



The TimeSafe[®] Configuration Management System

AccuRev Technical Notes

Version 3.8

August, 2005

August 16, 2005

AccuRev Technical Notes

August, 2005

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

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

Java is a trademark of Sun Microsystems, Inc.

Rational and **Rational Rose** are trademarks of IBM Corporation.

Table of Contents

Quick Evaluation of AccuRev (Windows)	1
AccuRev's Client-Server Architecture.....	1
Downloading the Installation Program	1
Running the Installation Program	2
Starting the Server as a User Other than the Administrator.....	6
Starting the AccuRev GUI	6
Running the Quick Setup Wizard.....	6
Registering Yourself as an AccuRev User, 7	
Creating and Populating an AccuRev Depot, 7	
Beginning Development Work.....	9
Seeing Other People's Changes.....	10
Adding a New File	10
What's Next?	10
 Quick Evaluation of AccuRev (Unix / Command Line Interface) ..	11
Downloading the Installation Program	11
Running the Installation Program	11
Starting the Server.....	14
Registering Yourself as an AccuRev User	14
Creating and Populating an AccuRev Depot	14
Removing a Depot, 15	
Beginning Work	15
Keeping Files, 15	
Promoting Files, 16	
Seeing Other People's Changes.....	16
Adding a New File	16
What's Next?	16
 Quick Evaluation of AccuRev (Unix Server & Windows Clients) ..	17
Setting Up the Unix Server	17
Proceeding through the Installation, 17	
Starting the Server, 21	
Registering Yourself as an AccuRev User, 21	
Using the Server Host to Create and Populate an AccuRev Depot, 21	
Removing a Depot, 22	
Setting Up the Windows Client.....	22
Downloading the Installation Program, 22	
Running the Installation Program, 23	
Using the Client Host to Create and Populate an AccuRev Depot, 26	
Beginning Development Work.....	32
Seeing Other People's Changes.....	32
Adding a New File	32
What's Next?	32

Converting to AccuRev from Directories Containing Baselevels	33
Creating a Depot	33
Processing the First Baselevel.....	33
Processing Subsequent Baselevels	34
Handling Additional Baselevel-to-Baselevel Differences, 35	
Cleaning Up	35
Creating a Maintenance Stream	37
Using the ACCUREV_IGNORE_ELEMS Environment Variable .	39
Eligible “Whole-Workspace” Commands.....	40
GUI Counterparts to the “Whole-Workspace” Commands, 40	
Commands that Don’t Apply to the Whole Workspace	40
Values for ACCUREV_IGNORE_ELEMS	41
Specifying Directories and Their Contents, 42	
Setting ACCUREV_IGNORE_ELEMS on a Unix System	42
Setting ACCUREV_IGNORE_ELEMS on a Windows System	42
Using Multiple AccuRev Servers	43
Setting Up Multiple Server Machines.....	43
Setting Up Client Machines	43
Configuring a Client Machine to Use Multiple Servers, 43	
Workspaces and Servers, 44	
The ‘wspaces’ File, 45	
Format of the ‘wspaces’ File, 45	
Location of the ‘wspaces’ File, 46	
The –H Command-Line Option, 46	
Techniques for Sharing Workspaces	47
Accessing a Windows-Based Workspace From Multiple Clients.....	47
Accessing a Unix-Based Workspace from Windows Clients	47
What’s the Difference between Populate and Update?	49
In a Nutshell	49
Example 1: Standard Update Scenario, 50	
Example 2: Restoring a Deleted File (“missing” by accident), 50	
Example 3: Adding Data to a Sparse Workspace (“missing” by design), 50	
Example 4: Handling Active Elements, 51	
Example 5: A Tale of Two Files, 51	
Data Structures Used by Populate and Update	51
How the Data Structures Get Their Data	53
Backing Stream, 53	
Workspace Stream, 54	
Workspace Tree, 56	
The Update Algorithm	57
Incomplete Updates, 58	
Incomplete Update: Command Interrupted, 59	
Incomplete Update: Checksum Failure, 59	
Performing the “Fixup” Update, 59	

Using a Trigger to Maintain a Reference Tree	61
Dispatch Command-Line Interface	63
Overview	63
Dispatch CLI Operations	64
Determining the Field-ID / Field-Name Correspondence, 64	
Selecting Issue Records with a Query.....	66
More Complex Queries, 66	
Special Field Types, 68	
Creating a New Issue Record.....	69
Modifying an Existing Issue Record.....	71
Using ‘modifyIssue’ to Create New Issue Records, 71	
Interface to the Change Package Facility.....	72
Adding Entries to a Change Package, 72	
Removing Entries from a Change Package, 73	
Listing the Contents of a Change Package, 74	

Quick Evaluation of AccuRev (Windows)

If you intend to use AccuRev for personal use or are just setting it up for evaluation, these steps will take you through a basic, single platform installation, initial depot setup, and basic AccuRev usage. Any installation parameters that you enter can be changed at a later time.

You do not need to log in as an administrator to evaluate the product. But a non-administrator won't be able to install AccuRev as a service. These instructions include a section on how to start the AccuRev server in console mode to get around this problem.

Should you need any help during the installation or afterwards, remember that your AccuRev download comes with 30 days of free support. Please feel free to contact us: email support@accurev.com or call 1-800-383-8170.

AccuRev's Client-Server Architecture

AccuRev operations use a simple client-server model:

- A user invokes a client program: either the command-line interface (CLI) program **accurev**, or the graphical user interface (GUI) program **acgui**. The computer on which the client program executes is termed the client machine.
- The client program bundles each user command into a transaction request, and sends the request to the server program, which is running on the server machine. The request is sent via TCP/IP, using a particular host/port-number combination. The server program executes the request by accessing the AccuRev repository on the server machine. If the request modifies the repository, the server program records the change as a transaction in one of the repository's storage depots.

The client machine and server machine can be the same computer. (That's the most likely setup for a pre-sales evaluation of AccuRev.) By changing the TCP/IP host/port-number combination, a client program can "switch its attention" back and forth among multiple server programs — and thus, multiple repositories.

Downloading the Installation Program

In a Web browser, go to our web site at <http://www.accurev.com>. On the Downloads page, check the Platform Support Notes to verify that the Java language will run correctly on your computer. (AccuRev Version 3.3.x requires Java Release 1.3.0 or higher.) You may need to install some operating system patches to establish this level of support. Both the installation program and AccuRev itself have a Java user interface.

