`acme_1`) want to have ***full*** privileges on all elements in the project. To set up these ACLs:

1. Log in to the AccuRev CLI with the `acme_1` superuser account.

   ```
   > accurev login acme_1
   ```

2. Change to the workspace directory for the prod_3000_devel project and grant yourself (`acme_1`) full access to all project files.

   ```
   >accurev eacl -a acme_1:full -R prod_3000_devel_files
   ```

   Note: Since we want to leave the default ***all:allow*** ACL in place for all the files in the project, and explicitly adjust the ACLs of specific files and specific users, it is critical that we use the ***-a*** (add) option here, and not ***-n*** (new). If we had specified ***-n***, user `acme_1` would have been granted ***full*** access, while *removing* access for everybody else. Removing access is the same as specifying ***deny***.

3. Explicitly deny both partners any access to the Acme proprietary elements:

   ```
   >accurev eacl -a part_1:deny,part_2:deny -R prod_3000_devel_files\acme_proprietary
   ```

Again, note the use of ***-a***, which retains the default ***allow*** access to these elements for other users (such as other Acme employees). Using ***-n*** here to set ***deny*** access to the partners would have been the same as setting ***all:deny***, since it would have removed the default ***all:allow*** setting.

4. Deny Partnership_1 any access to Partnership_2 files, while granting ***allow*** access to them to the Partnership_2 user (and retaining the default ***all:allow*** access to all other users).

```
>accurev eacl -a part_2:allow,part_1:deny -R
prod_3000_devel_files\partners\partner_2
```

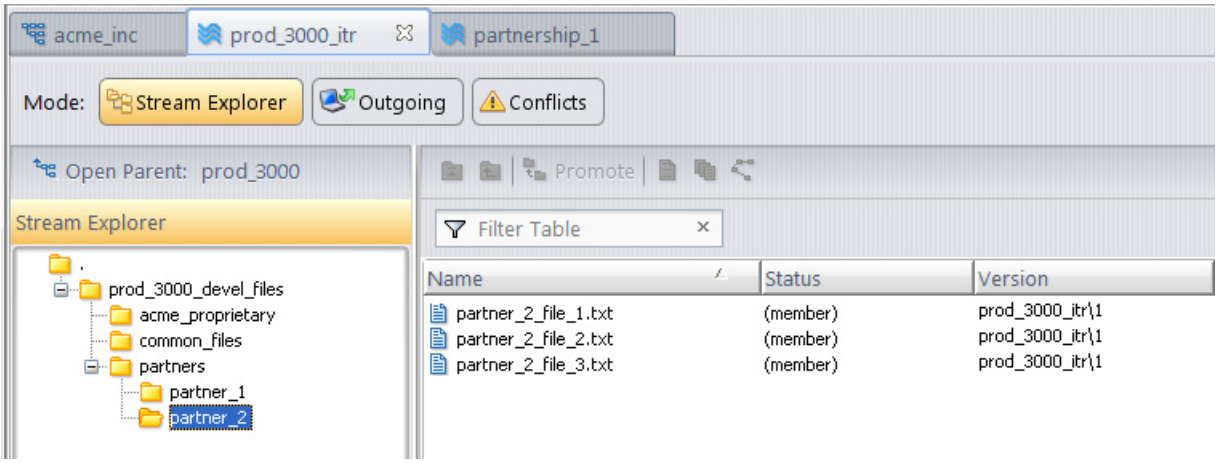
5. Deny Partnership_2 any access to Partnership_1 files, while granting ***allow*** access to them to the Partnership_1 user (and retaining the default ***all:allow*** access to all other users).

```
>accurev eacl -a part_1:allow,part_2:deny -R
prod_3000_devel_files\partners\partner_1
```

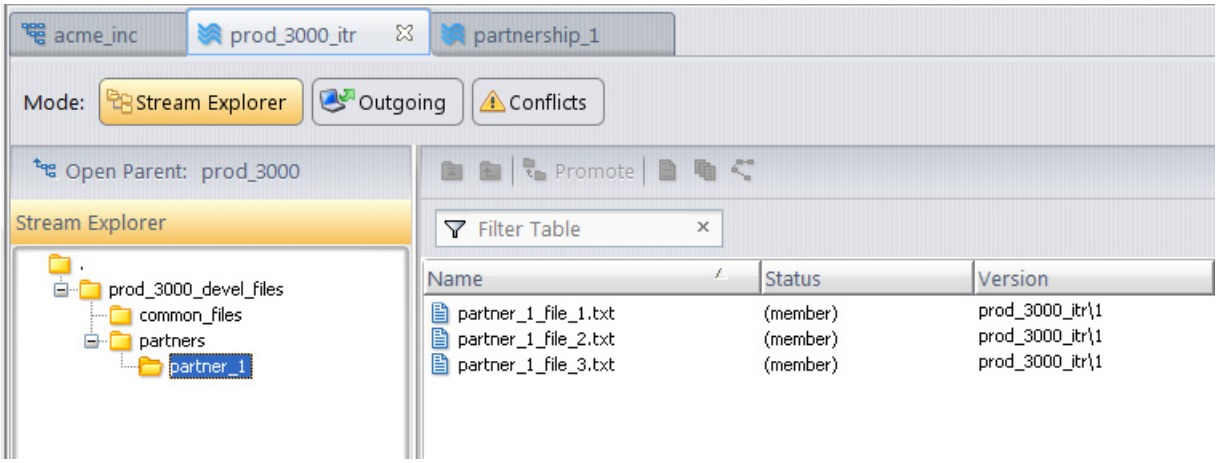
Once you (`acme_1`) promote all these elements up the stream, the effect of the ACLs becomes obvious.

Note: The ***promote*** has nothing to do with setting EACLs; it is just that the files were being set up for the first time in `acme_1`'s workspace and do not even exist in the stream hierarchy until they are promoted.

When you (`acme_1`, who has ***full*** accesss to all elements) view the contents of the prod_3000_itr stream, all folders and elements are visible. In the following example, the elements in partner_2 are displayed.



However, if user `part_1` logs in and views the same stream, he or she sees a very different display. Only the `partner_1` files and the common Acme files are visible. From user `part_1`'s perspective, the `partner_2` and Acme proprietary files and folders do not even exist.



User part_2 would see a similar display, except the partner_2 files and folder would be visible, while those of partner_1 could not be seen.

Note: Although the ACL setting is not visible from the GUI, an ACL list from the CLI will reveal that while user acme_1 has ***full*** access to the partner_1 elements and user ***part_2*** has no access to the partner_1 elements, user part_1 has **allow** access to the Acme common files through the default ***all*** ACL:

```
>accurev eacl common_files

/prod_3000_devel_files/common_files
  all:allow
  acme_1:full


>accurev eacl partners\partner_1

/prod_3000_devel_files/partners/partner_1
  all:allow
  acme_1:full
  part_1:allow
  part_2:deny
```

More Restrictive Scenario

This scenario is similar to the less restrictive scenario above, except that the default privilege for all users to all files will be ***deny***. We will then adjust ACLS to increase privileges as needed.

Similar to the previous scenario, we will start by giving user acme_1 full access to all project files, but this time we will use the ***-n*** option instead of ***-a***:

1. Log in to the AccuRev CLI with the acme_1 superuser account.

   ```
   > accurev login acme_1
   ```

2. Change to the workspace directory for the prod_3000_devel project and use the ***-n*** option to grant yourself (acme_1) full access to all project files, replacing the default ***all:allow*** ACL.

   ```
   >accurev eacl -n acme_1:full -R prod_3000_devel_files
   ```

   Note: Unlike the previous scenario which used ***-a***, all users except acme_1 are now denied access to all elements in the project by default.

3. Explicitly grant the user from each partnership ***allow*** access to the top of the project tree (but do not recurse through the tree structure, since we want to be selective about what they can and cannot access):

   ```
   >accurev eacl -a part_1:allow,part_2:allow prod_3000_devel_files
   ```

4. Now, provide the Partnership_1 user (part_1) and Partnership_2 user (part_2) ***allow*** access to the partners directory:

   ```
   accurev eacl -a part_1:allow,part_2:allow prod_3000_devel_files\partners
   ```

5. Explicitly grant the Partnership_1 user (part_1) ***allow*** access to the Partnership_1 files and to the Acme common files. Note that unlike the previous example, it is necessary now to explicitly grant allow access to the common files, since the ***all:allow*** default ACL was implicitly removed in Step 2.

   ```
   >accurev eacl -a part_1:allow -R prod_3000_devel_files\partners\partner_1

   >accurev eacl -a part_1:allow -R prod_3000_devel_files\common_files
   ```

6. Likewise, explicitly grant the Partnership_2 user (part_2) ***allow*** access to the Partnership_2 files and to the Acme common files.

   ```
   >accurev eacl -a part_2:allow -R prod_3000_devel_files\partners\partner_2
   ```

```
>accurev eacl -a part_2:allow -R prod_3000_devel_files\common_files
```

Because the default setting in this scenario is to deny access to users who have not been explicitly granted access, the above steps only address users acme_1, part_1, and part_2. You would need to continue these procedures to grant access to other Acme users as needed. In a real-life scenario, you would typically assign users to groups and assign ACLs to those groups rather than to individual users.

Restricting Access to Commands using Triggers

By default, any registered AccuRev user can execute any AccuRev command. Many organizations wish to restrict users' access to certain commands, such as the ability to maintain users, groups, and passwords, the ability to lock streams and create ACL permissions, and so on. Triggers provide a flexible mechanism for implementing command-based security.

Many AccuRev commands can be configured to "fire a trigger". This causes a user-defined script to execute either:

- before the command executes (pre-operation trigger), or afterward (post-operation trigger)
- on the client machine, or on the server machine

A pre-operation trigger can affect the execution of the command or cancel it altogether. Typically, a security-related trigger checks the identity of the user invoking the command, then decides whether or not to allow the command to proceed.

Some triggers are configured on a per-depot basis, using the ***mktrig*** command. These triggers monitor individual commands (***add***, ***keep***, and ***promote***). Three are pre-operation triggers that fire on the client machine; one is a post-operation trigger that fires on the server machine.

Other triggers are configured, on a per-depot or whole-repository basis, by placing a script in a well-known location on the server machine. These triggers monitor groups of commands, rather than individual commands.

For more on triggers, see the chapter *AccuRev Triggers* on page 77.

Which Security Feature Should I Use?

AccuRev's security features overlap to a considerable extent. For example, when a user invokes the ***promote*** command, any of these mechanisms can prevent the command from proceeding:

- a lock on the source or destination stream
- an ACL permission on the destination stream, on a higher-level stream, or on the entire depot
- a ***pre-promote-trig*** trigger script, which runs on the client machine
- a ***server_preop_trig*** trigger script, which runs on the server machine

How do you decide which feature to use in a given situation? There are no absolute rules, but here are some guidelines:

To script or not to script?

The trigger mechanism depends on execution of user-supplied scripts, written in Perl, Python, or some other scripting language. There's a trade-off: scripting required development time and significant expertise, but is infinitely flexible.

In many situations, you may be able to use the AccuRev software distribution's sample Perl scripts, which are designed for fill-in-the-blanks customization.

It makes sense to implement as much security as possible with locks and ACL permissions (and perhaps the sample trigger scripts), before delving into original trigger scripting.

Locks vs. ACL permissions

Roughly speaking, a lock controls the placing of data *into* a stream, whereas an ACL permission controls the reading of data *from* a stream. (There are plenty of exceptions to this general rule.)

Both locks and permissions can be targeted at specific users or groups. The ACL is more flexible: you can create any number of permissions for the same stream, but only limited number of locks.

Running trigger scripts: client machine vs. server machine

Running trigger scripts on the client machine conserves networking and server resources. But keep in mind that all client machines must have copies of the scripts (or must have network access to a central location where scripts are kept).

Running trigger scripts on the server machine provides administrative simplicity and centralized logging.

Implementing Secure Sockets Layer (SSL) Encryption

AccuRev supports the use of the Secure Sockets Layer protocol to provide encrypted communication between AccuRev clients and servers. The implementation of this protocol is managed through the use of SSL certificates. Communication between AccuRev servers and clients depends on an agreement (or *handshake*) between the AccuRev client and server that confirms a pairing of a *public key* and a *private key*.

In general, to configure SSL client-server communications, the AccuRev administrator must take the following actions:

1. Generate a private key.
2. Obtain an SSL certificate.
3. Enable SSL on the AccuRev server.

Each of these steps is described in more detail in the following sections.

Generating a Private Key

You can use an SSL toolkit (such as the one available at *https://www.openssl.org*) to generate a private key. When generating a private key, keep in mind the following points:

- AccuRev supports the use of RSA private keys
- AccuRev does *not* support the use of password-protected private keys
- AccuRev expects the private key file to have a .pem file extension (*.pem)
- AccuRev recommends that you generate a private key that is at least of 1024-bit strength

Once you have generated a private key, you can place the private key file anywhere on the server machine.

Note: For security purposes, AccuRev recommends that you set *read-only* permissions on the private key file.

Obtaining an SSL Certificate

Once you have generated a private key, you can obtain an SSL certificate. There are two ways to obtain an SSL certificate:

- Submit a request for an SSL certificate signed by a trusted authority (either by a trusted authority or by an intermediary)
- Create a self-signed certificate

AccuRev expects the SSL certificate file to have a .crt extension (*.crt).

When obtaining an SSL certificate, keep in mind that the same certificate can be used for multiple computers or computer names as long as they are specified in the *Subject Alternate Name* (SAN) field of the certificate. Once you have obtained an SSL certificate, you can place the SSL certificate file anywhere on the server.

AccuRev supports the use of both self-signed and trusted certificates. Regardless of which type of certificate you use, AccuRev prompts the user to accept the certificate the first time the user attempts to connect to an AccuRev server that has been SSL-enabled. If the user then accepts the certificate, the certificate is downloaded from the server to the client and stored in the user's profile directory which is, by default, the **.accurev** directory. (The location of the user's profile directory is determined by the current setting of the USERPROFILE environment variable in Windows and UNIX while, on Linux platforms, the location is in the **/home** directory.) The user can then connect to that server in future sessions without being prompted to accept that certificate again, unless the certificate expires or SSL is disabled on the server. If, however, the user should attempt to connect to a different AccuRev server that has been SSL-enabled, the user is also prompted to accept the certificate from that server.

Note: Additional configuration is required if you are using a certificate from a trusted authority. See *Considerations for Using Trusted Certificates* on page 71 for more information.

Considerations for Using Trusted Certificates

If you are using an SSL certificate from a trusted authority, you need to include the actual certificates in the certificate chain. You do this by appending intermediate and root certificates to the trusted SSL certificate as described here and as shown in the following example.

When adding intermediate and root certificates, note the following:

- The trusted authority's certificate must be the first one in the certificate file, followed by all intermediate certificates, and ending with the root certificate.
- Each certificate must be in Base64 encoded format (that is, PEM format).
- Each certificate must be preceded with `-----BEGIN CERTIFICATE-----` and followed with `-----END CERTIFICATE-----` on their own lines. These demarcations are present in each certificate and should not be removed.
- When editing the certificate file, make sure the editor you use recognizes non-standard characters like UNIX end-of-line characters. When you are done appending intermediate and root certificates, the format of your certificate file should resemble that shown in the following example.

```
-----BEGIN CERTIFICATE-----
MIIE0TCCA7mgAwIBAgIQLOmmgpHvkgfaMxBKj9tPojANBgkqhkiG9w0BAQUFADA8
```

```
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMVGhhd3RlLCBJbmMuMRYwFAYDVQQDEw1U
aGF3dGUgU1NMIENBMB4XDTE0MDMxMTAwMDAwMFoXDTE1MDMxMTIzNTk1OVowcDEL
MAkGA1UEBhMCVVMxFjAUBgNVBAgTDU1hc3NhY2h1c2V0dHMxEDAOBgNVBAcUB0Nv
bmNvcmQxFDASBgNVBAoUC0FjY3VSZHYgSW5jMSEwHwYDVQQDFBhzZWNyZXR3ZWFw
b24uYWNjdXJldi5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC5
r5A70M9l6B4alO2A9IKbWLtZ0Tm7xeWJ4SIE1sypxnba3njaN3bx0CXNvhxWP+68
VWRhfaSemti+c/BQ5aRhp6kBUYGQ6NLKtycmME3N+HgwpkUylHkEH7gnUgyCUu7L
PMPJgg3/6WByRykzpHHgX0vCX8EbUaWxsMaglQojJsOUm+L6eTcRBQpywJOidoMQ
hJ6pjlg7AzK8qqMNSpBbG8CrMQMmnixY5qGJ28I3WtJu/LNHAyrPxYq1bNJbsBtM
mYXI6/IHiq0u2ogPnvr5Ud8CV/VRZ6Mdaxllw7o2yfNjncw6dnP6dbHA7g8t0qPp
tXg+3ISiBXOQpVPCC9flAgMBAAGjggGZMIIBlTAJBgNVHRMEAjAAMEIGA1UdIAQ7
MDkwNwYKYIZIAYb4RQEHNjApMCcGCCsGAQUFBwIBFhtodHRwczovL3d3dy50aGF3
dGUuY29tL2Nwcy8wDgYDVR0PAQH/BAQDAgWgMB8GA1UdIwQYMBaAFKeig7s0RUA9
/NUwTxK5PqEBn/bbMDoGA1UdHwQzMDEwL6AtoCuGKWh0dHA6Ly9zdnItb3YtY3Js
LnRoYXd0ZS5jb20vVGhhd3RlT1YuY3JsMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggr
BgEFBQcDAjBpBggrBgEFBQcBAQRdMFswIgYIKwYBBQUHMAGGFmh0dHA6Ly9vY3Nw
LnRoYXd0ZS5jb20wNQYIKwYBBQUHMAKGKWh0dHA6Ly9zdnItb3YtYWlhLnRoYXd0
ZS5jb20vVGhhd3RlT1YuY2VyME0GA1UdEQRGMESCGnNlY3JldHdlYXBvbi5hY2N1
cmV2LmxvY2FsggxzZWNyZXR3ZWFwb26CGHNlY3JldHdlYXBvbi5hY2N1cmV2LmNv
bTANBgkqhkiG9w0BAQUFAAOCAQEAC2Xw1KcxXLCd1ffy3OdOBbopfrqFGeK9k3Kg
LaLZfdmc8Ncg996abVh2phDl+O9eKNIpMKBESvyD6DWGEoUBYO4Ql13+tHKnFfdb
jxxQHgCF1diyZmEx4buMGOoOkmMZay+FlUJxk3U1c0vqpNpQiXggWsqvT8++UIQc
2l3O56IKHGeWoe/ITvwzqn55XxfOLIyOys8Xy1//8N1lT5FLG/RdWo3eSJgU+vyv
FTN3xRlwWGxIJIWVg1efxgRgFO8b6PvelV5C4xGaMPzEsRYDcve7wZK+S8XFTCQi
WXkazTQiEdwFO86RmBUdkfC9+ukWowp6GFLI4checd/AcuOSdA==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEbDCCA1SgAwIBAgIQTV8sNAiyTCDNbVB+JE3J7DANBgkqhkiG9w0BAQUFADCB
qTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDHRoYXd0ZSwgSW5jLjEoMCYGA1UECxMf
Q2VydGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjE4MDYGA1UECxMvKGMpIDIw
MDYgdGhhd3RlLCBJbmMuICOgRm9yIGF1dGhvcml6ZWQgdXNlIG9ubHkxHzAdBgNV
BAMTFnRoYXd0ZSBQcmltYXJ5IFJvb3QgQ0EwHhcNMTAwMjA4MDAwMDAwWhcNMjAw
MjA3MjM1OTU5WjA8MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMVGhhd3RlLCBJbmMu
MRYwFAYDVQQDEw1UaGF3dGUgU1NMIENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAmeSFW3ZJfS8F2MWsyMip09yY5tc0pi8M8iIm2KPJFEyPBaRF6BQM
WJAFGrfFwQalgK+7HUlrUjSIw1nn72vEJ0GMK2Yd0OCjl5gZNEtB1ZjVxwWtouTX
7QytT8G1sCH9PlBTssSQ0NQwZ2ya8Q50xMLciuiX/8mSrgGKVgqYMrAAI+yQGmDD
7bs6yw9jnw1EyVLhJZa/7VCViX9WFLG3YR0cB4w6LPf/gN45RdWvGtF42MdxaqMZ
pzJQIenyDqHGEwNESNFmqFJX1xG0k4vlmZ9d53hR5U32t1m0drUJN00GOBN6HAiY
XMRISstSoKn4sZ2Oe3mwIC88lqgRYke7EQIDAQABo4H7MIH4MDIGCCsGAQUFBwEB
BCYwJDAiBggrBgEFBQcwAYYWaHR0cDovL29jc3AudGhhd3RlLmNvbTASBgNVHRMB
Af8ECDAGAQH/AgEAMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcmwudGhhd3Rl
LmNvbS9UaGF3dGVQQ0EuY3JsMA4GA1UdDwEB/wQEAwIBBjAoBgNVHREEITAfpB0w
GzEZMBcGA1UEAxMQVmVyaVNpZ25NUEtJLTItOTAdBgNVHQ4EFgQUp6KDuzRFQD38
1TBPErk+oQGf9tswHwYDVR0jBBgwFoAUe1tFz6/Oy3r9MZIaarbzRutXSFAwDQYJ
KoZIhvcNAQEFBQADggEBAIAigOBsyJUW11cmh/NyNNvGclYnPtOW9i4lkaU+M5en
```

```
S+Uv+yV9Lwdh+m+DdExMU3IgpHrPUVFWgYiwbR82LMgrsYiZwf5Eq0hRfNjyRGQq
2HGn+xov+RmNNLIjv8RMVR2OROiqXZrdn/0Dx7okQ40tR0Tb9tiYyLL52u/tKVxp
EvrRI5YPv5wN8nlFUzeaVi/oVxBw9u6JDEmJmsEj9cIqzEHPIqtlbreUgm0vQF9Y
3uuVK6ZyaFIZkSqudZ1OkubK3lTqGKslPOZkpnkfJn1h7X3S5XFV2JMXfBQ4MDzf
huNMrUnjl1nOG5srztxl1AsoaO6ERlFE9zMILViXIa4=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEIDCCAwigAwIBAgIQNE7VVyDV7exJ9C/ON9srbTANBgkqhkiG9w0BAQUFADCB
qTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDHRoYXd0ZSwgSW5jLjEoMCYGA1UECxMf
Q2VydGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjE4MDYGA1UECxMvKGMpIDIw
MDYgdGhhd3RlLCBJbmMuICOgRm9yIGF1dGhvcml6ZWQgdXNlIG9ubHkxHzAdBgNV
BAMTFnRoYXd0ZSBQcmltYXJ5IFJvb3QgQ0EwHhcNMDYxMTE3MDAwMDAwWhcNMzYw
NzE2MjM1OTU5WjCBqTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDHRoYXd0ZSwgSW5j
LjEoMCYGA1UECxMfQ2VydGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjE4MDYG
A1UECxMvKGMpIDIwMDYgdGhhd3RlLCBJbmMuICOgRm9yIGF1dGhvcml6ZWQgdXNl
IG9ubHkxHzAdBgNVBAMTFnRoYXd0ZSBQcmltYXJ5IFJvb3QgQ0EwggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQCsoPD7gFnUnMekz52hWXMJEEUMDSxuaPFs
W0hoSVk3/AszGcJ3f8wQLZU0HObrTQmnHNK4yZc2AreJ1CRfBsDMRJSUjQJib+ta
3RGNKJpchJAQeg29dGYvajig4tVUROsdB58Hum/u6f1OCyn1PoSgAfGcq/gcfomk
6KHYcWUNo1F77rzSImANuVud37r8UVsLr5iy6S7pBOhih94ryNdOwUxkHt3Ph1i6
Sk/KaAcdHJ1KxtUvkcx8cXIcxcBn6zL9yZJclNqFwJu/U3OrCfSMnZEfl2pSy94J
NqR32HuHUETVPm4pafs5SSYeCaWAe0At6+gnhcn+Yf1+5nyXHdWdAgMBAAGjQjBA
MA8GA1UdEwEB/wQFMAMBAf8wDgYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBR7W0XP
r87Lev0xkhpqtvNG61dIUDANBgkqhkiG9w0BAQUFAAOCAQEAeRHAS7ORtvzw6WfU
DW5FvlXok9LOAz/t2iWwHVfLHjp2oEzsUHboZHIMpKnxuIvW1oeEuzLlQRHAd9mz
YJ3rG9XRbkREqaYB7FViHXe4XI5ISXycO1cRrK1zN44veFyQaEfZYGDm/Ac9IiAX
xPcW6cTYcvnIc3zfFi8VqT79aie2oetaupgf1eNNZAqdE8hhuvU5HIe6uL17In/2
/qxAeeWsEG89jxt5dovEN7MhGITlNgDrYyCZuen+MwS7QcjBAvlEYyCegc5C09Y/
LHbTY5xZ3Y+m4Q6gLkH3LpVHz7z9M/P2C2F+fpErgUfCJzDupxBdN49cOSvkBPB7
jVaMaA==
-----END CERTIFICATE-----
```

Enabling SSL Encryption on the AccuRev Server

After you have obtained an SSL certificate, ensure that you have placed both the private key file and the SSL certificate file in any location on the AccuRev server before attempting to enable SSL encryption.

To enable SSL encryption on an AccuRev server, add the following three parameters to the **acserver.cnf** file:

- `SSL_ENABLED = TRUE`

  Enable the server for SSL encryption by setting the **SSL_ENABLED** parameter to **TRUE**.

- `SSL_CERTIFICATE = C:\Program Files (x86)\AccuRev\bin\ServerCert\AccuRev.crt`

  In this example, *C:\Program Files (x86)\AccuRev\bin\ServerCert\AccuRev.crt* represents the absolute path to the server's certificate file, **AccuRev.crt**. This path name cannot contain quotes.

- `SSL_PRIVATE_KEY = C:\Program Files (x86)\AccuRev\bin\ServerCert\AccuRev.pem`

In this example, C:\Program Files (x86)\AccuRev\bin\ServerCert\AccuRev.pem represents the absolute path to the server's private key file, **AccuRev.pem**. This path name cannot contain quotes.

After editing the **acserver.cnf** file, you must restart the AccuRev server to complete the process of encrypting communication between the server and its clients.

Managing SSL

The following sections describe methods of managing the process of SSL encryption in AccuRev.

Using the --thumbprint Option

In public key encryption, a certificate's thumbprint (also known as a "fingerprint") is the SHA1 hash of the binary representation of the certificate converted to a hexadecimal string; it is this string that is used to authenticate a longer public key. The **--thumbprint** option allows you to specify the certificate's thumbprint which, if it matches that of the SSL certificate on the AccuRev server, allows the certificate to be accepted automatically. This option is available for both the **enable_ssl** command and the **get_certificate** command. This feature is particularly useful in situations where a user is not present to accept an SSL certificate.

The **--thumbprint** option can be used to enable SSL on unattended machines by using a script that executes a command, for example, such as the following:

```
accurev enable_SSL --thumbprint="30 9b 7a f1 44 5f 8b 1f ac 7b 6f 8b aa bc 3f 7b b6 56
da c9"
```

For more information about the **thumbprint** command, refer to the descriptions of the **get_certificate** and **enable_ssl** commands in the AccuRev CLI Help.

Implementing SSL for Replicas

If replicas are being used, SSL encryption must be enabled on all machines. This means that:

- The master server must be SSL-enabled
- All clients must be SSL-enabled
- All replicas must be SSL-enabled as *both* a client (to the master server) and as a server (to all clients)

Replacing an Expired Certificate

If an SSL certificate expires, the AccuRev administrator must obtain a new SSL certificate. The existing private key can be used to obtain a new SSL certificate as long as its security has not been compromised. Otherwise, the administrator can generate a new private key and use that to obtain a new SSL certificate.

If the certificate name or location on the server has changed, the **SSL_CERTIFICATE** parameter of the **acserver.cnf** file must be updated to reflect the new file name or file path. Likewise, the **SSL_PRIVATE_KEY** parameter of the **acserver.cnf** file must be updated if the private key file name or location has been changed.

Disabling SSL Encryption on the AccuRev Server

To disable SSL encryption on the server, set the **SSL_ENABLED** parameter to **FALSE** in the **acserver.cnf** file:

```
SSL_ENABLED = FALSE
```

You could also delete this parameter or comment it out to disable SSL encryption on the server.

After editing the **acserver.cnf** file, you must restart the server to complete the process of disabling SSL.

When an SSL-enabled client attempts to connect to this server, the user is prompted to disable SSL on the client or exit the interface.

10. AccuRev Triggers

A trigger is a callback built into certain AccuRev commands. When a user enters the command, the corresponding trigger causes a user-defined or built-in procedure to be performed just before (pre-operation) or after (post-operation) 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.

> **Note**: In this chapter, "trigger script" refers to any executable program, written in any language, that is executed when a trigger fires.

AccuRev supports pre-operation triggers, post-operation triggers, and triggers that integrate issue management into AccuRev's configuration management. The latter category of triggers have pre- and post-operation components.

You set some triggers with the ***mktrig*** command; you set others by placing the trigger script at a special location; yet others are set through the Schema Editor in the AccuRev GUI.

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. (See *Trigger Script Exit Status* on page 99.) 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. It does not fire for an ***ln*** command (GUI: ***Paste Link***), which creates a link element.)

  The trigger script must specify the element type (directory, text, binary, or ptext) of each element to be created by the command. This overrides the element type specified with the ***add –E*** option.

- ***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 or a ***purge*** command (GUI: ***Revert to Basis Version*** or ***Revert to Most Recent Version***).

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. The following commands cause ***server_admin_trig*** to fire:

addmember	chstream	mkgroup	mkws	rmws
authmethod	chuser	mkref	reactivate	setacl

chdepot *(see note)*	chws *(see note)*	mkreplica	remove	setproperty
chgroup	ismember	mksnap *(see note)*	rmmember	unlock
chpassword	lock	mkstream	rmproperty	*defcomp*
chref	lsacl	mktrig	rmreplica	*replica_sync*
chslice	mkdepot	mkuser	rmtrig	*write_schema*

Note: The ***chdepot*** and ***chws*** commands cause the ***server_admin_trig*** to fire twice:

- ***chdepot***: ***server_admin_trig*** fires once to rename the depot object (chdepot), then again to rename the depot's root-stream object (chstream).
- ***chws***: ***server_admin_trig*** fires once to rename the workspace object (chws), then again to rename the workspace stream object (chstream).

It should also be noted that the ***mksnap*** command is an *alias* for the ***mkstream*** command. It calls the ***server_admin_trig*** once, to create the stream object (mkstream) with a type of snapshot.

The last three commands are not standard AccuRev CLI commands:

- The ***defcomp*** command is not visible to the user; it is used in the implementation of the include/exclude facility CLI commands ***incl***, ***excl***, ***incldo***, and ***clear***.
- The ***replica_sync*** command recognized by the ***server_admin_trig*** trigger corresponds to the ***replica sync*** command in the CLI.
- The ***write_schema*** command is generated by the AccuRev GUI when the ***Save*** button is clicked in the Schema Editor.

- ***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. The following commands cause ***server_preop_trig*** to fire:

add	cpkremove	promote	revert
archive	defunct	purge *(see note)*	unarchive
co	keep	reclaim	
cpkadd	ln	replica archive	

Note: ***purge*** is equivalent to the GUI command ***Revert to Basis***.

For ***add*** or ***keep***, the ***server_preop_trig*** script can specify the exclusive file locking state (parallel or serial) of the element(s) processed by the command. This overrides any specification made with the ***–E*** command-line option.

The ***server_admin_trig*** and ***server_preop_trig*** triggers are independent of each other and are fired by different sets of commands — for a given command, only one of these triggers will fire. But these triggers can fire in addition to the triggers enabled with the ***mktrig*** command (***pre-create-trig***, ***pre-keep-trig***, ***pre-promote-trig***, and ***server-post-promote-trig***) and the ***server_dispatch_post*** trigger.

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.
- ***server_dispatch_post***: fires on the server machine each time an AccuWork issue record is created or modified. This trigger is intended to enable email notification to interested users. A sample Perl script is available in the **examples/dispatch** subdirectory of the AccuRev installation directory.

server_master_trig

The ***server_master_trig*** trigger is a repository-wide server-side trigger. It is used mostly to provide customized behavior for elements promoted to staging streams in gated stream implementations. See *Chapter 11 Using Streams to Enforce Process* for more information.

Triggers in a Replication Environment

See *Triggers and Replication* on page 51.

Triggers and Security

See *The 'server_auth_trig' Script* on page 58.

Preparing to Use 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 sources 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.

   For faster processing on large XML files, try installing the **XML::SAX::ExpatXS** Perl module and using that module in your Perl scripts.

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. Before embarking on complex script customization, be sure to

read *The Trigger Parameters File* on page 82.

Note: The path to the AccuRev executable in the script must not contain unquoted spaces. Either properly quote the spaces for Perl, or use short filenames on Windows platforms (for example, **progra~1** if **accurev.exe** is located in the **Program Files** directory).

4. **Enable the trigger.** Enable the trigger, either with the ***mktrig*** command or by placing the script in the proper location. See *Enabling a Trigger* below for details.

Enabling a Trigger

Depending on its type, an AccuRev trigger is enabled in one of these ways:

- Executing an ***accurev mktrig*** command, specifying the location of the script. AccuRev simply records the location you specify in the repository; it doesn't make a copy of the script. Make sure that no one moves it!
- Placing the executable script file in the location prescribed for that type of trigger.

For details, consult the appropriate subsection below:

pre-create-trig, pre-keep-trig, pre-promote-trig, server-post-promote-trig

Use the ***mktrig*** command to enable use of the script in a particular depot. For example:

```
accurev mktrig -p WidgetDepot pre-keep-trig /usr/ac_scripts/addheader
```

The ***–p*** option isn't necessary if your current directory is in a workspace associated with that depot. When the trigger fires, AccuRev will search for the script at the specified pathname (in the example above, **/usr/ac_scripts/addheader**).

AccuRev recommends specifying an absolute pathname for the **server-post-promote-trig** only. Otherwise, when the trigger fires, AccuRev will use the search path of the AccuRev Server's user identity to find the specified script file.

You should not specify absolute pathnames for the **pre-create-trig**, **pre-keep-trig**, or **pre-promote-trig**. In this case, AccuRev uses the user's search path to find the specified script file.

server_admin_trig, server_master_trig

Place an executable file in subdirectory **triggers** of the **site_slice** directory:

- UNIX/Linux: the file must be named **server_admin_trig** or **server_admin_trig.pl** (or **server_master_trig** or **server_master_trig.pl**)
- Windows: the file must be named **server_admin_trig.bat** (or **server_master_trig.bat**)

Example:

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

server_preop_trig

Place an executable file in subdirectory **triggers** of the slice directory of one or more depots (***accurev show slices*** displays slice directory locations):

- UNIX/Linux: 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

Place an executable file in the AccuRev executables (**bin**) directory on the AccuRev Server machine:

- UNIX/Linux: the file must be named **server_dispatch_post** or **server_dispatch_post.pl**
- Windows: the file must be named **server_dispatch_post.bat**

  Note: for compatibility with previous AccuRev releases, the script can also be named **dispatch_email**, with the appropriate suffix.

Example:

```
C:\Program Files\AccuRev\bin\server_dispatch_post.bat
```

Notes on Triggers in Multiple-Depot Environments

If you have multiple depots, you may wonder whether to use a single trigger for all depots, or separate triggers for each depot. There is no single correct approach, as it depends on what you are attempting to accomplish, and what level of granularity you need to achieve. For example, the basic administrative features of the **server_admin_trig** trigger might apply to all depots. If you configure some of its other features, you might need to configure it for specific depots. On the other hand, it is likely that you would want to implement separate **server_preop_trig** triggers for each depot.