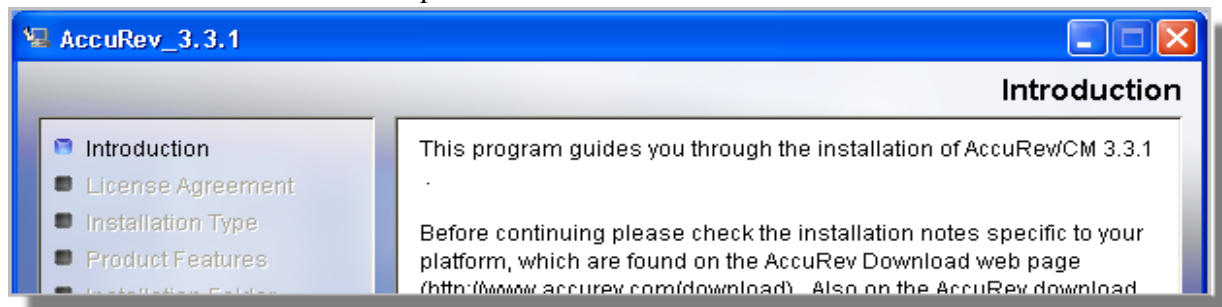
Obtain an AccuRev license key file, by filling out the registration form. You'll need to specify the pathname of this file during the installation process. Then, download the installation program, **AccuRevInstall.exe**.

Running the Installation Program

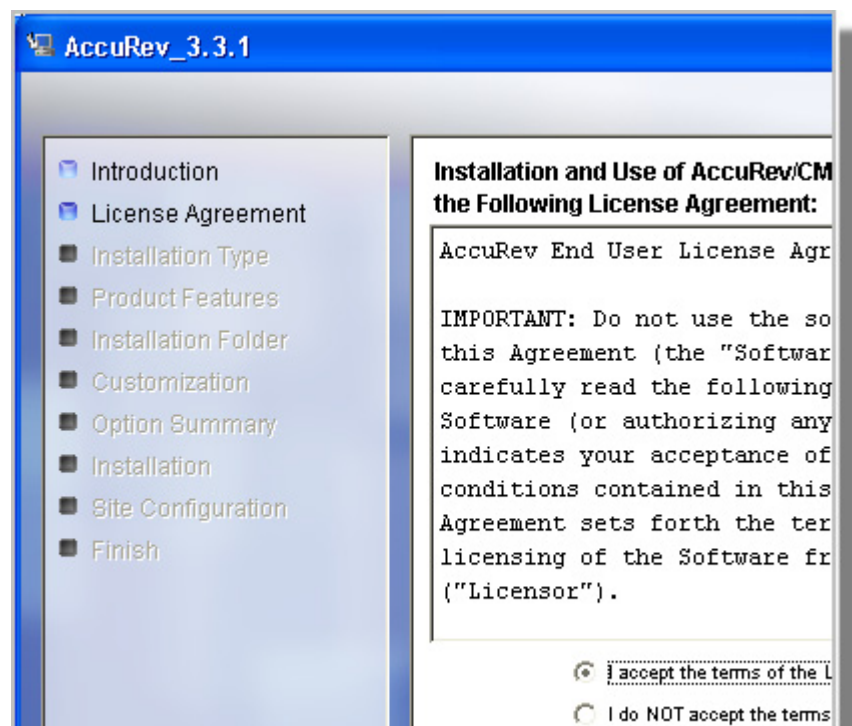
Note: after installation is complete, be sure to continue with the directions in this chapter. We'll guide you through initial depot setup and basic AccuRev usage.

Invoke the **AccuRevInstall** program. It's a "wizard": multiple screens that you navigate by clicking **Next** and **Previous** buttons. The wizard provides lots of helpful information, so you may not even need to consult the following screen-by-screen notes.

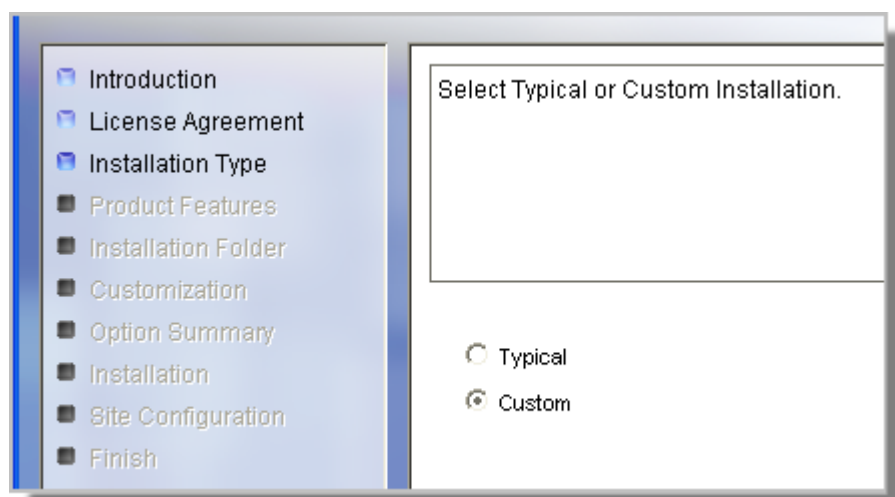
- **Introduction:** The screen explains how to use the wizard. Click **Next**.



- **License Agreement:** Read the agreement, select "I accept", and click **Next**.

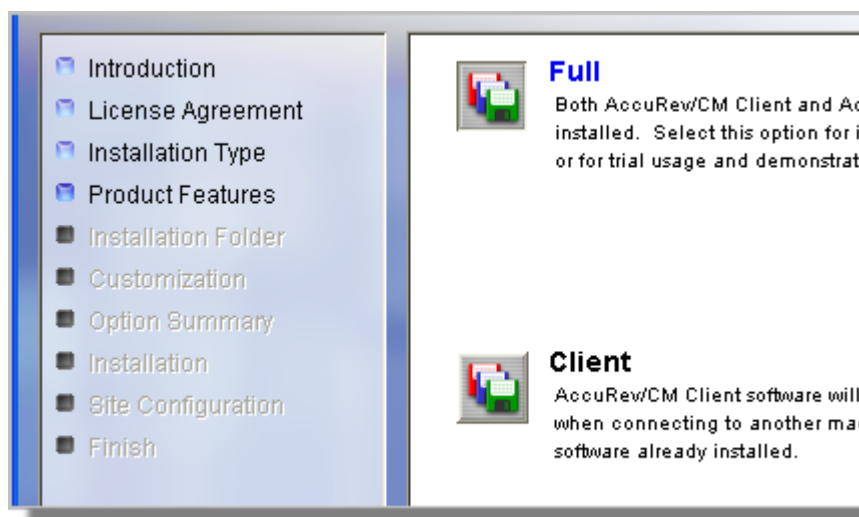


- **Choose Install Type:** If you select **Typical**, the installation program will make a few choices for you, skipping over a few of the subsequent screens.