It all depends on what you are attempting to accomplish. Carefully plan out what features you need and determine whether or not they will be affected by the requirements of different depots. Then implement as necessary.

Notes on Triggers in Multiple-Platform Environments

If you have a mixed environment where you have both Windows and UNIX/Linux clients accessing the same depot, you can set up triggers that will operate for both.

- Create trigger scripts that have the exact same name for both platforms, but ensure that the Windows version has the appropriate extension (e.g., **check_for_comments.bat**), and that the UNIX/Linux script has *no* extension (**check_for_comments**).
- Test them and ensure that the scripts run correctly on their respective platforms.
- Place both versions of the script in the same directory. (You can place the Windows and UNIX/Linux scripts in separate directories if you need to, but we recommend keeping them together for simplicity and ease of maintenance.) This directory needs to be visible to all clients--it needs to be exported or shared from the server, and mapped or mounted on the clients.
- Adjust the PATH on your clients to point to the proper directory for their platform. For example, the PATH setting "`S:\triggers`" on Windows clients might be "`/mnt1/ac/triggers`" on UNIX/Linux machines.

- Execute the ***mktrig*** command, specifying the script name without a suffix, and without a qualifying path:

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

  When called by a Windows client, the trigger script with an extension will get executed. When called by a UNIX/Linux client, the trigger script without the extension will get executed.

Remember to revise *all* versions of a script when you revise any one of them.

The Trigger Parameters File

When a trigger fires and executes a user-supplied script, AccuRev passes two arguments to the script:

- The first argument is the pathname of a flat-text file containing information about the transaction that is about to be performed (or was just completed).
- The second argument is the pathname of an XML-format file containing the same information. (In some cases, detailed below, the XML-format file contains a small amount of additional information that is not contained in the flat-text file.)

Exceptions: only one argument, the pathname of an XML-format file, is passed to a **server_preop_trig** script or a **server_admin_trig** script.

These files are called *trigger parameters files*. The flat-text file contains a series of values — usually one value per line — in a prescribed order. The XML-format file contains a set of elements below the top-level **<triggerInput>** element. Each element contains the information for one parameter: the parameter name is the element tag, the parameter value is the element contents (sometimes encoded as a set of subelements). For example, here are two trigger parameters files generated by the same user command:

Flat-text trigger parameters file	XML-format trigger parameters file
pre-create talon talon_dvt_john 4 adding some files this multi-line comment has four lines C:/wks/talon/dvt_john john /tools/cont.sh /tools/end.sh /tools/start.sh	<triggerInput> <hook>pre-create</hook> <depot>talon</depot> <stream1>talon_dvt_john</stream1> <changePackages></changePackages> <comment>adding some files this multi-line comment has four lines</comment> <topDir>C:/wks/talon/dvt_john</topDir> <principal>john</principal> <elemList> <elem>/tools/cont.sh</elem> <elem>/tools/end.sh</elem> <elem>/tools/start.sh</elem> </elemList> </triggerInput>

The following sections describe how information contained in the trigger parameters file varies among the trigger types:

- *Format of the "pre-create-trig" Trigger Parameters File* on page 83
- *Format of the "pre-keep-trig" Trigger Parameters File* on page 84
- *Format of the "pre-promote-trig" Trigger Parameters File* on page 86
- *Format of the "server-post-promote-trig" Trigger Parameters File* on page 87
- *Format of the "server_preop_trig" Trigger Parameters File* on page 88
- *Format of the "server_admin_trig" Trigger Parameters File* on page 94
- *Format of the "server_dispatch_post" Trigger Parameters File* on page 96

Format of the "pre-create-trig" Trigger Parameters File

The following table presents the information in the trigger parameters file sent to a **pre-create-trig** script. This information describes the creation of one or more new elements to a depot (***add***).

> **Note**: The trigger fires on creation of a new file or directory element, but not on creation of a new link element (CLI: ***ln***, GUI: ***Paste Link***).

The order of the parameters in this table reflects the order in which they appear in the flat-text trigger parameters file.

Parameter	Description
hook	Type of trigger: **pre-create**.
principal	AccuRev username of person invoking the command.
depot	Name of depot targeted by the command.
stream1	Name of the workspace stream in which the new elements are to be created.
topDir	Pathname to the top-level directory of the user's workspace tree, as it is listed by the ***show wspaces*** command.
comment	Zero or more comment lines specified by the user (see *Encoding of Command Comments* on page 98 below).
elemType	*(XML-format parameters file only)* Integer indicating the element type specified by the user: **0**=none specified, **2**=text, **3**=binary, **4**=ptext.
elemList elements	One or more files/directories to be added to the depot. For general notes, see *Encoding of Element Lists* on page 96 below.

In the flat-text trigger parameters file, the elements to be created are listed, one per line, at the end of the file:

```
/tools/cont.sh
/tools/end.sh
/tools/start.sh
```

<-- end-of-file of trigger parameters file

There is no need to supply an element count, since an end-of-file condition signals the end of the element list.

In the XML-format trigger parameters file, the element paths are encoded as **<elem>** sub-elements of **<elemList>**:

```
<elemList>
    <elem>/tools/cont.sh</elem>
    <elem>/tools/end.sh</elem>
    <elem>/tools/start.sh</elem>
</elemList>
```

(There is also an **<elements>** element with the same **<elem>** sub-element data.)

Overwriting the 'pre-create-trig' Trigger Parameters File

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

```
<element-pathname>  <element-type>
```

... where <*element-pathname*> is a pathname from the input "elemList", and <*element-type*> is a numeric code:

1 directory
2 text file
3 binary file
4 ptext file

Make sure that the element-type value is 1 for every directory in the original list. You can't change the element-type of a directory! You can however, change among the text-file, binary-file, and ptext-file types. For example, you might override AccuRev's default classification of file **ReadMe.html** as 2 (text-file), setting the element-type to 3 (binary-file) instead.

See *Controlling the Element Type and Exclusive File Locking State* on page 38 of the *AccuRev CLI User's Guide* for a discussion of element types.

Example: to have an ***add*** command create two text-file elements, two binary-file elements, and a directory element, a ***pre-create-trig*** script would replace its flat-text parameters file with this data:

```
/tools/end.sh 2
/tools/icons 1
/tools/icons/end.png 3
/tools/icons/start.png 3
/tools/start.sh 2
```

Note: there is currently no provision for the script to overwrite the XML-format trigger parameters file. The data to be passed to the AccuRev Server must be in flat-text format.

Format of the "pre-keep-trig" Trigger Parameters File

The following table presents the information in the trigger parameters file sent to a **pre-keep-trig** script. This information describes the creation of a new versions of one or more existing elements in a depot (CLI: ***keep***, GUI: ***Keep***).

The order of the parameters in this table reflects the order in which they appear in the flat-text trigger parameters file.

Parameter	Description
hook	Type of trigger: **pre-keep**.
principal	AccuRev username of person invoking the command.
depot	Name of depot targeted by the command.
stream1	Name of the workspace stream in which the new versions are to be created.
topDir	Pathname to the top-level directory of the user's workspace tree, as it is listed by the ***show wspaces*** command.
comment	Zero or more comment lines specified by the user (see *Encoding of Command Comments* on page 98 below).
elemType	*(XML-format parameters file only)* Integer indicating the element type specified by the user: **0**=none specified, **2**=text, **3**=binary, **4**=ptext.
elemList elements	A specification for each new element version to be created. For general notes, see *Encoding of Element Lists* on page 96 below.

In the flat-text trigger parameters file, the versions to be created are listed, one per line, at the end of the file. Each line contains three specifications:

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

There is no need to supply an element count, since an end-of-file condition signals the end of the element list. For example:

```
/tools/icons/end.png talon_dvt_john/5 3
/tools/icons/end.sh talon_dvt_john/9 2
/tools/icons/start.png talon_dvt_john/2 3
/tools/icons/start.sh talon_dvt_john/13 2
```

In the XML-format trigger parameters file, each version to be created is encoded as an as **<elem>** sub-element of **<elemList>**. The element's attributes specify the version-ID (**stream** and **version** attributes) and the element-type (**elemType** attribute). The element pathname is encoded as the contents of **<elem>**.

The following example contains the same data as the flat-text example above:

```
<elemList>
  <elem
      stream="talon_dvt_john"
      version="5"
      elemType="3">/tools/icons/end.png</elem>
  <elem
      stream="talon_dvt_john"
      version="9"
      elemType="2">/tools/icons/end.sh</elem>
  <elem
      stream="talon_dvt_john"
      version="2"
      elemType="3">/tools/icons/start.png</elem>
  <elem
```

```
        stream="talon_dvt_john"
        version="13"
        elemType="2">/tools/icons/start.sh</elem>
</elemList>
```

In either format, the element-type value can be either **2** (text file), **3** (binary file), or **4** (ptext file). Note that different versions of an element can have different types.

In addition to the **<elemList>** element, the parameters file includes an **<elements>** element, with additional information on each file: its element-ID and the real version-ID of the workspace's *current* version (not the one about to be created):

```
...
<elem
    eid="58"
    ver="3/4">/tools/icons/end.png</elem>
...
```

Format of the "pre-promote-trig" Trigger Parameters File

The following table presents the information in the trigger parameters file sent to a **pre-promote-trig** script. This information describes the creation of a new versions of one or more existing elements in a depot (CLI: ***promote***, GUI: ***Promote***).

> Note: the **pre-promote-trig** trigger also fires on execution of a CLI ***purge*** command (GUI: ***Revert to Basis***) — but only when the version is being purged from a dynamic stream, not a workspace.

The order of the parameters in this table reflects the order in which they appear in the flat-text trigger parameters file.

Parameter	Description
hook	Type of trigger: **pre-promote**.
principal	AccuRev username of person invoking the command.
depot	Name of depot targeted by the command.
stream1	Name of the workspace or stream that the versions are to be promoted from.
action	*(XML-format parameters file only)* (purge) The character string **purge**.
option_I	*(XML-format parameters file only)* The issue number (or numbers, SPACE-separated) specified by the user with the ***promote –I*** command-line option.
stream2	*(XML-format parameters file only)* Name of the stream that the versions are to be promoted to.
topDir	Pathname to the top-level directory of the user's workspace tree, as it is listed by the ***show wspaces*** command.
comment	Zero or more comment lines specified by the user (see *Encoding of Command Comments* on page 98 below).
elemType	*(XML-format parameters file only)* Integer indicating the element type specified by the user: **0**=none specified, **2**=text, **3**=binary, **4**=ptext.

Parameter	Description
elemList elements	One or more files/directories to be promoted. For general notes, see *Encoding of Element Lists* on page 96 below.

In the flat-text trigger parameters file, the elements to be created are listed, one per line, at the end of the file:

```
/tools/cont.sh
/tools/end.sh
/tools/start.sh
                    <-- end-of-file of trigger parameters file
```

There is no need to supply an element count, since an end-of-file condition signals the end of the element list.

In the XML-format trigger parameters file, the element paths are encoded as **<elem>** sub-elements of **<elemList>**:

```
<elemList>
    <elem>/tools/cont.sh</elem>
    <elem>/tools/end.sh</elem>
    <elem>/tools/start.sh</elem>
</elemList>
```

In addition to the **<elemList>** element, the parameters file includes an **<elements>** element, with additional information on each element: its element-ID and the real version-ID of the version to be promoted):

```
  <elements>
  <elem
      eid="51"
      ver="8/13">/tools/cont.sh</elem>
  </elements>
```

Overwriting the 'pre-promote-trig' Trigger Parameters File

A ***pre-promote-trig*** script can work in tandem with a ***server-post-promote-trig*** script, providing customized "before and after" processing around the execution of ***Promote*** commands:

- The ***pre-promote-trig*** script overwrites its flat-text triggers parameters file.
- The *first line* of the overwritten parameters file becomes the value of the <*fromClientPromote*> parameter passed to the ***server-post-promote-trig*** script.

  Note: there is currently no provision for a ***pre-promote-trig*** script to pass data to a ***server-post-promote-trig*** script by overwriting the XML-format trigger parameters file.

Format of the "server-post-promote-trig" Trigger Parameters File

The following table presents the information in the trigger parameters file sent to a **server-post-promote-trig** script. This information is generated by AccuRev, and describes the ***Promote*** command that has just executed. The first line of this file provides a mechanism for passing user-specified data from a ***pre-promote-trig*** script to a ***server-post-promote-trig*** script. See *Overwriting the 'pre-promote-trig' Trigger Parameters File* on page 87.

The order of the parameters in this table reflects the order in which they appear in the flat-text trigger parameters file.

Parameter	Description
hook	Type of trigger: **server-post-promote**.
principal	AccuRev username of person invoking the command.
depot	Name of depot targeted by the command.
stream1	Name of the stream that the versions were promoted to.
source_stream	Name of the stream the versions were promoted from.
dest_stream	Name of the stream the versions were promoted to.
comment	Zero or more comment lines specified by the user in the ***Promote*** command (see *Encoding of Command Comments* on page 98 below).
fromClientPromote	A single text line containing the issue number (or numbers, SPACE-separated) specified by the user during the ***Promote*** command. If no issue number was specified, the flat-text parameters file contains a blank line and the XML-format file contains an empty element.
transNum	The transaction number of the ***Promote*** transaction that just completed.
transTime	The time of the ***Promote*** transaction that just completed.
changePackages	*(XML-format parameters file only)* A set of <changePackageID> subelements, specifying the same information as <fromClientPromote>.
elemList elements	A specification for each version that was promoted. For general notes, see *Encoding of Element Lists* on page 96 below.

The following example shows the data encoded in **<elemList>** and **<elements>**:

```
<elemList>
    <elem
        stream="9"
        version="7"
        elemType="2">/dir00/sub00/file04.txt</elem>
  </elemList>
  <elements>
    <elem
        eid="8"
        ver="9/7">/dir00/sub00/file04.txt</elem>
  </elements>
```

Format of the “server_preop_trig” Trigger Parameters File

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

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

```
    <ip> ... </ip>
    ...
</triggerInput>
```

The set of subelements under the **<triggerInput>** element depends on the user's command—the **issues** parameter is generated only for the ***promote*** command, for example. The following table provides a summary. For full details, see the sample **server_preop_trig** script in the **examples** directory in the AccuRev installation area.

Parameter	Description
hook	Type of trigger: **server_preop_trig**.
command	The user command. See *Server-Side Triggers* on page 77 for a list of commands that can fire this trigger.
principal	AccuRev username of person invoking the command.
ip	The IP address of the client machine.
stream1	The user's workspace stream.
stream2	The workspace's parent (backing) stream. For ***promote*** commands, this is the stream being promoted to.
stream3	The new name of the workspace or stream.
depot	Name of depot targeted by the command.
objectName	Name of object targeted by the command.
objectType	Type of object targeted by the command: **1**=reference tree; **2**=workspace; **3**=stream; **5**=user; **6**=group
user	AccuRev username being modified.
newKind	New user kind (**dispatch** or **cm**).
newName	New AccuRev username.
fromClientPromote	(***promote***) The number of the AccuWork issue record entered by the user, when prompted by thechange-package-level integration.
changePackagePromote	(***promote***) A set of <changePackageID> subelements, specifying the change packages (that is, issue records) specified in the user's command. These forms of the ***promote*** command generate a <changePackagePromote> element: • ***promote –I*** (user specifies issue record number(s) on the command line) • ***promote*** (user prompted to specify issue record number(s) by the change-package-level integration) • ***promote –Fx*** (user specifies a set of issue records with an XML file) The user can also specify issue record(s) through the AccuRev GUI.

Parameter	Description
issues	(***promote***) The **<issues>** element contains **<issue>** subelements for all issues that are being promoted, whether or not the issue was explicitly selected for the promote (as can be the case when promoting by file, when one or more issues can be implicitly selected). Each **<issue>** subelement specifies the following information: • Issue number • Workflow stage the issue will be associated with after it transitions from its current workflow stage • The issue's current workflow stage • Whether or not the issue will be complete in the destination stream upon completion of the **promote** operation
comment	Comment string specified by the user. If the comment spans multiple lines, line-terminators are embedded in the string, but the final line does not have a line-terminator.
elements	(***add***, ***co***, ***keep***) A set of **<elem>** subelements, each specifying one element processed by the user's command. For the ***add*** and ***keep*** commands, **hierType** is an **<elem>** attribute, which indicates the exclusive file locking state specified with the **–E** command-line option: **serial** or **parallel**.
output_file	*(XML-format parameters file only)* The name of the file that the **server_preop_trig** script creates and AccuRev reads. See *Controlling Element Type and Exclusive File Locking with a "server_preop_trig" Script*, below.

Controlling Element Type and Exclusive File Locking with a "server_preop_trig" Script

The trigger parameters file sent to a ***server_preop_trig*** script contains an **<output_file>** element — for example:

```
<triggerInput>
  <hook>server_preop_trig</hook>
  <output_file>cache/0_0.out</output_file>
  ...
```

The script can create a file at this relative pathname (it doesn't exist when the trigger fires), in order to control the element type and/or exclusive file locking state of some or all of the elements processed by the user command.

The XML element named **<elements>** in the trigger parameters file contains the data that the script needs to generate the output file— for example:

```
<elements>
  <elem
        count="0"
        eid="0"
        elemType="text"
        hierType="parallel">/dir03/sub05/able.txt</elem>
```

```
  <elem
        count="1"
        eid="0"
        elemType="text"
        hierType="parallel">/dir03/sub05/baker.bin</elem>
  <elem
        count="2"
        eid="0"
        elemType="text"
        hierType="parallel">/dir03/sub05/carr.doc</elem>
</elements>
```

For each AccuRev element to be processed, **<elements>** contains information about how the new version of the element will be created — unless the script intervenes. This includes both the element type (**elemType** attribute) and the exclusive file locking state (**hierType** attribute).

Note: **<elemList>** contains a subset of the data in **<elements>**, and can be safely ignored.

Suppose the example code above was passed to the ***server_preop_trig*** script by the ***add*** command, which the user invoked to place three files under version control: **able.txt**, **baker.bin**, and **carr.doc**. And suppose that the script decides to specify that:

- Elements **baker.bin** and **carr.doc** are to be placed in the **serial** exclusive file locking state.
- The first version of **baker.bin** is to have the **binary** element type.

In this case, the output file should contain the following code:

```
<elemList>
  <elem count="1"  hierType="serial"  elemType="binary"></elem>
  <elem count="2"  hierType="serial"></elem>
</elemList>
```

Notes:

- The top-level XML element in the output file is **<elemList>**, not **<elements>**.
- Each **<elem>** XML subelement identifies an AccuRev element through the **count** attribute (representing the position on the command-line); no element pathname is required.
- The value of the **hierType** attribute must be either **serial** or **parallel**.
- An **<elem>** is required only for AccuRev elements whose exclusive file locking state is to be changed from the default (or with a ***keep*** command, to be changed from its existing state). Thus, there need not be an **<elem>** for file **able.txt**, which is to be created in the default locking mode, **parallel**.
- The number of **<elem>**s need not match the number of AccuRev elements being processed by the command; if there are "too many", the final **<elem>**s are silently ignored; if there are "too few", the final AccuRev elements get the default processing.
- A ***server_preop_trig*** script can coexist with a ***pre-create-trig*** script, both of them making element-type specifications. The ***pre-create-trig*** script must specify an element type for every new element; this is not a requirement for the ***server_preop_trig*** script. If both scripts specify an element type for the same element, the ***server_preop_trig*** script "wins".

Suppressing Transition Execution on Promote

A workflow can optionally specify a transition to be executed when an issue meeting one or more conditions is promoted into a stream. For example, your workflow might execute a “ready for QA” transition when an issue in the workflow stage Complete is promoted into your integration stream.

You can suppress transition execution:

- Using the ***-q*** option for the ***promote*** command. See the *AccuRev® CLI User’s Guide* for more information.
- Using the ***server_preop_trig*** script, as described here.

The trigger parameters file sent to a ***server_preop_trig*** script contains an **<output_file>** element — for example:

```
<triggerInput>
   <hook>server_preop_trig</hook>
   <command>promote</command>
   <output_file>cache/0_0.out</output_file>
  ...
```

The script can create a file at this relative pathname (it doesn’t exist when the trigger fires), in order to control transition execution on some or all of the elements processed by the user command.

The XML element named **<issues>** in the trigger parameters file contains the data that the script needs to generate the output file— for example:

```
<issues>

   <issue id="11" destination_stage="WIP" current_stage="NEW"
   workflow="Enhancement" complete="true"/>

   <issue id="12" destination_stage="WIP" current_stage="NEW"
   workflow="Enhancement" complete="false"/>

</issues>
```

The **<issue>** subelement contains information about how the issue will be modified when it is promoted—unless the script intervenes.

Suppose the example code above was passed to the ***server_preop_trig*** script by the ***promote*** command, which the user invoked to promote issue 11 and issue 12. And suppose that the script decides to specify that trigger execution for issue 11 should be suppressed. In this case, the output file should contain the following code:

```
<ISSUES>

   <ISSUE id="11" apply_transition="false" />

</ISSUES>
```

The transition for issue 12 is not executed under any circumstances because it is incomplete.

> Note: Using **promote -q** overrides any transition execution behavior specified in the ***server_preop_trig*** script **<output_file>** element.

Sample server_preop_trig rules for Change Packages

The Perl snippets below take the examples provided in the sample `server_preop_trig.pl` file in the AccuRev installation `examples` folder one step further and show how you can prevent non-administrators

from purging files from higher-level streams, and how to enforce promote-by-issue in non-workspace streams. Depending on the policies at your site, such changes could be considered best practices to ensure that all changes can be tracked to a specific AccuWork (or third-party ITS) issue.

```
######################################################## CUSTOMIZE ME
#### Add to (or replace) the example code below to
#### implement validation for the PROMOTE command.
########################################################

   # EXAMPLE VALIDATION:
   # only a user listed as an administrator can promote versions
   # to a stream in the "admin_stream" list

   #if ( defined($admin_stream{$stream2}) and `$::AccuRev ismember $principal "$admingrp"` == 0 )
{
   #  print TIO "Promoting to a stream identified as an 'admin stream' disallowed:\n";
   #  print TIO "server_preop_trig: You are not in the $admingrp group.\n";
   #  close TIO;
   #  exit(1);
   #}

   # EXAMPLE VALIDATION:
   # only a user listed as an administrator can run the Promote
   # command without entering a comment
   if ( $comment eq "" and `$::AccuRev ismember $principal "$admingrp"` == 0 ) {
      print TIO "Empty comments for 'promote' command disallowed:\n";
      print TIO "server_preop_trig: You are not in the $admingrp group.\n";
      close TIO;
      exit(1);
   }
   # end of EXAMPLE VALIDATION

   #This will prevent users from promoting or cross promoting individual files.
   #Only users defined in the $admingrp group will be allowed to promote by file.
   #This will prevent issues from becoming incomplete which can cause coalescing problems.

   #foreach my $changepackage (keys(%{$$xmlinput{'changePackagePromote'}})){
   #my @issues = (@{$$xmlinput{'changePackagePromote'}[0]{'changePackageID'}});
   my @noissue = (@{$$xmlinput{'changePackagePromote'}});
      foreach my $issue (@noissue) {
      #foreach my $issue (@issues) {
      #print "Array Issue num = $issue\n";
         if ($issue == 0 and `$::AccuRev ismember $principal "$admingrp"` == 0 ){
         print TIO "Promotion by file is disallowed.\n";
         print TIO "You need to promote by issue, please select the issue which needs promotion and
                     promote\n";
         print TIO "Only users in the $admingrp group are able to promote by file as this can cause
                     incomplete issues.\n";
         close TIO;
         exit(1);
         }
      }

   # end of EXAMPLE VALIDATION

   # no problems, allow command to proceed
   close TIO;
   exit(0);
}

#### end of validation for PROMOTE command

####
#### Validation for PURGE command
```

```
####

if ($command eq "purge") {
# at this point, the following variables will have meaningful values:
#   $hook Trigger name
#   $command  AccuRev command being run
#   $principalUsername of person invoking command
#   $ip     IP address of AccuRev client machine
#   $stream1  Stream from which versions are being purged
#   $depotDepot name
#   $fromClientPromoteData passed from pre-promote-trig script
#   @elems     Element list

######################################################## CUSTOMIZE ME
#### Add to (or replace) the example code below to
#### implement validation for the PURGE command.
########################################################

   # EXAMPLE VALIDATION:
   # only a user listed as an administrator can promote versions
   # to a stream in the "admin_stream" list

   #if ( defined($admin_stream{$stream1}) and `$::AccuRev ismember $principal "$admingrp"` == 0 )
{
   #  print TIO "Purging from a stream identified as an 'admin stream' disallowed:\n";
   #  print TIO "server_preop_trig: You are not in the $admingrp group.\n";
   #  close TIO;
   #  exit(1);
   #}
   # end of EXAMPLE VALIDATION

   #Prevent users from purging elements so AccuWork issue will not become incomplete in streams.
   unless ( $stream1=~/_$principal/ or `$::AccuRev ismember $principal "$admingrp"` == 1 ) {
      print TIO "You can not perform \"Revert to Basis\" or purge operations in streams.\n";
      print TIO "This will prevent CR's from disappearing in streams due to them becoming
                 incomplete issues.\n";
      print TIO "Only users in the $admingrp group will be authorized to preform this
                 operation.\n";
      close TIO;
      exit(1);
   }

   # no problems, allow command to proceed
   close TIO;
   exit(0);
}

#### end of validation for PURGE command
```

Format of the “server_admin_trig” Trigger Parameters File

The parameters file passed to a ***server_admin_trig*** script is in XML format: The set of subelements under the **<triggerInput>** element depends on the user’s command. The following table provides a summary. For full details, see the sample **server_admin_trig** script in the **examples** directory in the AccuRev installation area.

Parameter	Description
hook	Type of trigger: **server_admin_trig**.

Parameter	Description
command	The user command. See *Server-Side Triggers* on page 77 for a list of commands that can fire this trigger.
principal	AccuRev username of person invoking the command.
ip	The IP address of the client machine.
stream1	The stream targeted by the user command.
stream2	The parent (backing) stream of **stream1**.
stream3	(**chws**, **chstream**) The new name of the workspace or stream.
depot	Name of depot in which the AccuWork issue database resides.
objectName	(**remove**, **rmws**, **reactivate**) Name of object targeted by the command.
objectType	(**remove**, **rmws**, **reactivate**) Type of object targeted by the command: **1**=reference tree; **2**=workspace; **3**=stream; **5**=user; **6**=group; **7**=sessions
streamType	(**mkstream**, **mksnap**) The kind of stream being created by the command: **regular**, **passthru**, or **snapshot**.
streamName	(**setproperty**, **rmproperty**) The name of the stream for which the property is being set or removed.
propertyKind	(**setproperty**) Specifies whether you are setting properties for a stream (**stream**) or principal (**principal**)
propertyName	(**setproperty**, **rmproperty**) The property name. If using a reserved property for a stream, must be either **streamStyle** (to set a stream's color) or **streamCustomIcon** (to decorate a stream with build-related artifacts like icons and tooltips that indicate build success or failure).
propertyValue	(**setproperty**) The value associated with the property you are setting for a stream or principal. If you are specifying either **streamStyle** or **streamCustomIcon** properties, see *Encoding of Reserved Stream Properties* on page 97.
user	(**chuser**, **chpasswd**) AccuRev username being modified.
group	(**ismember**, **addmember**, **rmmember**) AccuRev group name
newKind	(**chuser**) New user kind (**dispatch** or **cm**).
newName	(**chuser**, **chgroup**) New AccuRev username or group name.
fromClientPromote	The number of the AccuWork issue record entered by the user, when prompted by the change-package-level integration.
changePackagePromote	A set of <changePackageID> subelements, specifying the change packages (that is, issue records) specified in the user's command.
comment	Comment string specified by the user. If the comment spans multiple lines, line-terminators are embedded in the string, but the final line does not have a line-terminator.
elemList elements	A set of <elem> subelements, each specifying one element processed by the user's command.

Format of the “server_dispatch_post” Trigger Parameters File

The parameters file passed to a ***server_dispatch_post*** script is in flat-text format. The order of the parameters in the table below is the order in which they appear in the file.

Parameter	Description
hook	Type of trigger: **server_dispatch_post**.
depot	Name of depot targeted by the command.
stream1	The stream targeted by the user command.
fromClientPromote	Two SPACE-separated fields: • The number of the transaction that created the previous version of this issue record. (The number **0** indicates that this is a newly created issue record.) • The issue number
transNum	The number of the transaction that created this new version of the issue record.
transTime	The time at which **transNum** was created.
comment	Comment string specified by the user. If the comment spans multiple lines, line-terminators are embedded in the string, but the final line does not have a line-terminator.
principal	AccuRev username of person invoking the command.
elemList elements	A set of <elem> subelements, each specifying one element processed by the user’s command.

Encoding of Element Lists

In both kinds of trigger parameters files, each element is listed by its path relative to the depot’s top-level directory:

```
/tools/cont.sh
```

The path begins with a slash in order to simplify constructing the element’s full pathname on the client machine: just append the given element pathname to the **topDir** pathname (the top-level directory of the user’s workspace tree).

In the flat-text trigger parameters file, the elements (or elements-to-be) to be processed by the user command are listed, one per line, at the end of the file:

```
/tools/cont.sh
/tools/end.sh
/tools/start.sh
                        <-- end-of-file of trigger parameters file
```

(Unlike the set of comment lines, there is no need to supply an element count; an end-of-file condition signals the end of the element list.)

- For **pre-create-trig** and **pre-promote-trig**, the element pathname appears alone on the line.

- For **pre-keep-trig**, each element is followed by the version-ID of the version about to be created (with the workspace specified by name), followed by the element-type code:

  ```
  /dir07/sub04/file02.txt  rack_dvt_john/3  2
  ```

  As always the element-type coding is: 1=directory, 2=text file, 3=binary file, 4=ptext file.
- For **server-post-promote-trig**, each element is followed by the real version-ID of the promoted version (with the workspace specified by number), followed by the element-type code:

  ```
  /doc/Chapter_03.rtf  9/7  4
  ```

In the XML-format ***server_preop_trig*** trigger parameters file, the element paths are encoded as **<elem>** sub-elements of the **<elemList>** element:

```
<elemList>
    <elem>/tools/cont.sh</elem>
    <elem>/tools/end.sh</elem>
    <elem>/tools/start.sh</elem>
</elemList>
```

Encoding of Reserved Stream Properties

In the XML-format **server_admin_trig** trigger parameters file, the **streamStyle** and **streamCustomIcon** reserved stream properties are encoded as follows:

streamStyle

```
<style>
   <color>
      <background-color>hex_value</background-color>
   </color>
</style>
```

streamCustomIcon

```
<streamicon>
   <image>running | success | warning | failed</image>
   <clickurl>URL to build machine</clickurl>
   <tooltip>tooltip text</tooltip>
</streamicon>
```

Restricting the Ability to Set Stream and Principal Properties

If you want to restrict the ability to set stream and principal properties to only those users defined as AccuRev administrators:

1. Add the following variables to the # set variables section of your **server_admin_trig** trigger if they have not been defined already:

   ```
   $streamName = $$xmlinput{'streamName'}[0];
   $propertyKind = $$xmlinput{'propertyKind'}[0];
   $propertyName = $$xmlinput{'propertyName'}[0];
   $propertyValue = $$xmlinput{'propertyValue'}[0];
   ```

2. Add the following logic to restrict the ability to set principal properties:

```
if (($command eq "setproperty") && ($property_kind eq "principal")) {

    if ( `$::AccuRev ismember $principal "$admingrp"` == 0 ) {
        print TIO "You cannot set a properties for users or groups.\n";
        print TIO "server_admin_trig: You are not in the $admingrp group.\n";
        close TIO;
        exit(1);
    }
    close TIO;
    exit(0);
}
```

3. Add the following logic to restrict the ability to set stream properties:

```
if (($command eq "setproperty") && ($property_kind eq "stream")) {

    if ( defined($admin_stream{$property_stream}) and `$::AccuRev ismember
$principal "$admingrp"` == 0 ) {
        print TIO "You cannot set properties for a stream identified as an 'admin
stream'.\n";

        print TIO "server_admin_trig: You are not in the $admingrp group.\n";
        close TIO;
        exit(1);
    }
    close TIO;
    exit(0);

}
```

4. To restrict the ability to remove principal and stream properties, repeat *Step 2* and *Step 3*, and change occurrences of `setproperty` to `rmproperty`.

5. If you choose to modify the `print TIO` text, make sure to not change the values of variables such as `$admingrp`.

Encoding of Command Comments

In the flat-text trigger parameters file, the user's comment is indicated by a line-count (0 or greater), followed by the lines of the comment, if any:

```
4                                <-- number of comment lines to follow
adding some files
this multi-line
comment has
four lines
```

In the XML-format trigger parameters file, the user's comment is encoded as the contents of the **<comment>** element: a single string. For a multi-line comment, this string has line-terminators embedded:

```
  <comment>adding some files   <-- embedded line-terminator
this multi-line                <-- embedded line-terminator
comment has                    <-- embedded line-terminator
four lines</comment>
```

Note that the final line-terminator is automatically stripped from all comment strings.

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.

Trigger Script Exit Status

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

- A zero exit status indicates success: the AccuRev command 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:

- 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 file system 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, 3, or 4, 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 promoted:

  `<pathname-of-element>`

  <*pathname-of-element*> is not a full file system 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. **Note:** The **server_post_promote** trigger is a separate trigger action than the Change-Package-Level Integration which is specifically called only for workspace-to-stream promotes (see "Change Package-Level Integration" in the "AccuWork chapter of the *AccuRev On-Line Help* for more detail).

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 AccuRev user identity of the AccuRev Server itself. Methods for setting an AccuRev username for the AccuRev Server process are described in *AccuRev User Identity of the Server Process* on page 14.

We recommend against using the ***login*** command to set the AccuRev username in the script itself. You would have to include the password for the AccuRev username in the script; this presents a significant security risk.

'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 script 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.

Disabling Triggers

The procedure for disabling a trigger depends on how the trigger was created, as summarized in the following table:

Table 1. Procedures for disabling triggers

If you created the trigger by:	Disable it with:	Comment:
Using the Schema Editor.	The Schema Editor.	Change Package-level integration.

Using the **mktrig** command.	The **rmtrig** command.	Applies to:
		`pre-create-trig`
		`pre-keep-trig`
		`pre-promote-trig`
		`server-post-promote-trig`

Putting scripts in a "triggers" directory.	Rename the script, or the entire directory.
	
	Depot specific triggers:
	`.../storage/depots/depotname/triggers`
	
	Global repository (site_slice) triggers:
	`.../storage/site_slice/triggers`
		Server-side triggers
		
		Depot specific trigger:
		`server_preop_trig`
		
		Global repository site_slice trigger:
		`server_admin_trig`
		
		NOTE: If you rename the script, you must modify the name of the script, not just the extension. For example, to disable server_admin_trig, it is NOT sufficient to rename the script from server_admin_trig.bat to server_admin_trig.BAK. You must modify the filename itself, such as server_admin_trig_BAK.bat, otherwise AccuRev will return an error that the trigger server_admin_trig cannot be executed.

11.Using Streams to Enforce Process

Gated streams are a special type of stream that, when used with AccuRev triggers, allow you to define the conditions under which changes can be promoted; triggers on gated streams update the StreamBrowser to provide at-a-glance status of those changes. When used in front of build or test streams, gated streams can help limit the occasions when faulty code is promoted, reducing the negative impact on development -- you don't want to make your changes available to others who share or rely on your code if it doesn't compile or breaks the build, for example, and gated streams help ensure this doesn't happen.