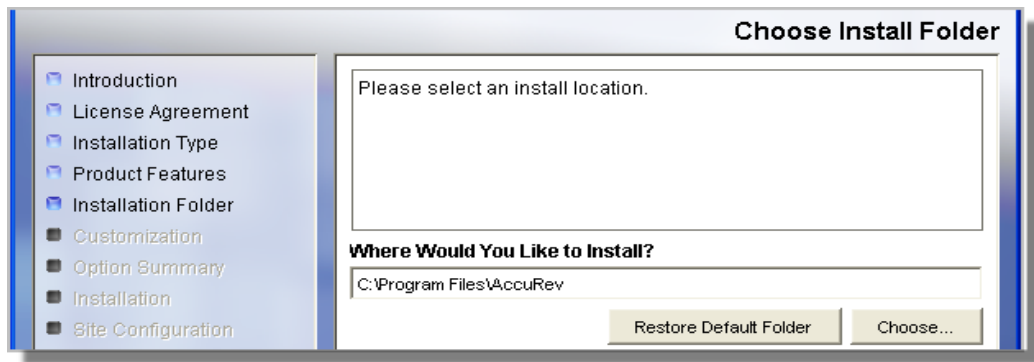


If you are not sure about the **Typical** vs. **Custom** choice, read ahead to the discussion of the various customizations before proceeding.

- **Choose Product Features:** For a product evaluation to be conducted on a single machine, choose **Full**, then click **Next**. If your evaluation involves running the AccuRev Server on a separate machine (e.g. to test network performance), choose **Full** when installing on the server machine; choose **Client** when installing on each client machine.



- **Choose Install Folder:** The wizard suggests a default location for installing AccuRev. This location will store the product itself (about 27MB of executables and configuration files), along with the repository (one or more “depots”) for your version-controlled files.



The installation program uses a heuristic (an intelligent guess) to determine whether you have Windows administrative privileges. If so, it suggests a public location — for example, within **C:\Program Files**. If not, it suggest a location within your home directory.

To choose an alternate location, type a pathname in the textbox or click **Choose** and navigate to an existing location. Before clicking **Next**, make sure the pathname in the textbox specifies a directory that does not yet exist. If you go the **Choose** route, select a location and then append “\AccuRev” (or maybe “\AccuRev3”) to the pathname before proceeding to the next screen.

- **Verify System Time:** This appears for a **Full** installation only. It gives you a chance to change the Windows system clock before installing AccuRev. See *System Clock Synchronization* on page 31 of the *AccuRev Concepts Manual* for a discussion of the importance of an accurate system clock.
- **Optional Customizations:** These customization steps appear if you select a **Custom** install instead of a **Typical** install.
 - **Configuration files:** A set of steps enables you to specify the contents of the client configuration file and, for a Full installation, the server configuration file. If you’re installing on top of an existing AccuRev installation, the wizard first asks if you wish to change the existing settings. The configuration file settings are:
 - **Folder for Server Data Storage: (Full installation only)** Determines the location of the AccuRev repository, which stores your version-controlled files along with the databases that keep track of the files.

The default is to place the repository in a folder named **storage** under the main “install folder”.

You way want to place the repository on a separate disk — perhaps one that is faster and/or backed up more frequently than the disk where you’ve chosen to install the AccuRev executables. The repository’s disk must be located on the machine where the AccuRev server process executes.

- **License Key File: (Full installation only)** Names the license key file that you obtained after filling in the AccuRev registration form. If you’re installing on top of an existing

AccuRev installation, the wizard first asks if you wish to replace your existing license key file.

- **Set Host and Port:** For a product evaluation to be conducted on a single machine, just click **Next** to accept the defaults: the AccuRev Server will run on your machine, using TCP/IP port 5050.

For a **Client** installation, enter the hostname or IP address of the server machine in the **Host** field. You may need to enter a fully-qualified hostname, such as **helios.newmoon.com**. You may also need to specify a different port number (see below).

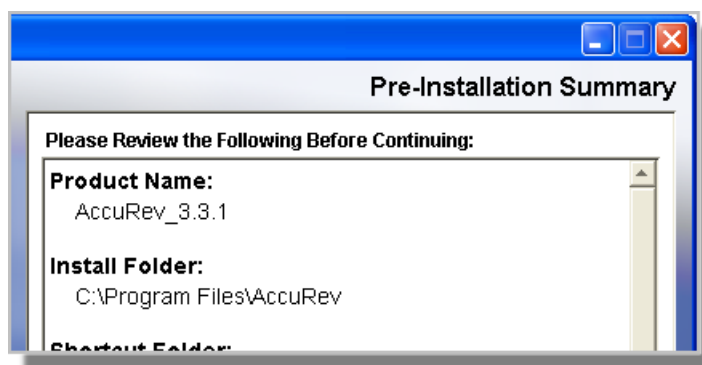
For a **Full** installation, make sure that no network applications running on this machine use port 5050. If this port is already being used, change the port number to one that is not used by any of the other applications. Then, make sure that all other users perform a Custom Client installation of AccuRev, specifying the same alternative port number.

- **Adjust System PATH Variable:** Adjusting the search path facilitates usage of the AccuRev command-line interface: you'll be able to invoke the CLI using the simple program name **accurev**, instead of having to type its full pathname. Adjusting the search path is not necessary if you will be using AccuRev's graphical user interface only, not the command-line interface.
- **Source Control Integration:** Support for IDEs that use the Microsoft SCC interface is always installed. This setting controls whether AccuRev becomes the default provider of SCC services to such IDEs.
- **Java Virtual Machine:** The AccuRev graphical user interface is programmed in the Java™ language. This means that your computer must have a Java virtual machine (Java VM, also known as a "Java runtime environment", or JRE) installed. The wizard will install a Java VM in the AccuRev installation area if you select "Install a Java VM". We recommend this; it won't affect any other Java installation or Java-based applications on your machine.

To enable AccuRev to use a Java virtual machine that is already installed on your machine, the wizard searches for an executable named **java** and displays its pathname if it finds one. You can click **Search for Others** to have the wizard search for additional Java VMs on your computer. Or you can click **Choose Another**, then navigate to another Java VM.