Read this chapter to learn:

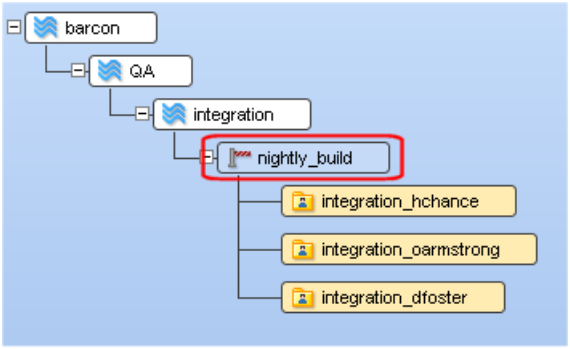
- What a gated stream is, and where and how to use them in your stream hierarchy
- How to create and work with gated streams and their child streams (called staging streams)
- How to use triggers to submit changed files to external tools like compilers, build managers, or test suites
- How triggers update the StreamBrowser based on results returned by those tools

For more information: AccuRev provides other features -- Access Control Lists (ACLs), stream locks, and workflow, to name a few -- that help you manage change and enforce processes in the enterprise. See *Chapter 9 AccuRev Security Overview* in the *AccuRev Administrator's Guide* to learn about ACLs and other tools for restricting access to streams. See Chapter 4, Using Workflow to Enforce SCM Policy, in the *AccuRev Web Interface User's Guide* to learn about issue management in AccuRev.

What is a Gated Stream?

A *gated stream* is a stream that you use to control whether changes promoted into it are automatically promoted out of it, often based on meeting externally established criteria (such as passing a test suite or completing a successful nightly build). Gated streams combine attributes of both dynamic and pass-through streams, and they have features that are unique to them. For example, like dynamic streams, the content of a gated stream changes over time; and like pass-through streams, versions promoted to a gated stream pass through it to the gated stream's parent. Unique to gated streams is a special type of stream called a staging stream. Staging streams are described in the following section.

The following illustration shows a gated stream, *nightly_build*, which has been created off the *integration* dynamic stream:

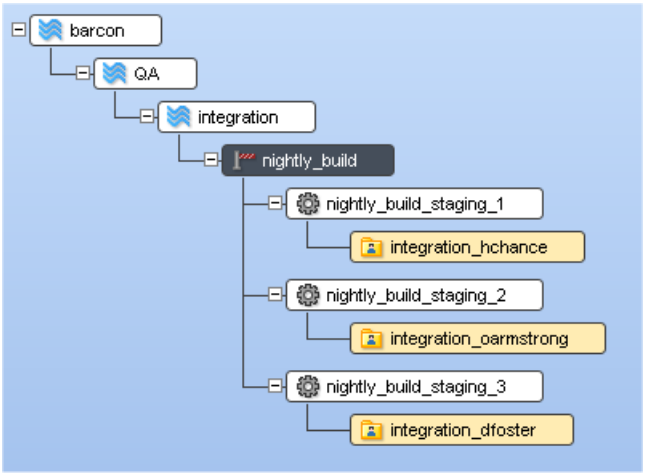


The symbol to the left of the stream name depicts a gate. Also, notice that the body of the stream is transparent, the same as with pass-through streams. This is a visual reminder that gated streams behave in much the same way as pass-through streams.

You create gated streams explicitly, as you would any other stream in AccuRev -- simply choose **Gated stream** in the New Stream dialog box in the AccuRev desktop GUI, or use the **--gated** option for the **mkstream** command in the AccuRev command line interface (CLI). (You cannot create gated streams using the AccuRev Web UI.) See *Creating Gated Streams* on page 112 for more information.

Staging Streams

A *staging stream* is a child of a gated stream that AccuRev creates automatically when you create or reparent a stream or workspace off the gated stream. You cannot create staging streams explicitly (nor can you reparent them). The following illustration shows a staging stream, *nightly_build_staging_1*, which was created off the *nightly_build* gated stream when the workspace *integration_hchance* was reparented to the gated stream.



The symbol to the left of the staging stream name depicts a gear -- this represents the fact that staging streams are "where the work gets done". More on this topic in a moment.

Default staging stream names are based on the name of the gated stream, with a suffix of *_staging_n*, where *n* is a number. Consider the preceding illustration: if another workspace is created or reparented off the *nightly_build* gated stream, the new staging stream associated with that workspace will be *nightly_build_staging_4*. If you choose to rename a staging stream, the general rules about stream names apply (they must be unique across the repository, they cannot start with a digit or a period, and so on).

Staging streams are not displayed in the StreamBrowser by default. They appear only when:

- They have active elements in their default group (as is the case after you promote changes from a workspace, for example), or
- You choose **Show All Staging Streams** from the gated stream's right-click menu. This command causes the StreamBrowser to display all of the gated stream's staging streams, regardless of whether or not they have a default group. (This is shown in the preceding illustration.)

Tip: Do not be concerned if you see staging streams "disappear" from the StreamBrowser. Once changes are promoted out of the staging stream and it no longer has a default group, AccuRev removes the staging stream from the StreamBrowser to avoid visual clutter. You can display them, if desired, using the **Show All Staging Streams** command.

The server_master_trig Trigger

The previous sections described gated and staging streams, how they are related, and how they behave. But it is the **server_master_trig** trigger -- and more precisely, the customizations you define for it -- that determines what AccuRev does with changes that are promoted into a gated stream.

The **server_master_trig** trigger is a repository-wide trigger that runs after certain AccuRev commands are executed. In the case of a gated streams implementation, this trigger runs after changes are promoted to the staging stream. Because it is repository-wide, the **server_master_trig** trigger affects all depots you have created on the AccuRev server.

Triggers are Customized

Triggers are written in the Perl programming language, and while you can write triggers to do whatever you like, the sample **server_master_trig** trigger installed with AccuRev provides a commented template to help you customize it based on your needs. Among other things, you can customize the **server_master_trig** trigger to:

- Submit changes to external tools like compilers and build managers
- Manage messages returned by those tools
- Modify the stream's appearance -- changing its color or displaying an icon -- to represent the status of external jobs (AccuRev provides icons for "running", "success", "failed", and "warning" status, for example)
- Provide links to logs and other resources for troubleshooting problems with external tools (*http://example.com/test_suites*, for example)
- Display tooltips that describe the current status (*Test suite 2 passed*, for example)
- Call other triggers -- like **email_post_promote**, which can be configured to send email to members of your team to alert them of changes requiring their attention

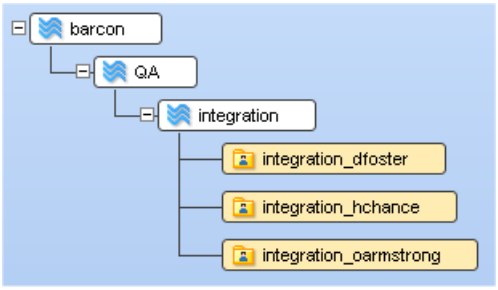
See *Implementing the server_master_trig Trigger* on page 110 for more information.

Example: A Simple Use Case

Now that you have been introduced to all of the components that are part of the AccuRev gated streams feature set -- gated streams, staging streams, and the **server_admin_trig** trigger -- let's walk through a simple use case to see how these components work together to help ensure that changes are promoted and shared with others only when ready.

The barcon Depot

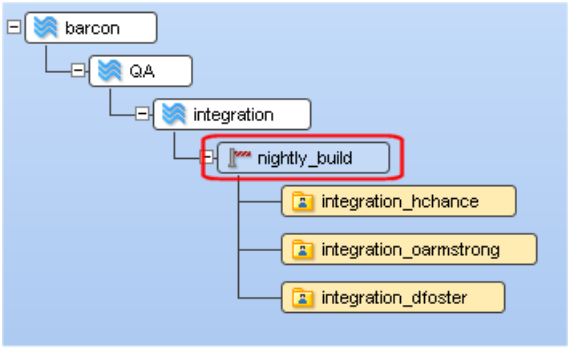
The barcon depot uses a stream hierarchy that might look familiar to many software development organizations. It contains a product release stream, **barcon**; a **QA** stream for testing; and a build stream, **integration**. Individual developers dfoster, oarmstrong, and hchance have workspaces off the integration stream.



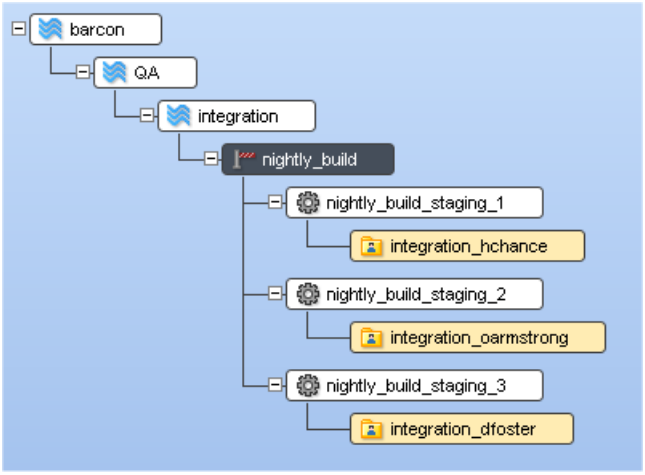
In this stream structure, the developers promote their changes to the **integration** stream. A build is run when changes are promoted, as well as on a nightly schedule; if the build succeeds, the developer promotes the changes to the **QA** stream for testing. Other developers on the team are free to update their workspaces as soon as changes are promoted to **integration**.

Identifying the Need for Gated Streams

Occasionally, changes promoted by a developer break the build. This not only slows down the development process, but it creates additional problems for the other developers on the team who have updated their workspaces with these changes. While there are ways to address such problems after the fact -- using demote, for example -- the administrator decides to create a gated stream between the workspaces and the integration stream, and to use the **server_master_trig** trigger to ensure that code that breaks the build cannot be promoted.



Once the administrator creates the gated stream **nightly_build**, the developers reparent their workspaces to it, as shown in the preceding illustration. Under the covers, AccuRev creates a staging stream for each of the reparented workspaces, like those shown here:



Recall that AccuRev does not display staging streams by default -- they appear only if they include active elements, or, as is the case here, if you explicitly elect to display them using the **Show All Staging Streams** choice on the gated stream's right-click menu.

Setting Up the Trigger

Once the gated stream has been created and the necessary workspaces have been reparented to it, the final step to implementing gated streams is to set up the **server_master_trig** trigger. By default, the sample **server_master_trig** trigger installed with AccuRev:

- Locks the staging stream while the trigger is executing
- Displays a status icon that indicates the status of the changes (or, if needed, problems encountered executing the trigger)
- Displays a tooltip for that icon

As mentioned previously, you can write Perl to make an AccuRev trigger do anything you want. For the **barcon** depot, the **server_master_trig** trigger has been customized to start a build when changes are promoted, and it includes code that updates the status of the staging stream based on the build's results.

Promoting Changes

Now when changes are promoted into the **nightly_build** stream, AccuRev:

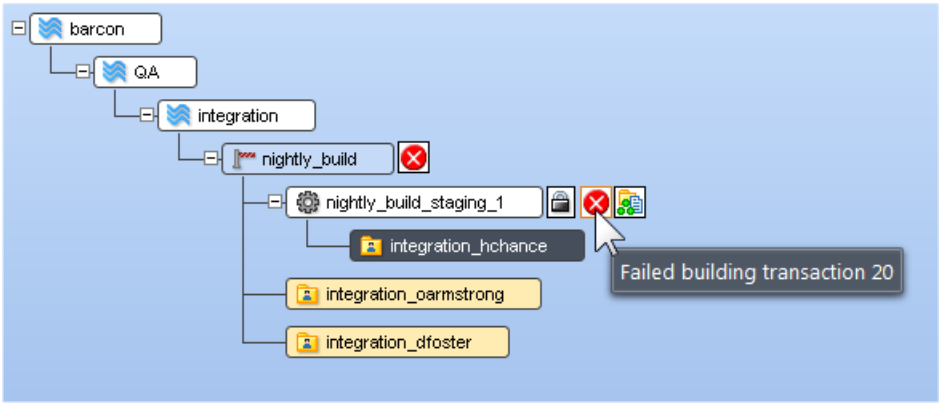
- Displays the staging stream for the changes (**nightly_build_staging_1**, for example)
- Executes the **server_master_trig** trigger
- Displays the status -- of either the trigger execution or the external job, as summarized in the following table

While a trigger is running, you might see one of the following status icons on the staging stream associated with the workspace from which the changes were promoted:

Icon	Description
 Running (animated)	The job associated with the promoted changes is running.
 Success	The job associated with the promoted changes completed successfully.

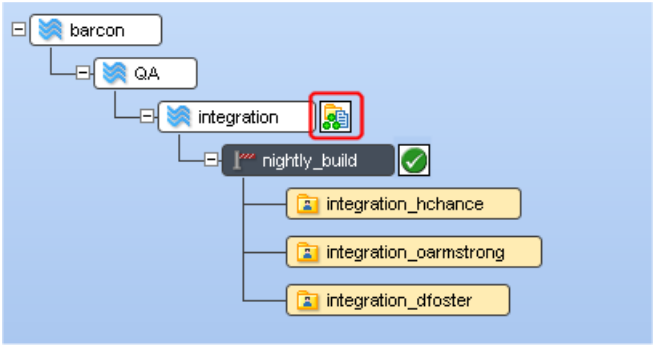
Icon	Description
Warning	The job associated with the promoted changes completed with warnings.
Failed	The job associated with the promoted changes completed with failures.

In this example, the changes promoted to **nightly_build** broke the build, as indicated by both the status icon and tooltip displayed on the **nightly_build_staging_1** staging stream:

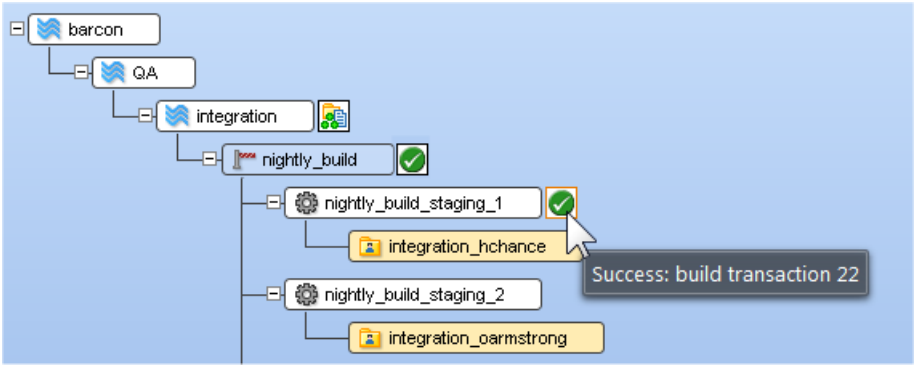


At this point, the user *hchance* needs to fix his code before he promotes it back to the **nightly_build_staging_1** staging stream. Once he does, the trigger re-executes and the build is run again. This time, assuming the fixes are valid and the build succeeds, the changes are promoted from the staging stream, through the gated stream, where they become part of the **integration** stream's default group and will be available to the others when they update their workspaces.

Tip: Note that the gated stream displays a status icon that reflects the status of its staging streams, even when those streams are not displayed. See *Understanding Gated Stream Status* on page 109 for more information.



Once the staging stream is empty -- that is, once all of the active elements in its default group have been successfully promoted -- AccuRev removes it from the StreamBrowser. (AccuRev hides staging streams by default if they don't have any active changes.) You can use the **Show All Staging Streams** choice on the gated stream's shortcut menu to display empty staging streams, as shown in the following illustration:



Understanding Gated Stream Status

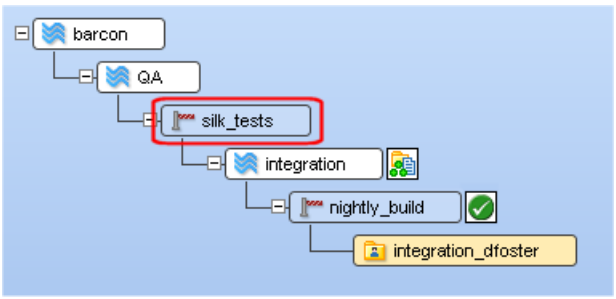
AccuRev displays a status icon on gated streams that represents a summary of the status of all of its staging streams. The same status icons are used for both gated and staging streams, but they have a slightly different meaning when displaying summary status information on gated streams, as described in the following table:

Icon	Description
Running (animated)	The status of at least one of the gated stream's staging streams is *Running*. The *Running* status takes precedence over all others.
Failed	The status of at least one of the gated stream's staging streams is *Failed*, and none is *Running*.
Warning	The status of at least one of the gated stream's staging streams is *Warning*, and none is *Failed* or *Running*.
Success	The status of all of the gated stream's staging streams is *Success*.

The "roll-up" status icon on gated streams lets you quickly check on the status of the jobs associated with a gated stream's staging streams, even if those staging streams are not currently displayed in the StreamBrowser. Note that staging streams that have no active changes in the default group do not contribute to the roll-up status for the gated stream

Adding Gated Streams

You can use multiple gated streams in your stream hierarchy. In the barcon stream hierarchy, for example, the administrator has decided to add a second gated stream, this one in front of the **QA** stream to ensure that any changes promoted out of the **integration** stream pass a suite of tests on the Silk Test platform.



Implementing Gated Streams

The process of implementing gated streams involves the following steps:

1. Implement the **server_master_trig** trigger on your AccuRev server.

   See *Implementing the server_master_trig Trigger* on page 110 for more information.

2. Create one or more gated streams in your stream hierarchy.

   See *Creating Gated Streams* on page 112 for more information.