- **Compiled GUI:** Gives you the option of running compiled Java code, rather than the traditional interpreted code. Compiled code rocks!

- **Pre-Installation Summary:**
Click **Install**. The wizard copies files to the installation area on your machine.



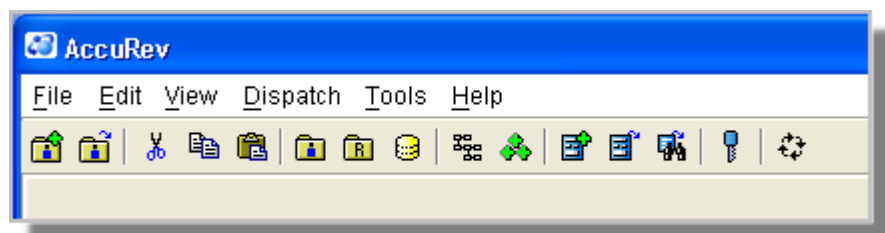
An **AccuRev** shortcut is placed on the Windows desktop as part of the installation process. This shortcut starts the AccuRev graphical client user interface.

Starting the Server as a User Other than the Administrator

If you are running on Windows 95/98/ME or are just unable to install the AccuRev service because you lack administrator privileges, just double-click the **AccuRev Demo Server** shortcut on the desktop. Alternatively, run the **server_start.bat** script in the AccuRev executables (**bin**) directory.

Starting the AccuRev GUI

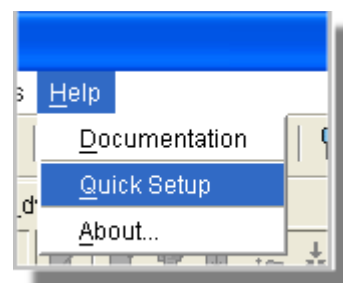
Double-click the **AccuRev** shortcut on your desktop. The main screen appears:



Running the Quick Setup Wizard

The first time you start the GUI, the AccuRev Quick Setup wizard should appear. If it doesn't, run the command **Help > Quick Setup**. This wizard steps you through the process of:

- Registering yourself as an AccuRev user.
- Creating a new storage depot for permanent storage of a set of files.
- Placing an existing set of files under version control, by storing them in the new depot.



Before you begin, make sure that the set of files to be placed under version control are on a disk that is accessible from the machine where you've installed AccuRev.

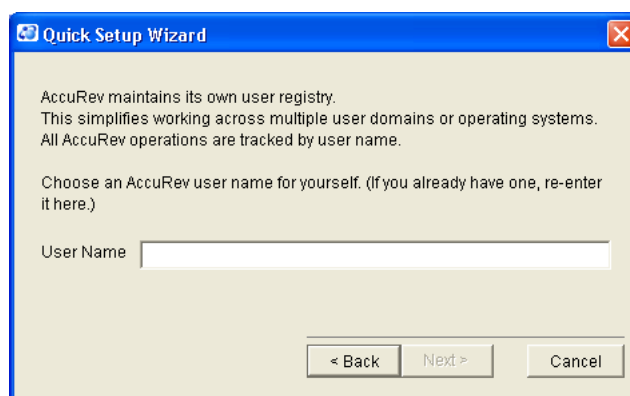
The following sections describe the wizard's operation in detail.

Registering Yourself as an AccuRev User

AccuRev maintains a username registry that is separate from the operating system's username registry. AccuRev usernames are called principal-names. Every AccuRev transaction log entry includes the principal-name of the person who initiated the transaction.

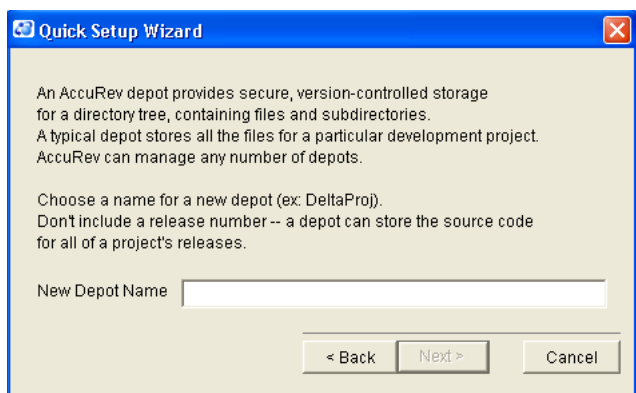
The wizard prompts you to create a principal-name for yourself. If you already have a principal-name, just enter it again in this dialog box.

Note: the currently active principal-name is displayed in the status bar at the bottom of the GUI window. You can change it at any time with the **Tools > Change Active User** command.

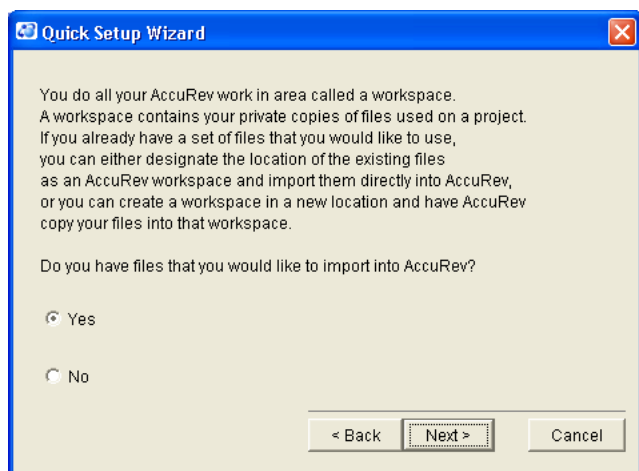


Creating and Populating an AccuRev Depot

Next, the wizard prompts you to specify the name for a new storage depot. Enter a name for the depot (e.g. **brass** for a project code-named "Brass").



Now, you can indicate, by selecting **Yes**, that you have a directory tree whose files are to be placed in the new depot. The alternative (if you select **No**) is to create an empty depot.

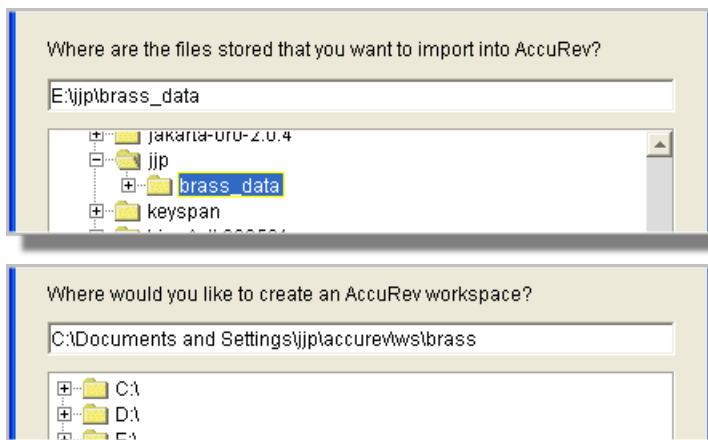


The next steps involve the creation of a new workspace, which enables you to work with the depot's files.

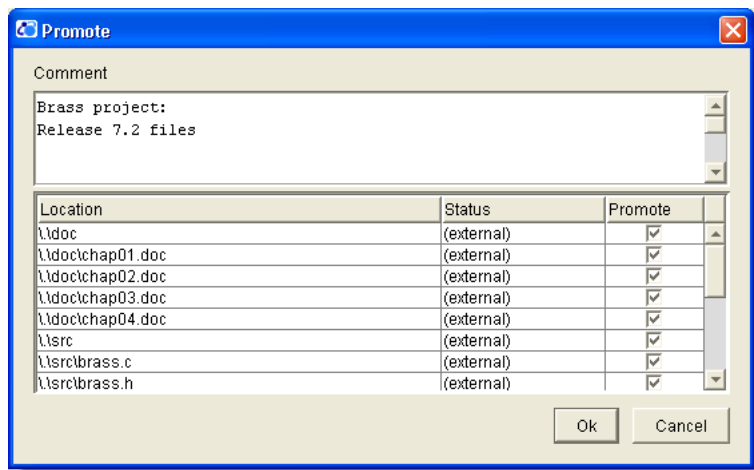
- **If you selected “Yes”, indicating that you have files to be placed in the new depot:**

The wizard asks “Do you want to create a workspace in a new location ...”:

- Select **Yes** to create a workspace in new location, and copy your existing files to that new location. The wizard will first prompt you to specify the location of the existing files. Then, it will prompt you to specify a location for the new workspace.
- Select **No** to convert an existing location into a workspace. for this alternative. The wizard will prompt you to specify the existing location.



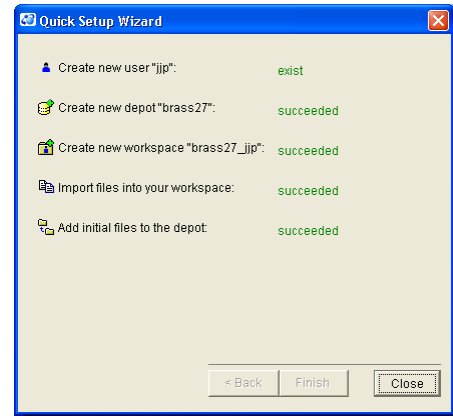
In either case, your existing files are copied to depot in a series of operations that ends with a **Promote** command. click the **OK** button to approve the promotion of the files. This makes the files available to other users' workspaces that you can create later:



- **If you selected “No”, indicating that you want to create an empty depot:**

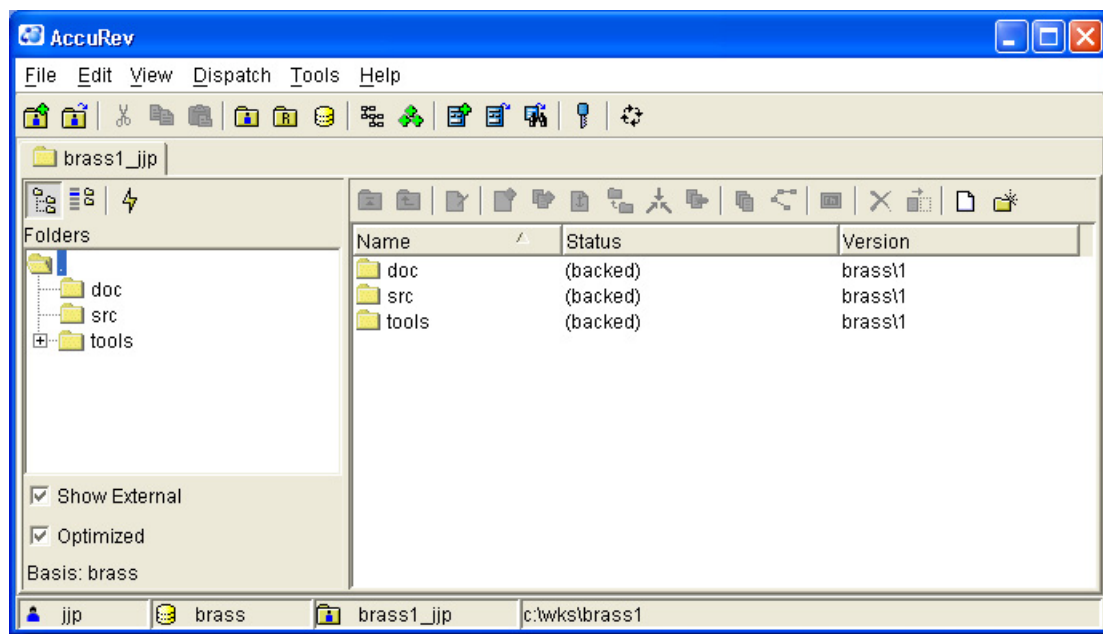
The wizard will prompt you to specify a location for the new workspace.

Finally, the wizard displays a message box, summarizing the work it has accomplished.



Note: To remove a depot from the repository, use the administrative command **maintain rmdepot**. This operation is irreversible! For details, see *Removing a Depot from the AccuRev Repository* on page 54 of the *AccuRev Administrator's Guide*.

After the Quick Setup Wizard ends, AccuRev displays the contents of the newly created workspace. This “Explorer-style” display includes navigation and details panes:



Beginning Development Work


Now that you've created a depot and populated it with files, you're ready to begin making changes to those files. There's no need to “check out” files; simply edit them. To edit a file, double-click it. Alternatively, right-click it and select **Edit** from the context menu.

If you wish to keep some work that you have done, select the elements you want to keep, right-click on the selection and choose **Keep**. You are prompted for a comment, which is optional. This

creates a new version in the depot, in your workspace's private stream. You can keep a file as often as you like.

When your changes are ready for other people to see, first keep the file, then promote it. To promote elements, select them, right-click on them, and select **Promote**. Promote places your most recently kept version of the file in the backing stream, making it available for others to use. The next time they update their workspace, they will get a local copy of your promoted file.