3. Reparent workspaces and streams to your gated streams as required.

The following sections describe these procedures in greater detail. *Working with Gated and Staging Streams* on page 112 and *Troubleshooting* on page 114 provide information you might find useful once you have your gated streams implementation up and running.

Implementing the server_master_trig Trigger

This section describes how to implement the **server_master_trig** trigger -- what changes to make to it and how to enable it -- to support gated streams. Note that this section describes aspects of trigger implementation that are unique to the **server_master_trig** trigger. See *Chapter 10 AccuRev Triggers* for more information on AccuRev triggers and how to use them.

Step 1: Create a Non-expiring Session

The first thing to do in order to implement the **server_master_trig** trigger is to create a non-expiring session on the AccuRev Server by logging in to the AccuRev Server using `accurev login -n <username> <password>`. This is typically the admin user. Make a note of this user's username as you will need include it in the **server_master_trig** trigger when you edit it.

This is a one-time procedure, and one which you might have already performed if you are already using server-side triggers in your AccuRev installation. See *Trigger Script Execution and User Identities* on page 100 for more information.

Step 2: Edit the server_master_trig Trigger

All sample triggers are installed to the **examples** directory where you installed AccuRev, **c:\Program Files (x86)\AccuRev\examples**, for example. In order to edit the **server_master_trig** trigger:

1. Make a copy of the **server_master_trig** trigger. Do not edit the **server_master_trigger.pl** file in the **examples** directory.
2. Locate the following section in the trigger:

   ```
   # Unix Example
   # $::AccuRev = "/usr/accurev/bin/accurev";
   #
   # Windows Example
   $::AccuRev = "C:\\progra~1\\accurev\\bin\\accurev.exe";
   ```

   You use this section to configure the path of the AccuRev executable. Note that there are separate sections for UNIX and Windows users. The Windows section is uncommented by default.

   a. If necessary, comment out the Windows example and uncomment the UNIX example.

   b. Modify the path shown for `$::AccuRev=` as needed.

3. Next, locate the following section in the trigger:

```
# Unix Example
# $ENV{'HOME'} = "/home/replace_with_username";
#
# Windows Example
$ENV{'HOMEDRIVE'} = "c:";
$ENV{'HOMEPATH'} = "\\Users\\replace_with_username";
```

   You use this section to configure the path to the home directory for the AccuRev user under which this trigger script will be run. Note that there are separate sections for UNIX and Windows users. The Windows section is uncommented by default.

   a. If necessary, comment out the Windows example and uncomment the UNIX example.

   b. Replace `replace_with_username` with the `<username>` you specified using the `login -n` command in the preceding section, *Step 1: Create a Non-expiring Session* on page 110.

4. Finally, locate the following section in the trigger:

```
########################################################## CUSTOMIZE ME
##### START: CUSTOM EXTERNAL ACTION #####
```

   You use this section of the **server_master_trig** trigger script to specify what it is you want the trigger to do. For example, you can use Perl to:

   - Write a simple condition that restricts promotes to a particular stream
   - Send email to an individual who is tasked with reviewing code before it can be promoted
   - Submit the changed files to kick off a build or run a test suite

5. When you have finished modifying your copy of **server_master_trig.pl**, save your changes.

Step 3: Put server_master_trig.pl on the Server

Once you have saved your changes to **server_master_trig.pl**:

1. Create the a **\storage\site_slice\triggers** directory where you installed AccuRev if one does not already exist.
2. Copy the modified **server_master_trig.pl** to **storage\site_slice\triggers**.

Step 4: Windows Only: Create a Batch File for server_master_trig.pl

Once you have saved your changes to **server_master_trig.pl**, you need to create batch file if your AccuRev server is running on Windows. It is the batch file that AccuRev executes when elements are promoted into the gated stream.

1. Navigate to the **\storage\site_slice\triggers** directory where you copied the **server_master_trig** trigger.
2. At that location, run:

   ```
   pl2bat server_master_trig.pl
   ```

   This creates a batch file with the same name as the trigger (**server_master_trig.bat**).

Creating Gated Streams

You can create gated streams anywhere in your depot's stream hierarchy that you choose, but it is common to create them as children of your build, test, and integration streams to ensure that code from individual contributors does not introduce errors that will affect other users. You might still wish to include manual processes such as code reviews as part of your development process, of course, but gated streams allow you to automate aspects of the development lifecycle to ensure a baseline level of quality and operability.

You can create gated streams using the StreamBrowser or the AccuRev CLI as described in the following procedures.

From the StreamBrowser

To create a gated stream from the StreamBrowser:

1. Identify the stream you want to use as the gated stream's parent stream.
2. Right-click the parent stream and choose **New Stream** from the right-click menu.
3. In the New Stream dialog box, enter a name for the gated stream in the **Name** field.
4. In the **Stream Type** field, choose **Gated stream**.
5. Click OK.

Using the CLI

To create a gated stream using the AccuRev CLI:

1. Identify the stream you want to use as the gated stream's parent stream.
2. At the command line, enter:

```
accurev mkstream -s <gated stream name> -b <backing stream name> --gated
```

Working with Gated and Staging Streams

Generally speaking, gated streams adhere to the same rules as other AccuRev streams. For example, you can rename and reparent them as you would other streams, but there are a few considerations of which you should be aware:

- You cannot reparent a staging stream. A staging stream is always the child of the gated stream for which it was created.
- You can delete a staging stream only if it has no children.
- You can delete a gated stream only if its staging streams meet the conditions for deletion.

Showing Empty Staging Streams

AccuRev hides empty staging streams by default. AccuRev considers a staging stream to be empty if it has no default group. If you want, you can choose to display empty staging streams.

To show empty staging streams:

1. Right-click the gated stream whose empty staging streams you want to show.
2. Choose **Show All Staging Streams** from the menu.

To hide them again, right-click the gated stream and choose **Hide Empty Staging Streams**.

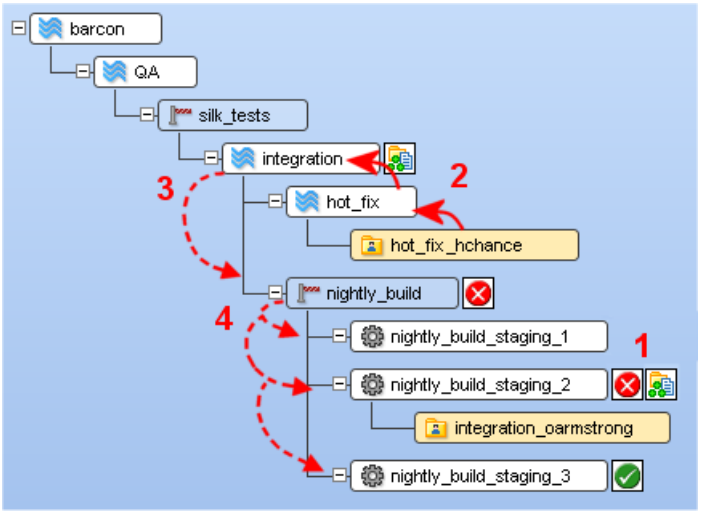
Note: The **Include hidden streams** check box in the StreamBrowser shows (or hides) streams that have been removed from the depot. It has no effect on whether or not staging streams are displayed in the StreamBrowser.

When Jobs Do Not Succeed

The external jobs associated with staging streams will not always succeed -- promoted changes might fail to compile, or they might break a build, for example. When jobs do not succeed, the staging stream's default group retains the active elements until the changes are promoted again, at which point the **server_master_trig** trigger runs again, the external job runs again, and so on.

Any time the staging stream has active elements in its default group -- as is the case if the external job fails, for example -- AccuRev "listens" for changes in the stream hierarchy that might affect the external job. If changes are promoted to a stream *above* the gated stream, AccuRev re-executes the **server_admin_trig** trigger when those changes are inherited by the staging stream.

Consider the following illustration:



Here, (1) user *oarmstrong* has promoted changes to the staging stream **nightly_build_staging_2**. Unfortunately, his changes broke the build, and the trigger returned a status of "failed". Note that his changes remain in the staging stream's default group. Shortly thereafter, before he has the opportunity to promote his fixes, (2) user *hchance* promotes her changes to the **hot_fix** stream. When her changes are promoted again to the **integration** stream, they are automatically inherited (3) by the **nightly_build** gated stream, and again (4) by its three staging streams.

Because AccuRev is listening for changes that occur in the backing streams of staging streams that have active elements in the default group, as soon as *hchance*'s changes are promoted to the **integration** stream, the **nightly_build_staging_2** staging stream will inherit these changes and re-execute the trigger (using these latest changes).

Overriding Running Status

There might be occasions when you want to manually override the Running status on a staging stream. For example, if your software development process requires a code review before changes can be promoted, your reviewer needs a way to set the staging stream status when the review is complete.

You can manually set the staging stream status using the **stagingStreamResult** reserved stream property:

```
accurev setproperty -r -s <stream name> stagingStreamResult "<status>"
```

If the stream name contains spaces, you must enclose it in quotation marks. Valid values for *<status>* are "running", "success", "failed", and "warning".

For example, if you want to set the status of staging stream **acme dev** to success, you enter:

```
accurev setproperty -r -s "acme dev" stagingStreamResult "success"
```

The staging stream status result is reset the next time changes are promoted to the staging stream.

What about streamCustomIcon?

The reserved stream property **streamCustomIcon** can be used to manually set the icon, tooltip, and URL for any stream, including gated and staging streams. However, changes to a staging stream made using **streamCustomIcon** do not affect the roll-up status reflected on the gated stream.

Troubleshooting

This section describes tools and techniques you can use to troubleshoot problems you might encounter with your gated stream implementation.

Trigger Log

The trigger log, **trigger.log**, is located in the **\storage\site_slice\logs** directory where you installed AccuRev. It is always a good place to start when investigating problems with the **server_master_trig** trigger.

Common Trigger Issues

The following table summarizes some of the common trigger issues you might encounter with your gated streams implementation and how to address them. Note that AccuRev uses the same status icons when it encounters these issues as it does during normal operation of the **server_master_trig** trigger.

Issue	Icon	Tooltip
Invalid user. The user identified in the `Script user setup` portion of **server_master_trig.pl** must be the user logged in to AccuRev server or the trigger will not run. To correct this problem, the script user should log in to AccuRev using `accurev login -n`.	 Running (animated)	Starting trigger...
Syntax error. You might have a syntax error in the trigger Perl script. You can check for syntax errors by running: `perl -wc server_master_trig.pl`	 Running (animated)	Starting trigger...
Perl module missing. The trigger will not start if it uses a Perl module that has not been installed. The syntax check performed by running the following command will ensure that you have access to all the Perl modules your trigger script requires: `perl -wc server_master_trig.pl`	 Running (animated)	Starting trigger...

Issue	Icon	Tooltip
The $::AccuRev path is incorrect. Locate the AccuRev path variable in the first CUSTOMIZE ME section of server_master_trig.pl and ensure that you are using the correct entry for your operating system (examples for both UNIX and Windows are provided, but one should be commented out) and that the path is correctly specified.	 Running (animated)	Starting trigger...
The trigger cannot be located. This error occurs if AccuRev cannot locate the **server_master_trig** trigger. To correct this problem, verify that the trigger has been saved to the **\storage\site_slice\triggers** directory where you installed AccuRev, and, if running in Windows, that a **.bat** file for the trigger has been created there.	 Warning	Cannot find trigger
The trigger cannot be executed. This error occurs only on Linux if the **server_master_trig.pl** does not have the executable permission enabled. You can use **chmod** to change the trigger script's access mode.	 Warning	Cannot execute trigger

Tip: Consider customizing the trigger to display a different tooltip for the running icon to distinguish trigger issues from normal operation. For example, you might want to display "Build running..." during builds or "Code review pending..." while waiting for a reviewer to finish her code review.

12.The ‘maintain’ Utility

This document describes AccuRev’s **maintain** utility, an administrative tool for occasional use under the guidance of an AccuRev Support Services representative. Before executing a **maintain** command, you must first stop the AccuRev Server.

The **maintain** commands that require a database administrator password will prompt you for it. For security reasons, you cannot specify a database administrator password on the command line.

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 in the next section. Following that are “how to” sections involving use of **maintain** commands.

Specifying a Database Admin Username and Password

Several **maintain** options require a database administrator username and password (`<db-admin>`). For security reasons, you cannot enter a database administrator password on the command line when invoking a **maintain** command. You can only provide a password when prompted by **maintain**, or through the encrypted DB_PASS entry in `<ac-install>/bin/acserver.cnf`. Here are the rules for providing this information to a **maintain** command:

- If you specify a value for `<db-admin>` when starting the **maintain** command, **maintain** will prompt you to enter the database administrator password.
- If you do not specify a value for `<db-admin>` when starting the **maintain** command, **maintain** will attempt to connect using the DB_USER and the encrypted DB_PASS values specified in `<ac-install>/bin/acserver.cnf`. In the case of standard installations, the command will fail since, for security reasons, the database role specified by DB_USER typically does not (and should not) have database administrative privileges. You will then have to specify a value for `<db-admin>` on the command line, and be prompted for the password.
- If you try to specify a value for the database administrator password when starting the **maintain** command on the command line, **maintain** will display an error message.

‘maintain’ Command Reference

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 ""
```

chslice

```
maintain chslice <slice-number> <new-location>
```

Changes the location of an existing slice in the repository. Use the *show slices* command to get a listing of slice numbers and their current locations.

Note: this command does not physically move the slice. See the *chslice* command in the *AccuRev CLI User's Guide* for more information.

chuser

```
maintain chuser <user-ID> <new-username>
```

Changes the principal-name (AccuRev username) of an existing user. You specify the user by the unique numeric user-ID, which is immutable. This command is similar to the **accurev chuser** command.

concheck

```
maintain concheck [ site | diag ] <db-admin>
```

Tests AccuRev's ability to connect to the database.

When **site** is specified, the command verifies the existence of the AccuRev database as well.

When **diag** is specified, the command performs a series of diagnostics (server initialization, database connection, query execution, and database creation) and displays timing information for each one.

Also see *Specifying a Database Admin Username and Password* on page 115.

dbcheck

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

Performs certain checks on the whole AccuRev database or a specified *depot-name*:

- Checks the connectivity to the AccuRev database.
- Checks the database-schema version information.
- Checks the access to the **streams** table.
- For each active depot, checks access to the depot schema, the database-schema version, and the monotonicity of transaction numbers and element-IDs.

Note: Stop the AccuRev Server before running this command.

dbupgrade

```
maintain dbupgrade [ --auto_commit ] <db-admin>
```

Upgrades the AccuRev database when the schema changes between AccuRev versions (between 5.x and 6.x, for example). This process may take several minutes or several hours, depending upon the size of the database being upgraded. You should always perform a full backup before running **dbupgrade**. See *Backup/Restore of the AccuRev Repository* on page 119 for more information.

The **dbupgrade** wizard will prompt you through the options and values that you need to provide, including database admin name and password. If you are upgrading from AccuRev 4.x, the wizard gives you the option to perform a "dry run" upgrade -- one that does not write to the database -- to allow you to verify the UTF-8 changes that are required for the current AccuRev database before actually committing them. If you already have a UTF-8 database (AccuRev 5.2 and later) you will not be given the option for a dry-run because no UTF-8 conversion is required.

During the upgrade, **dbupgrade** checks and corrects (if necessary) the integrity of information in AccuRev depots. These integrity checks focus primarily on the **trans** (transactions), **trans_entry** (transaction-entries), **ver** (version), **vvr** (virtual-version), and **anc** (ancestry) tables.

The --auto_commit option. If you have a large number of depots (several hundred or more), consider using the **--auto_commit** option. The **--auto_commit** option instructs **dbupgrade** to commit each depot immediately after it is checked, rather than waiting until every depot has been inspected before committing all changes. This requires fewer database resources and can help speed the **dbupgrade** process.

If a failure occurs when using this option, some depots will have been upgraded to the new database version, while others will have not. If you can resolve the problem, you can run **dbupgrade** with the **--auto_commit** option again; **dbupgrade** will skip the depots that have already been upgraded and resume the upgrade process with the remainder. If you are unable to resolve the problem that lead to the failure, you can return to your pre-upgrade state by reinstalling your previous AccuRev release and then restoring from the backup you created prior to running **dbupgrade**.

See "Using the 'maintain dbupgrade' Command" in the current *AccuRev Installation and Release Notes* for the most up-to-date information about using **dbupgrade**. Also see *Specifying a Database Admin Username and Password* on page 115.

mkdbuser

```
maintain mkdbuser <db-admin>
```

Makes a database user using the DB_USER and DB_PASS values from the **acserver.cnf** file in the AccuRev **bin** directory. The AccuRev Server uses this username to communicate with the database.

mksite

```
maintain mksite <db-admin>
```

This command creates an empty database for use with AccuRev 5.0 and higher.

Also see *Specifying a Database Admin Username and Password* on page 115.

restore

```
maintain restore <backup-file-spec> <db-admin>
```

Provides a facility for restoring AccuRev metadata from backup (see *Backup/Restore of the AccuRev Repository* on page 119).

The <*backup-file-spec*> can be either a file name (if the backup was saved to $BACKUP_LOC) or the full path and name of the backup file.

Also see *Specifying a Database Admin Username and Password* on page 115.

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 119.

rmsite

```
maintain rmsite <db-admin>
```

Removes the PostgreSQL database used by AccuRev 5.0 and higher.

Also see *Specifying a Database Admin Username and Password* on page 115.

setcnf

```
maintain setcnf <parameter> [<value>]
```

Writes the <*parameter*> and <*value*> specified to the **acserver.cnf** file in the AccuRev **bin** directory in the form `PARAMETER=value`. If the <*parameter*> is DB_PASS, this command encrypts the plain-text password given as well.

server_properties

```
maintain server_properties [ update <db-admin> ]
```

Displays the following server properties from both the **acserver.cnf** file and the AccuRev database: SITE_SLICE_LOC, MASTER_SERVER, and PORT. In addition, the values from each location are compared and the command displays whether or not they match.

When the **update** option is specified, the values that do not match are written from the **acserver.cnf** file into the database.

*Important! Whenever you edit the values for SITE_SLICE_LOC, MASTER_SERVER, or PORT in the **acserver.cnf** file, you **must** also run the **maintain server_properties update** command before the new values will take effect.*

show slices

```
maintain show slices [ all ]
```

For each AccuRev depot, displays the slice number (**Slice#**) and the full pathname to the directory within the repository that stores the data for that depot (**Location**). Use the **all** parameter to include deactivated (removed) slices in the listing. Use the *chslice* command to change the location of a slice.

su

```
maintain su [ { -a | -r } <accountname> ]
```

Add or remove a superuser account for working with element-level security ("EACL"s). If run with no arguments, **maintain su** displays a list of any superusers. See *Element-Level Security (EACLs)* on page 60 for details.

vercheck

```
maintain vercheck [{ -c | -q } [ -e <eid> ] <depot-name>
```

Checks the storage containers in the specified depot's data directory tree to verify that a storage container file (**.sto**) exists for each file version recorded in the depot database. It also reports CRC

mismatches, which occur when the calculated checksum (CRC) of a **.sto** file does not match the checksum recorded in the corresponding version record.

You can use these vercheck options to correct CRC mismatches:

- **-q**: computes the checksum of the **.sto** file and replaces the CRC value in the version record with the newly computed value.
- **-c**: changes the **.sto** file by removing all its CR characters -- that is, all bytes with the value 0x0D -- and then performs the changes associated with **-q**.

By default, these options correct all versions of all files with CRC mismatches. You can restrict CRC mismatch processing to versions of a particular element with the **-e** option, leaving other versions as-is.

Backup/Restore of the AccuRev Repository

An **accurev** command (**backup**) and a **maintain** command (**restore**) are involved in the scheme for backing up and restoring the AccuRev repository with a minimum of disruption to development activities.

The **accurev backup** command can be executed while the AccuRev Server is running and development activities are ongoing. For more information, see *Backing Up the Repository* on page 6.

At any time after you've executed an **accurev backup** command and verified that it has completed successfully, you can restore the repository to its state at the time the **backup** command was started using **maintain restore**. This is an offline procedure — the AccuRev Server must be stopped when you run it. This procedure is documented in *Restoring the Repository* on page 8.

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 6. Much of a depot's data is stored in its slice of the repository. Use the command **accurev show slices** to determine the pathname of a depot's slice; you'll need it in Step 4 below.

Depot Removal Procedure

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

1. Stop the AccuRev Server:
   - UNIX/Linux: use the **acserverctl** utility, located in the AccuRev **bin** directory:

     ```
     acserverctl stop
     ```

- Windows: use the **Services** control panel, or enter the command **net stop accurev** in a Command Prompt window.

2. Remove the depot using the **maintain** command, which 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 **rmdepot** command removes the depot's records from the repository database (everything except its name and slice information). See Step if you want to rename the deleted depot to reuse the name. It will only appear in an **accurev show –fi depots** command listing (the **–fi** option includes removed items)

3. Restart the AccuRev Server:

   - UNIX/Linux: use the **acserverctl** utility, located in the AccuRev **bin** directory:

     ```
     acserverctl start
     ```

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

4. Remove the depot's directory from the AccuRev **storage** directory tree. Be careful not to remove any other depot's directory! If you're not sure where this information is located, use the commands **accurev show –fi depots** and **accurev show –fi slices** to determine the pathname. (These commands require the AccuRev Server to be running.)

Reusing a Depot's Name

If you want to reuse the depot's name with **accurev mkdepot**, you must first rename the deleted depot with **accurev chdepot -p <*depot-name*> <*new-name*>**.

13. License Management

This chapter describes AccuRev license management -- the types of licenses supported by AccuRev's license management software, Reprise License Manager (RLM), how they work and how to set them up, and how to set up authentication for the RLM server.

AccuRev Product Licenses

AccuRev provides these product licenses:

- Enterprise (AccuRev SCM plus AccuWork)
- AccuWork
- AccuReplica
- GitCentric

Replica licenses allow one of the other product licenses listed above to be used on a replica server. See *Replication Server Licenses* on page 127 for more information.

Standard/Flexible Licenses

RLM supports two license types for each AccuRev product:

- *Standard* -- a "floating" license that is valid for one week. A standard license is appropriate for users who need guaranteed access to AccuRev for an extended period of time.
- *Flexible* -- a "floating" license that is valid for one day (24 hours). A flexible license can be used by multiple users who need less-frequent, non-guaranteed access to AccuRev.

An important feature of RLM is the ability for a company to select 7-day standard licenses or 24-hour flexible licenses depending on the usage pattern of the end users.

For example, assume that your company employs 100 software developers. If your developers use AccuRev every day to get their job done, you probably would want to get standard licenses which ensure that your developers will always be able to have access to AccuRev whenever they need it.

However, if your developers typically work on several other projects which may not be under AccuRev control, or if you have other employees such QA or documentation people who may only require occasional access to AccuRev, you may find flexible licenses are more cost-effective.

Rather than purchasing 100 standard licenses to accommodate every user who might ever require access to AccuRev-controlled files, you might instead purchase fewer flexible licenses that can be used as needed. The optimal number of licenses would, of course, need to be determined by you and your AccuRev representative, based on your specific work patterns and requirements.

Note that you cannot mix standard and flexible licenses on the same server.

How Flexible Licenses Work

Flexible and standard licenses are differentiated by a “minimum check-out time” setting. Note that flexible licenses are NOT returned and made available to other users as soon as a user logs out. Nor are standard licenses tied forever to a particular user name.

Instead, standard licenses have a relatively long minimum check-out period and flexible licenses have a relatively short one. At this time, standard licenses have a seven-day check-out period, while flexible licenses have a 24-hour check-out period.

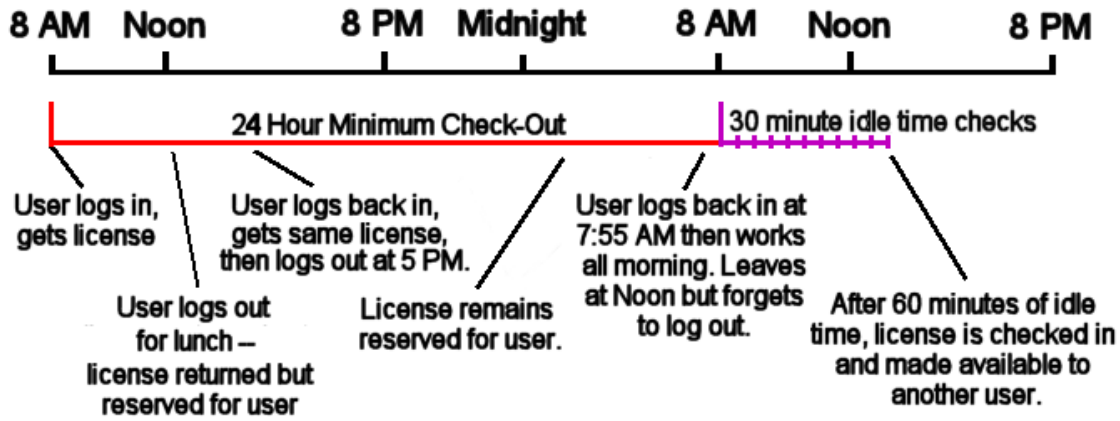
This means that a user who is granted a standard license is guaranteed access to that license for a full week. Even if that user logs out and doesn’t log back in for two days, that license will not be granted to anybody else.

However, a flexible license is guaranteed to its user for a much shorter period — in this example, a single 24-hour day. If that user needs it at 9:00 AM and is finished with it by 11:00 AM, the license will not become available to another user until 9:00 AM the next morning. On the other hand, it will be returned and made available to another user after only a single day— much sooner than the week it takes for a standard license to become available to other users.

Note that the minimum check-out time does *not* mean that the license will suddenly return to the available license pool at the end of that period. If you are still actively using the license when the minimum check-out time is reached, your license will be automatically renewed as long as you do not exceed a minimal idle time setting. The minimal idle time is set to 60 minutes by default. During this period, AccuRev sends a “heartbeat” to the license manager every thirty minutes if the user is still active. (User administrators can extend the 60 minute period with RLM administration functions, though they cannot shorten it.) Once the user has been inactive for 60 minutes, the license is checked in and made available to somebody else.

The key concept to note here is: *A license key, whether standard or flexible, will not time-out and be made available to different user as long as the original user is actively working with AccuRev.*

The figure below illustrates how these time-outs affect an AccuRev license. In this example, the times are representative of a flexible license. A standard license would be similar, except the minimum check-out time would be seven days.



In addition to these license manager settings, licenses are affected by the following actions:

- log outs -- When the user logs out of AccuRev, the license is checked in. However, if the log out occurs within the minimum check out time, the same license will be returned if that user logs in again.

- session time outs -- If the AccuRev session times out, the license is checked in. If and when the user logs in to renew the session, a license is checked out again. This may be a new license, or the original license, depending on the minimum check out time. In any case, no more than one license is checked out to renew the session.
- multiple computers -- If a user logs in from different physical computers or clients during the course of the day, the license manager keeps track of this and will not use more than one license to accommodate these multiple log ins.

Enabling License Management

AccuRev license management is enabled automatically after a successful AccuRev installation. In some instances, however, you might want to adjust the default license settings, and AccuRev and RLM server configurations, so that they are appropriate for your environment. This and the following sections describe how to accomplish these tasks.

License, Configuration, and Options Files

This section describes three files related to AccuRev license management:

- **accurev.lic** (the AccuRev license file)
- **acserver.cnf** (the AccuRev server configuration file)
- **accurev.opt** (the ISV options file)

Generally speaking, you do not need to modify any of these files, but this section describes common modifications -- such as changing the listener port or extending the minimum check-out period -- and how to perform them.

The accurev.lic License file

The AccuRev license file, **accurev.lic**, is a simple text file that you receive from AccuRev when you purchase your licenses. This file is typically located in the **\storage\site_slice directory** where you installed AccuRev. For example: **C:\Program Files (x86)\AccuRev\storage\site_slice\accurev.lic**.

The first line of the **accurev.lic** file specifies the port that the RLM server is listening on. For example:

```
HOST localhost ANY 2375
```

Its default value is 2375. You can change this port if necessary. See *Changing the RLM Listener Port Value* on page 124 for more information.

For additional information about where to place the **accurev.lic** file and any edits that you may need to make to it, see the *AccuRev Installation and Release Notes* and the instructions that you receive with the license file.

The acserver.cnf Configuration File

The **acserver.cnf** file is a configuration file that is generated during installation. It is typically located in the **bin** directory where you installed AccuRev. For example:
C:\Program Files (x86)\AccuRev\bin\acserver.cnf.

The LICENSE_PORT value identifies the port on which the RLM server is listening. For example:

```
LICENSE_PORT = 2375
```

Its default value is 2375, the same as the port value on the HOST line in the **accurev.lic** file.

Note: The **acserver.cnf** LICENSE_PORT for RLM differs from the existing AccuRev PORT value (default value 5050), which clients use to connect to the AccuRev host.

ISV options and accurev.opt

RLM provides the ability to define an "ISV options" file that allows you to adjust certain parameters, such as extending the idle time allowed before a license gets checked back in, or maintaining a list of users who are prevented from obtaining a license. By default, AccuRev installs an ISV options files named **accurev.opt** in the same folder as the license manager executable (**rlm.exe**), typically ***<ac-install>*/rlm** (where ***<ac-install>*** is your AccuRev installation directory). This file has a time-out setting line `"TIMEOUT 120",` to enable automatic idle license check-in.

A common modification of this file is to reserve licenses for specific users so that they stay checked out beyond the usual expiration period.

For example, if a principal developer with the user name "jsmith" is working on an important project and requires special access to a dedicated license, if you are an administrator you can specify a RESERVE line in the **accurev.opt** file such as:

```
RESERVE 1 accurev-ent user jsmith
```

Note: Make sure not to accidentally modify the existing `"TIMEOUT 120"` when you edit **accurev.opt**.

For more information about ISV files, see the RLM documentation referenced in *Additional RLM Documentation* on page 127.

Changing the RLM Listener Port Value

If you need to change the value of the port on which the RLM server is listening, you must change the value in both the HOST line in the **accurev.lic** file and the LICENSE_PORT line in the **acserver.cnf** file as described here:

1. Change the value at the end of the HOST line in the **accurev.lic** file (the value that follows ANY).
2. Change the LICENSE_PORT in the **acserver.cnf** file to the same value.
3. Stop the AccuRev Server. See *Stopping and Starting the AccuRev and RLM Servers* on page 128 if you need help with this step.
4. Stop and restart the RLM Server. See *Stopping and Starting the AccuRev and RLM Servers* on page 128 if you need help with this step.
5. Restart the AccuRev Server. See *Stopping and Starting the AccuRev and RLM Servers* on page 128 if you need help with this step.

Accessing the RLM Web Interface

The RLM Web Interface allows you to perform administrative tasks such monitoring RLM server status, shutting down and starting up the RLM server, and changing the RLM Web Interface password.

To access the RLM Web Interface, point your web browser to port 5054 on the license server machine. For example:

```
http://<license_server_machine>:5054
```

If you are logged into the license server machine, you can use **localhost** instead of the server name.

Tip: The RLM Web Interface works best with Microsoft Internet Explorer. Not all features of the RLM Web Interface, notably, setting the username/password, are supported in other web browsers.

See the *RLM License Administration Manual* for more information on features of the RLM Web Interface. (See *Additional RLM Documentation* on page 127.)

Password Required by Default

When you access the RLM Web Interface, you are prompted for a username/password to authenticate yourself with the RLM server. The default username/password values are admin/admin. AccuRev recommends that you change them.

Note: AccuRev recommends that you change the default username/password authentication credentials and carefully consider granting access to others.

Changing the RLM Password

To change the default password:

1. Log in to the web interface as user admin.
2. Select the **Change Password** menu item, and enter a new password.

   Passwords cannot contain the colon character (:).
3. When you save your changes, the password is encrypted and written to the **rlm.pw** password file located in the **rlm** directory where you installed AccuRev.

Note: Do not enter or change the RLM password directly in the **rlm.pw** password file.

Suppressing Authentication

If you wish, you can suppress authentication by removing (or renaming) the **rlm.pw** password file from the **rlm** directory where you installed AccuRev. If this file is not present in the **rlm** directory when a user attempts to access the RLM Web Interface, no authentication is required.

The **rlm.pw** password file can also be used to grant and restrict access to the RLM server. See *Specifying RLM Privileges* on page 125 for more information.

Note: AccuRev recommends that you retain RLM Web Interface authentication.

Specifying RLM Privileges

In addition to controlling access to RLM using authentication, you can also specify privileges for each user to whom you grant access using the **rlm.pw** file. Privileges are summarized in the following table. See *Additional RLM Documentation* on page 127.

Table 1. Summary of RLM User Privileges

Privilege Name	Description
all	Special privilege name, enables all privileges
edit_meter*	Allows modifying count for meter counters
edit_options*	Allows editing options files for ISV servers
edit_rlm_options*	Allows editing license files and options files for the rlm server
edit_xfer*	Allows editing server-server license transfer settings for ISV servers
logfiles	Enables the functions which change log files - switch, switchr, newlog
remove*	Allows the user to remove a license from a running process

Table 1. Summary of RLM User Privileges

Privilege Name	Description
reread	Allows access to the functions which do reread commands on license servers
shutdown*	Allows access to the functions which shut down license servers
status	Allows display of status and debug log information from the license servers

*Enable “status” privilege even if not explicitly set.

Format of the rlm.pw File

Each line in the **rlm.pw** file corresponds to one user and has the following format:

<username>:<password>:<list-of-privileges>

For example:

```
tom:$5ukMApW1jixwcrGqRALO91:all
harry::edit_options,edit_rlm_options,reread
```

In this example, Tom has a password (it is encrypted when it is created) and his privileges are set to `all`. Harry has no password; his privileges are `edit_options`, `edit_rlm_options`, and `reread`.

How to Specify RLM Privileges

To specify RLM privileges:

1. Open the **rlm.pw** file in the **rlm** directory where you installed AccuRev.
2. Enter privileges for the desired user in a comma-separated list.

   Note: Do not enter or change the RLM password directly in the **rlm.pw** password file.
3. Save and close the file.

Note: A user with no privileges assigned will have access to the following menu items in the RLM Web Interface: "Activate License", "Diagnostics", "RLM Manual...", "System Info", "About", "Change Password", and "Logout".

Configuring Multiple AccuRev Servers

It is easy to configure two or more AccuRev Servers to use the same license manager. You simply need to edit the **acserver.cnf** file on each AccuRev server to point to the same AccuRev RLM server.

This means, for example, that if you have 100 AccuRev licenses and two AccuRev Servers, each AccuRev Server could have 50 simultaneous users, or 90 and 10, or 25 and 75, or any other combination that adds up to 100.

Tip: The ability to configure multiple AccuRev servers using the same license manager gives you the flexibility to set up an AccuRev test server without the need to acquire additional licenses.

Configuration Example

Here are two sample **acserver.cnf** files, showing how to configure two AccuRev servers to share the same AccuRev RLM server on a machine named “zappa”.

AccuRev Server #1 (a Windows XP laptop named “volman”):

```
C:\Program Files\AccuRev\bin>cat acserver.cnf

MASTER_SERVER = volman
PORT = 5050
SITE_SLICE_LOC = C:\Program Files\AccuRev\storage\site_slice
DEPOTS_DEFAULT = C:\Program Files\AccuRev\storage\depots
# log level 2 or 3 is recommended by accurev support team
LOG_LEVEL = 3
REPLICATION_ENABLED = true
LICENSE_MANAGER = reprise
LICENSE_SERVER = zappa
LICENSE_PORT = 2375
```

AccuRev Server #2 (a Linux box named “kaylan”):