Seeing Other People's Changes

To update your workspace with other people's changes, click the  **Update** button in the navigation pane's toolbar.

Adding a New File

Simply edit a file with your favorite editor. When you're ready for AccuRev to add the initial version of your new file, right-click on it and select **Add to Depot**. Enter a comment (optional, just as with **Keep**).

You can promote the new file immediately. Alternatively, you can continue editing, and then keep and promote.

What's Next?

This chapter has covered only the most basic installation options and features of AccuRev. See the *AccuRev User's Guide* to learn more about AccuRev's capabilities. There are both CLI and GUI editions of this manual.

Quick Evaluation of AccuRev

(Unix / Command Line Interface)

If you intend to use AccuRev for personal use or are just setting it up for evaluation, these steps will take you through a basic, single platform installation, initial depot setup, and basic AccuRev usage. Any installation parameters that you enter can be changed at a later time.

You do not need to log in as **root** to install and use AccuRev.

Should you need any help during the installation or afterwards, remember that your AccuRev download comes with 30 days of free support. Please feel free to contact us: email **support@accurev.com** or call **1-800-383-8170**.

Downloading the Installation Program

In a Web browser, go to our web site at <http://www.accurev.com>. On the Downloads page, verify that the Java language will run correctly on your computer. (AccuRev Version 3.3.x requires Java Release 1.3.0 or higher.) You may need to install some operating system patches to establish this level of support. Both the installation program and AccuRev itself have a Java user interface.

Then, download the installation program, **AccuRevInstall.bin**, for your hardware/software platform.

Running the Installation Program

Note: after installation is complete, be sure to continue with the directions in this chapter. We'll guide you through initial depot setup and basic AccuRev usage.

Start the installation process with one of the following commands:

```
sh ./AccuRevInstall.bin                (graphical wizard)
sh ./AccuRevInstall.bin -i console      (non-graphical wizard)
```

Following is a transcript of a non-graphical installation, with some annotations. For help with a graphical installation, see the corresponding chapter for Windows systems, *Quick Evaluation of AccuRev (Windows)* on page 1.

- **Introduction:** A quick explanation of how to use the wizard. Press **Enter**.

```
InstallAnywhere will guide you through the installation of AccuRev 3.x.
```

```
Respond to each prompt to proceed to the next step in the installation.
```

```
If you want to change something on a previous step, type 'back'.
```

```
You may cancel this installation at any time by typing 'quit'.
```

```
PRESS <ENTER> TO CONTINUE:
```

- **License Agreement:** Read the agreement, type **Y**, and press **Enter**.

Installation and use of AccuRev 3.x requires acceptance of the following License Agreement:

AccuRev End User License Agreement

IMPORTANT: Do not use the software accompanying this Agreement (the "Software")

. . .

DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENT? (Y/N): **Y**

- **Choose Install Directory:** The wizard suggests a default location for installing AccuRev. This location will store the product itself (about 12MB of executables and configuration files), along with the repositories ("depots") that hold version-controlled files. If you are the **root** user, it suggests a public location; otherwise, it suggests a location within your home directory.

Make sure the pathname specifies a directory that does not yet exist.

Where would you like to install?

Default Install Folder: /opt/accurev

ENTER AN ABSOLUTE PATH, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
: **/usr/local/accurev**

INSTALL FOLDER IS: /usr/local/accurev
IS THIS CORRECT? (Y/N): **Y**

- **Choose Product Features:** For a product evaluation to be conducted on a single machine, choose **Full**. If your evaluation involves running the AccuRev Server on a separate machine (e.g. to test network performance), choose **Full** when installing on the server machine; choose **Client** when installing on each client machine.

Please choose the Feature Set to be installed by this installer.

->1- Full
2- Client

ENTER THE NUMBER FOR THE FEATURE SET, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
: **2**

- **Choose Link Location:** Control the creation of symbolic links to the executables.

Where would you like to create links? If you put links in your Path, you will be able to run AccuRev easily from anywhere on your system.

```
->1- Default: /usr/bin
  2- In your home folder
  3- Choose another location...

  4- Don't create links
```

```
ENTER THE NUMBER OF AN OPTION ABOVE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
: 3
ENTER THE ABSOLUTE PATH TO THE SHORTCUT DIRECTORY
: /home/jjp/bin
```

```
SHORTCUT DIRECTORY IS: /home/jjp/bin
IS THIS CORRECT? (Y/N): y
```

- **Choose a Directory for AccuRev Server Data Storage:** This appears for a **Full** installation only. Press **Enter** to create the directory within the installation area.

Choose a Directory for the AccuRev Server Software data repository. This location will store all data managed by AccuRev, so choose a location with enough space. The folder will be created if it does not already exist.

Choose a Directory [/usr/local/accurev/storage]:

- **Adjust System PATH Variable:** Accept the offer to add the AccuRev executables directory to your search path.

Add to Path? (Y/N): y

- **Choose Java Virtual Machine:** The AccuRev graphical user interface is programmed in the Java™ language. This means that your computer must have a Java virtual machine (Java VM, also known as a “Java runtime environment”, or JRE) installed. The wizard will install a Java VM in the AccuRev installation area if you select “1. Install a Java VM”.

To enable AccuRev to use a Java virtual machine that is already installed on your computer, the wizard searches for an executable named **java** and displays its pathname as choice #2 if it finds one. You can also specify a Java VM that has some other name (choice #3).

Please choose a Java VM for use by AccuRev. Installing a VM is recommended. AccuRev requires at least a v1.3 VM.

```
->1- Install a Java VM specifically for this application

  2- /workspaces/u2/java/jdk1.3.1_01/bin/java

  3- Choose a Java VM already installed on this system
```

```
ENTER THE NUMBER FOR THE JAVA VM, OR PRESS <ENTER> TO ACCEPT THE
CURRENT SELECTION: 2
```

- **Pre-Installation Summary:** After checking the summary, press **Enter**. The wizard copies files to the installation area on your machine.

Please Review the Following Before Continuing:

Product Name:
AccuRev_3.x

. . .

PRESS <ENTER> TO CONTINUE:

- **Configure Installed Software — Set Server Host and Port:** For a product evaluation to be conducted on a single machine, click **Next**. If the client and server software will run on different machines, enter the hostname or IP address of the server machine in the **Host** field. You may need to enter a fully-qualified hostname, such as **helios.newmoon.com**.

To be truly careful, make sure that no network applications in your environment use port 5050. You can change the port number to any number that is not used by other applications. If you do, make sure to use the same port number for each AccuRev client and server installation.