```
$ cat  acserver.cnf

MASTER_SERVER = kaylan
PORT = 4800
SITE_SLICE_LOC = /home/dvanvliet/accurev_61/storage/site_slice
DEPOTS_DEFAULT = /home/dvanvliet/accurev_61/storage/depots
# log level 2 or 3 is recommended by accurev support team
LOG_LEVEL = 2
# if server is started by root, uncomment to run as another user.
# need to 'chmod' all files under SITE_SLICE and DEPOTS_DEFAULT
# when changing users.
#USER = nobody
#GROUP = nobody
LICENSE_MANAGER = reprise
LICENSE_SERVER = zappa
LICENSE_PORT = 2375
```

Replication Server Licenses

Installation and administration of replication licenses is automatic -- you do not need to install a license file on the replication server. You just need to calculate how many users you need to support on the master server plus each replica server, plus one additional license for each replica server to be used as a “machine license”. Purchase enough licenses to cover the total number of users who will be accessing AccuRev as well licensing the appropriate number of those users who will be accessing via a Replica server. As long as the license file on the master server covers all of your users, you are all set.

When a user logs into a replica server, the replica server contacts the master server and verifies that the user is permitted to have both a replica license as well as either an Enterprise or an AccuWork license. The master server and replica server negotiate the needed licenses, with no additional attention required by the AccuRev administrator.

The license server ensures that all required licenses are available before checking them out, to ensure that one license does not get used for a replica session that ultimately cannot be successfully started due to unavailability of another license.

Additional RLM Documentation

The *RLM License Administration Manual* is available via the RLM Web Interface and directly from the RLM web site, here: `http://www.reprisesoftware.com/RLM_Enduser.html`.

This document’s intended audience is administrators at end user sites, rather than end users themselves.

Note: As of this writing, the *RLM License Administration Manual* describes the features and functionality supported by the RLM server that is currently installed with AccuRev (RLM version 10.1). As RLM documentation is not installed with the RLM server (rather, it is hosted on the RLM web site), be aware that the available RLM documentation might not always match the version of your RLM server.

Stopping and Starting the AccuRev and RLM Servers

Some license management procedures, like changing the value of the port on which the RLM server is listening, require stopping and starting both AccuRev and RLM servers. This section describes those procedures for Windows and UNIX/Linux users.

Stopping the Servers

The following table summarizes the procedures you can use to stop AccuRev and RLM servers on both Windows and UNIX/Linux.

Platform	Server	Procedure
Windows	Either	Services window > Locate service, right-click, choose **Stop Service**
	AccuRev	`net stop accurev`
	RLM	`net stop "accurev rlm"`
UNIX	AccuRev	`cd <ac-install>/bin` `./acserverctl stop`
	RLM	`rlmdown -c <path to accurev.lic>`

Starting the Servers

The following table summarizes the procedures you can use to start AccuRev and RLM servers on both Windows and UNIX/Linux.

Platform	Server	Procedure
Windows	Either	Services window > Locate service, right-click, choose **Start Service**
	AccuRev	`net start accurev`
	RLM	`net start "accurev rlm"`
UNIX	AccuRev	`cd <ac-install>/bin` `./acserverctl start`
	RLM	`rlm -c <path to accurev.lic> -dlog <path to RLM log file>`

For more detailed information on working with RLM servers, refer to the *RLM License Administration Manual.* (See *Additional RLM Documentation* on page 127.)

14. Using Client-only Install Packages

AccuRev 6.1 and earlier included the Client-only Package Download Utility as part of a full AccuRev installation. This utility allowed AccuRev administrators to download client-only installation packages for supported platforms at any time. (Administrators could also choose to download client-only installation packages during a full AccuRev installation.) Once the client-only packages were downloaded to the server, AccuRev users were able to use the Upgrade Client feature to upgrade their clients using these packages.

Starting with AccuRev 6.2, the Client-only Package Download Utility is no longer available, and administrators no longer have the ability to download client-only packages as part of the full AccuRev installation. AccuRev users will still be able to use the Upgrade Client feature provided the client-only packages are accessible on the AccuRev Server.

This chapter describes the steps to take to allow AccuRev users to continue using the Upgrade Client feature.

Prerequisites

During installation, AccuRev creates platform-specific directories to store client-only installation packages in **\bin\installers** where you installed AccuRev Server. The directory created for the Windows client-only installation package might be **c:\Program Files (x86)\AccuRev\bin\installers\Windows**, for example.

Before continuing, ensure that the directories needed for the client-only installation packages exist in **\bin\installers**. If they don't, create them as needed using the following names:

AIX

HP-UX

Linux

Mac OS X

Solaris Intel

Solaris Sparc

Windows

Downloading the Client-only Installation Packages

To download AccuRev client-only installation packages:

1. Go to the Product Updates section on the Micro Focus SupportLine page:

   http://supportline.microfocus.com/

2. Select a search mechanism, then click **Find Updates**.

3. Check the license agreement, and then expand the **Product** and **Version** tabs to locate the appropriate AccuRev release.
4. Locate the client-only installation packages you require, and click the download button. When prompted, save the **.exe** or **.bin** file to the appropriate location on your AccuRev Server.
5. Rename the file you just downloaded as **AccuRevClientInstall.**[**exe** | **bin**], and copy it to the appropriate location on your AccuRev Server. For example, if you downloaded **accurev-6.2.0-windows-clientonly.exe**, save it to **c:\Program Files (x86)\AccuRev\bin\installers\Windows\AccuRevClientInstall.exe**.

   Important! Client-only install packages must be named AccuRevClientInstall.exe or AccuRevClientInstall.bin or they will not be recognized by the client upgrade feature.

   The client-only installation package is now available for AccuRev users, who can access the package using the:

   - AccuRev GUI: **Help** > **Upgrade Client**
   - CLI: `accurev upgrade_client`

A. Code Page Support

This appendix lists code pages supported by AccuRev. Equivalent values (`ANSI_X3.4-1968` and `CSASCII`, for example) are displayed in the same groups.

- `ANSI_X3.4-1968 ANSI_X3.4-1986 ASCII CP367 IBM367 ISO-IR-6 ISO646-US ISO_646.IRV:1991 US US-ASCII CSASCII`
- `UTF-8`
- `C99`
- `JAVA`
- `CP819 IBM819 ISO-8859-1 ISO-IR-100 ISO8859-1 ISO_8859-1 ISO_8859-1:1987 L1 LATIN1 CSISOLATIN1`
- `ISO-8859-2 ISO-IR-101 ISO8859-2 ISO_8859-2 ISO_8859-2:1987 L2 LATIN2 CSISOLATIN2`
- `ISO-8859-3 ISO-IR-109 ISO8859-3 ISO_8859-3 ISO_8859-3:1988 L3 LATIN3 CSISOLATIN3`
- `ISO-8859-4 ISO-IR-110 ISO8859-4 ISO_8859-4 ISO_8859-4:1988 L4 LATIN4 CSISOLATIN4`
- `CYRILLIC ISO-8859-5 ISO-IR-144 ISO8859-5 ISO_8859-5 ISO_8859-5:1988 CSISOLATINCYRILLIC`
- `ARABIC ASMO-708 ECMA-114 ISO-8859-6 ISO-IR-127 ISO8859-6 ISO_8859-6 ISO_8859-6:1987 CSISOLATINARABIC`
- `ECMA-118 ELOT_928 GREEK GREEK8 ISO-8859-7 ISO-IR-126 ISO8859-7 ISO_8859-7 ISO_8859-7:1987 CSISOLATINGREEK`
- `HEBREW ISO-8859-8 ISO-IR-138 ISO8859-8 ISO_8859-8 ISO_8859-8:1988 CSISOLATINHEBREW`
- `ISO-8859-9 ISO-IR-148 ISO8859-9 ISO_8859-9 ISO_8859-9:1989 L5 LATIN5 CSISOLATIN5`
- `ISO-8859-10 ISO-IR-157 ISO8859-10 ISO_8859-10 ISO_8859-10:1992 L6 LATIN6 CSISOLATIN6`
- `ISO-8859-13 ISO-IR-179 ISO8859-13 ISO_8859-13 L7 LATIN7`
- `ISO-8859-14 ISO-CELTIC ISO-IR-199 ISO8859-14 ISO_8859-14 ISO_8859-14:1998 L8 LATIN8`
- `ISO-8859-15 ISO-IR-203 ISO8859-15 ISO_8859-15 ISO_8859-15:1998`
- `ISO-8859-16 ISO-IR-226 ISO8859-16 ISO_8859-16`
- `KOI8-R CSKOI8R`
- `KOI8-U`
- `KOI8-RU`
- `CP1250 MS-EE WINDOWS-1250`
- `CP1251 MS-CYRL WINDOWS-1251`
- `CP1252 MS-ANSI WINDOWS-1252`
- `CP1253 MS-GREEK WINDOWS-1253`
- `CP1254 MS-TURK WINDOWS-1254`
- `CP1255 MS-HEBR WINDOWS-1255`
- `CP1256 MS-ARAB WINDOWS-1256`
- `CP1257 WINBALTRIM WINDOWS-1257`
- `CP1258 WINDOWS-1258`

- 850 CP850 IBM850 CSPC850MULTILINGUAL
- 862 CP862 IBM862 CSPC862LATINHEBREW
- 866 CP866 IBM866 CSIBM866
- MAC MACINTOSH MACROMAN CSMACINTOSH
- MACCENTRALEUROPE
- MACICELAND
- MACCROATIAN
- MACROMANIA
- MACCYRILLIC
- MACUKRAINE
- MACGREEK
- MACTURKISH
- MACHEBREW
- MACARABIC
- MACTHAI
- HP-ROMAN8 R8 ROMAN8 CSHPROMAN8
- NEXTSTEP
- GEORGIAN-ACADEMY
- GEORGIAN-PS
- KOI8-T
- MULELAO-1
- CP1133 IBM-CP1133
- ISO-IR-166 TIS-620 TIS620 TIS620-0 TIS620.2529-1 TIS620.2533-0 TIS620.2533-1
- CP874 WINDOWS-874
- VISCII VISCII1.1-1 CSVISCII
- TCVN TCVN-5712 TCVN5712-1 TCVN5712-1:1993
- ISO-IR-14 ISO646-JP JIS_C6220-1969-RO JP CSISO14JISC6220RO
- JISX0201-1976 JIS_X0201 X0201 CSHALFWIDTHKATAKANA
- CN GB_1988-80 ISO-IR-57 ISO646-CN CSISO57GB1988
- EUC-JP EUCJP EXTENDED_UNIX_CODE_PACKED_FORMAT_FOR_JAPANESE CSEUCPKDFMTJAPANESE
- EUC-JP-MS EUCJP-MS EUCJP-OPEN EUCJP-WIN
- MS_KANJI SHIFT-JIS SHIFT_JIS SJIS CSSHIFTJIS
- CP932 SJIS-OPEN SJIS-WIN WINDOWS-31J CSWINDOWS31J
- ISO-2022-JP CSISO2022JP
- ISO-2022-JP-1
- ISO-2022-JP-2 CSISO2022JP2
- CN-GB EUC-CN EUCCN GB2312 CSGB2312
- CP936 GBK
- GB18030
- ISO-2022-CN CSISO2022CN

- ISO-2022-CN-EXT
- HZ HZ-GB-2312
- EUC-TW EUCTW CSEUCTW
- BIG-5 BIG-FIVE BIG5 BIGFIVE CN-BIG5 CSBIG5
- CP950
- EUC-KR EUCKR CSEUCKR
- CP949 UHC
- CP1361 JOHAB
- ISO-2022-KR CSISO2022KR
- 437 CP437 IBM437 CSPC8CODEPAGE437
- CP737
- CP775 IBM775 CSPC775BALTIC
- 852 CP852 IBM852 CSPCP852
- CP853
- 855 CP855 IBM855 CSIBM855
- 857 CP857 IBM857 CSIBM857
- CP858
- 860 CP860 IBM860 CSIBM860
- 861 CP-IS CP861 IBM861 CSIBM861
- 863 CP863 IBM863 CSIBM863
- CP864 IBM864 CSIBM864
- 865 CP865 IBM865 CSIBM865
- 869 CP-GR CP869 IBM869 CSIBM869
- CP1125

Index

A

acclient.cnf configuration file 28
    configuring clients for replica servers 47
AccuRev Client
    downloading Client-only installation packages to the AccuRev Server 129
AccuRev Server
    configuring SSL 73
    downloading Client-only installation packages to 129
accurev_server process 13
ACLs, EACLs, ACLS, comparison of different ACL types 60
acserver user 13
acserver.cnf configuration file 5, 16, 28
    enabling replication 45
acserverctl control utility (UNIX) 21
admin_replication 43
archive command (accurev) 38
archiving
    container files and gateway area 37
    reclaiming file space 39
    restoring archived versions 40
    versions archived 38

B

backup command (accurev) 119
backup process, repository 6

C

chref command (accurev) 9
chslice command (accurev) 9
chws command (accurev) 9
client configuration file 28
code pages
    supported code pages 131
contacting technical support x
container files, archiving 37

D

data compression
    enabling data compression on replica servers 50
database, repository
    moving the db directory 9
db directory
    moving 9
depots, repository
    file and metadata storage 11
    moving 9
    removing with maintain utility 119

E

EACL 60
EACLs
    for replica-user 64
    usage scenarios 65
element-level security 60

F

files
    client configuration file 28

G

gated streams
    creating 110
    deleting 112
    described 103
    how status is represented 109
    managing 112
    manually setting staging stream status 113
    overriding staging stream status 113
    reparenting workspaces to 104
    staging stream status and 109
    staging streams and 104
    status 109
    triggers for 105
    troubleshooting problems 113
    using 103
gateway area 37
    reclaiming gateway space 39
group properties
    restricting the ability to set 97

L

logs
    logs for triggers 100
    using trigger.log to troubleshoot gated streams 113

M

maintain utility 115–??
master repository 43
master server
    enabling replication 45
mktrig command 81
mktrig command (accurev) 80
multi-depot environments and triggers 81
Multi-platform environments and triggers 81

P

performance
    improving performance on replica servers 50
pre-create-trig trigger script 83
pre-keep-trig trigger script 84
pre-promote-trig trigger script 86
principal properties
    restricting the ability to set 97
Private Key
    generating 70
properties
    restricting the ability to set stream and principal properties 97

R

reclaim command (accurev) 40
reference trees, moving 9
reparenting
    staging streams 112
    workspaces to gated streams 104
replica repository 43
replica server
    client use 47
    installing and configuring 47–??
    updating 48
replica servers
    improving performance with data compression 50
replica sync command (accurev) 51
replica-user
    EACL permissions 64
repositories, AccuRev
    access controls 5
    backup and restore procedures 7
    ensuring data integrity 5
    master repository 43
    migration procedure 55–??
    replica repository 43
    restoring with maintain utility 119
    synchronizing master and replicas 51

S

Secure Sockets Layer (SSL)
    configuring on the AccuRev server 73
    generating a private key 70
    implementing 70
    obtaining an SSL certificate 71
    trusted certificates and 71
security, element-level 60
server replication procedure ??–47
server_admin_trig trigger script 77
    parameters file for 94
    restricting setproperty and rmproperty with 97
server_dispatch_post trigger script 79, 96
server-post-promote-trig trigger script 87
server_preop_trig trigger script 78
servers, AccuRev
    master server 43
    multi-server installations 27–??
    multithreading limitations 17
    open file handle limits by platform 24
    server configuration file 16
    simple and verbose logging 18
    startup 15
    UNIX acserverctl utility 21
    UNIX groups and admin users 16
    Watchdog monitor process 20
    Windows server control 22
server_start.bat script 15
site_slice
    moving 9
SITE_SLICE_LOC server parameter 16

SSL
see Secure Sockets Layer 70
staging streams
deleting 112
described 104
gated stream status and 109
hiding 112
how status is rolled up for gated streams 109
managing 112
manually setting status 113
overriding status 113
related to gated streams 104
reparenting 112
troubleshooting problems 113
using 104
stream properties
restricting the ability to set 97
StreamBrowser
hiding staging streams in 112
streams
creating gated streams 110
support
contacting technical support x
system clock synchronization 31–??
detecting clock discrepencies 31
eliminating clock discrepencies 32

T

technical support
contacting x
thumbprint
SSL option 74
trigger parameter files 82–99
encoding and contents 96–99
pre-create-trig trigger 83
pre-keep-trig trigger 84
pre-promote-trig trigger 86
server_admin_trig trigger 94
server_dispatch_post trigger 96
server-post-promote trigger 87
server_preop_trig trigger 88
trigger scripts 77–100
firing after Promote 87
firing after server operations 79, 96
firing before Add to Depot 83
firing before Keep 84
firing before Promote 86
firing before server operations 77, 78, 94
prerequisites 79
user identity 100
triggers
and file extensions 81
disabling 101
enabling 80
firing after user commands 78
firing before user commands 77
log for 100
mktrig command 81
multi-depot environments 81
multi-platform environments 81
PATH 81
post-op triggers 78
pre-create-trig
description 77
enabling 80
parameters file format 83
pre-keep-trig
description 77
enabling 80
parameters file format 84
pre-op triggers 77
pre-promote-trig
description 77
enabling 80
parameters file format 86
server_admin_trig
description 77
enabling 80
parameters file format 94
server_dispatch_post
description 79
enabling 80
parameters file format 96
server-post-promote-trig
description 78
enabling 80
parameters file format 87
server_preop_trig
description 78
enabling 80
server_preop_trigtrig

parameters file format 88
triggers for gated streams 105
using trigger.log to troubleshoot gated streams with 113
troubleshooting
problems with gated and staging streams 113
trusted certificates
using with AccuRev 71
TSO. See timestamp optimization
typographical conventions x

U

Update command
updating replica servers 48
user properties
restricting the ability to set 97
users, AccuRev
acserver 13
changing names and passwords 115
UNIX groups and admin users 16

W

Watchdog server monitor 20
Web UI 29
workspaces
moving 9
reparenting to gated streams 104