- **Configure Installed Software — Start AccuRev Server:** On Win/NT and Win/2000 machines, the AccuRev server can be configured as a service, so that the server starts automatically along with the operating system. Select **Yes** to make this happen.

If you select **No**, or if your machine has a version of Windows that does not support services, you must start the server manually.

Starting the Server

Start the AccuRev Server in the background:

```
accurev_server &
```

This sends a short message to **stdout**. All subsequent messages go to the server's log file.

Registering Yourself as an AccuRev User

Use **mkuser** to create a new AccuRev user. In most cases, your AccuRev username (also known as a principal-name) should match your Unix username.

```
accurev mkuser john_smith
```

Creating and Populating an AccuRev Depot

Follow these steps to place an existing source tree under AccuRev's control:

1. Create a depot with **mkdepot**.

```
accurev mkdepot -p gizmo
```

AccuRev will store the depot in a default location. (It can easily be moved at a later time.)

2. A workspace is a personal area used for editing and testing your work. Make a workspace out of your source tree with the **mkws** command. For example, if your sources are in **/home/jsmith/ws/my_gizmo**:

```
accurev mkws -w my_gizmo -b gizmo -l /home/jsmith/ws/my_gizmo
```

The argument to the **-b** option should be the same as the depot name.

3. **cd** to the top directory of your source tree containing the files you wish to put into the newly created depot:

```
cd /home/jsmith/ws/my_gizmo
```

4. To see all of the files in the workspace which need to be added, use the **stat -x** command.

```
accurev stat -x
```

5. Place all of the files in the workspace under AccuRev version control using the **add -x** command.

```
accurev add -x
```

(You can filter out files with certain extensions (e.g. **.bak**) if you don't want to place *all* the files under version control. See the **add** reference page for details.)

6. Make the newly added files available to other workspaces by promoting them:

```
accurev promote -k
```

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 54 of the *AccuRev Administrator's Guide*.

Beginning Work

Now that you've created a depot and have populated it with the files in your workspace, you're ready to begin making changes to the files you've added. There's no need to check out files; simply edit them.

Keeping Files

If you wish to keep some work that you have done on a file, use:

```
accurev keep -c "comment required" <filename>
```

This creates a new version in the repository in your private workspace stream. You can **keep** a file as often as you like.

Promoting Files

When your changes are ready for other people to see, first keep the file, then promote it:

```
accurev promote <filename>
```

The **promote** command places the most recently kept version of the file in the backing stream for others to use. The next time they update their workspace they will get a local copy of your promoted file.

Seeing Other People's Changes

To update your workspace with other people's changes, use:

```
accurev update
```

Adding a New File

Create a file in the workspace with an editor or by copying it from another location. When you're ready for AccuRev to add the initial version of your new file:

```
accurev add <filename>
```

Add is just like a keep, but it does not require a comment. The file must exist before you add it. You can **promote** now, or continue editing and then **keep** and **promote**.

What's Next?

This chapter has covered only the most basic installation options and features of AccuRev. See the *AccuRev User's Guide* to learn more about AccuRev's capabilities. There are both CLI and GUI editions of this manual.

Quick Evaluation of AccuRev

(Unix Server & Windows Clients)

Whether you intend to use AccuRev for personal use or are just setting it up for evaluation, this chapter takes you through a basic Unix-server/Windows-client installation, initial data setup and basic AccuRev usage. Any installation parameters that you enter can be changed at a later time.

Should you need any help during the installation or afterwards, remember that your AccuRev download comes with 30 days of free support. Please feel free to contact us: email support@accurev.com or call 1-800-383-8170.

Setting Up the Unix Server

1. In a Web browser, go to our web site at <http://www.accurev.com>. On the Downloads page, verify that the Java language will run correctly on your computer. (AccuRev Version 3.3.x requires Java Release 1.3.0 or higher.) You may need to install some operating system patches to establish this level of support. Both the installation program and AccuRev itself have a Java user interface.
2. Obtain an AccuRev license key file, by filling out the registration form. You'll need to specify the pathname of this file during the installation process. Then, download the installation program, for Unix systems, **AccuRevInstall.bin**.
3. Go to the directory where you downloaded the installation program.
4. Start the installation process with one of the following commands:

```
sh ./AccuRevInstall.bin                (graphical wizard)
sh ./AccuRevInstall.bin -i console     (non-graphical wizard)
```

Proceeding through the Installation

This section contains a transcript of a non-graphical installation, with some annotations. For help with a graphical installation, see [Quick Evaluation of AccuRev \(Windows\)](#) on page 1.

- **Introduction:** A quick explanation of how to use the wizard. Press **Enter**.

```
InstallAnywhere will guide you through the installation of AccuRev 3.x.
```

```
Respond to each prompt to proceed to the next step in the installation.
```

```
If you want to change something on a previous step, type 'back'.
```

```
You may cancel this installation at any time by typing 'quit'.
```

```
PRESS <ENTER> TO CONTINUE:
```

- **License Agreement:** Read the agreement, type **Y**, and press **Enter**.

Installation and use of AccuRev 3.x requires acceptance of the following License Agreement:

AccuRev End User License Agreement

IMPORTANT: Do not use the software accompanying this Agreement (the "Software")

. . .

DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENT? (Y/N): **Y**

- **Warning: Non-Administrator Installation:** If you are not logged in as **root**, AccuRev will be installed inside your home directory, or in some other location where you have “write” permission. In addition, you’ll have to start the AccuRev server process manually, not with a Unix “rc” (boot-time) script. In this case the following warning appears:

You currently do not seem to have root privileges. If you wish to install AccuRev for all users on the system: quit the installer now, log out, log in as root, and start the installer again. If you continue, you will be able to install AccuRev within your user account.

You appear to be logged in as 'XXX' currently.

PRESS <ENTER> TO CONTINUE:

- **Choose Install Directory:** The wizard suggests a default location for installing AccuRev. This location will store the product itself (about 12MB), along with the repositories (“depots”) that hold version-controlled files. Make sure the pathname specifies a directory that does not yet exist.

Where would you like to install?

Default Install Folder: /opt/accurev

ENTER AN ABSOLUTE PATH, OR PRESS <ENTER> TO ACCEPT THE DEFAULT

: **/usr/local/accurev**

INSTALL FOLDER IS: /usr/local/accurev

IS THIS CORRECT? (Y/N): **Y**

- **Choose Product Features:** For this Windows-client/Unix-server evaluation, choose **Full** (choice #1) for the Unix machine.

Please choose the Feature Set to be installed by this installer.

->1- Full

2- Client

ENTER THE NUMBER FOR THE FEATURE SET, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
: 1

- **Choose Link Location:** Control the creation of symbolic links to the executables.

Where would you like to create links? If you put links in your Path, you will be able to run AccuRev easily from anywhere on your system.

->1- Default: /usr/bin
2- In your home folder
3- Choose another location...

4- Don't create links

ENTER THE NUMBER OF AN OPTION ABOVE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
: 3

ENTER THE ABSOLUTE PATH TO THE SHORTCUT DIRECTORY
: /home/jjp/bin

SHORTCUT DIRECTORY IS: /home/jjp/bin
IS THIS CORRECT? (Y/N): y

- **Verify System Time:** This gives you a chance to change the Unix system clock (in another window) before installing AccuRev. See *System Clock Synchronization* on page 31 of the *AccuRev Concepts Manual* for a discussion of the importance of an accurate system clock.

The system time is:

Tuesday, November 20, 2001 9:08:25 PM EST

If the system time is not correct, please correct it now before continuing with the installation. AccuRev relies on a correct system time for proper operation. For production AccuRev Servers, it is strongly recommended that a time synchronizing daemon is in use on the system.

PRESS <ENTER> TO CONTINUE:

- **Select License Key File:** Specify the name of the license key file that you obtained after filling in the AccuRev registration form.

Select the AccuRev license key file to install. A license key file is required in order to activate AccuRev Server. If you do not have a license key file, you can obtain one from the AccuRev web site (<http://www.accurev.com/download>).

Choose a license key file [/home/jjp/keys.txt]:

- **Choose a Directory for AccuRev Server Data Storage:** Press **Enter** to create the directory within the installation area.

Choose a Directory for the AccuRev Server Software data repository. This location will store all data managed by AccuRev, so choose a location with enough space. The folder will be created if it does not already exist.

Choose a Directory [/usr/local/accurev/storage]:

- **Adjust System PATH Variable:** Accept the offer to add the AccuRev executables directory to your search path.

Add to Path? (Y/N): **Y**

- **Choose Java Virtual Machine:** The AccuRev graphical user interface is programmed in the Java™ language. This means that your computer must have a Java virtual machine (Java VM, also known as a “Java runtime environment”, or JRE) installed. The wizard will install a Java VM in the AccuRev installation area if you select “1. Install a Java VM”.

To enable AccuRev to use a Java virtual machine that is already installed on your computer, the wizard searches for an executable named **java** and displays its pathname as choice #2 if it finds one. You can also specify a Java VM that has some other name (choice #3).

Please choose a Java VM for use by AccuRev. Installing a VM is recommended. AccuRev requires at least a v1.3 VM.

->1- Install a Java VM specifically for this application

2- /workspaces/u2/java/jdk1.3.1_01/bin/java

3- Choose a Java VM already installed on this system

ENTER THE NUMBER FOR THE JAVA VM, OR PRESS <ENTER> TO ACCEPT THE
CURRENT SELECTION: **2**

- **Pre-Installation Summary:** After checking the summary, press **Enter**. The wizard copies files to the installation area on your machine.

Please Review the Following Before Continuing:

Product Name:
AccuRev_3.x

. . .

PRESS <ENTER> TO CONTINUE:

- **Configure Installed Software — Set Server Host and Port:** For a product evaluation to be conducted on a single machine, click **Next**. If the client and server software will run on

different machines, enter the hostname or IP address of the server machine in the **Host** field. You may need to enter a fully-qualified hostname, such as **helios.newmoon.com**.

To be truly careful, make sure that no network applications in your environment use port 5050. You can change the port number to any number that is not used by other applications. If you do, make sure to use the same port number for each AccuRev client and server installation.

- **Configure Installed Software — Start AccuRev Server:** On Win/NT and Win/2000 machines, the AccuRev Server can be configured as a service, so that the server starts automatically along with the operating system. Select **Yes** to make this happen.

If you select **No**, or if your machine has a version of Windows that does not support services, you must start the server manually.

Starting the Server

To start up AccuRev, run the server:

```
accurev_server &
```

Although this will print a short message to **stdout**, all subsequent output goes to the server's log file.

Registering Yourself as an AccuRev User

Use **mkuser** to create a new AccuRev user, with the principal name you selected above:

```
accurev mkuser john_smith
```

Using the Server Host to Create and Populate an AccuRev Depot

Follow the steps in this section to place an existing source tree, accessible on the server machine, under AccuRev's control.

Note: if the data is accessible only (or just more conveniently) on the client host, skip this section and proceed to *Setting Up the Windows Client* on page 22.

Create a depot with **mkdepot**.

```
accurev mkdepot -p gizmo
```

AccuRev will store the depot in a default location. (It can easily be moved at a later time.)

Now, **cd** to the top directory of your source tree containing the files you wish to put into the newly created depot.

A workspace is a personal area used for editing and testing your work. Make a workspace out of your source tree with the **mkws** command. Assuming you are at the top of your source tree, use

“.” as the argument to the **-l** option of **mkws**. If you've placed your sources in **/home/jsmith/ws/my_gizmo**:

```
cd /home/jsmith/ws/my_gizmo
accurev mkws -w my_gizmo -b gizmo -l .
```

To see all of the files in the workspace which need to be added, use the **stat -x** command.

```
accurev stat -x
```

Place all of the files in the workspace under AccuRev version control using the **add -x** command.

```
accurev add -x
```

(You can filter out files with certain extensions (e.g. **.bak**) if you don't want to place *all* the files under version control. See the **add** reference page for details.)

Make the newly added files available to other workspaces by promoting them:

```
accurev promote -k
```

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 54 of the *AccuRev Administrator's Guide*.

Setting Up the Windows Client

This section describes the procedure for setting up a Windows AccuRev client to work with a Unix AccuRev server. AccuRev is a client/server tool that uses TCP/IP to communicate over a network. Network configuration problems will directly impact AccuRev's performance. So before installing, make sure your client host is properly configured:

- **TCP/IP Check.** Make sure TCP/IP is enabled on your NT server.
- **Time Synchronization.** The system clocks on the AccuRev server hosts and all its client hosts must be synchronized to within a 5-second tolerance. If a client host drifts outside this tolerance, the AccuRev client program will refuse to work. See *System Clock Synchronization* on page 31 of the *AccuRev Concepts Manual*.

Downloading the Installation Program

In a Web browser, go to our web site at <http://www.accurev.com>. On the Downloads page, verify that the Java language will run correctly on your computer. (AccuRev Version 3.3.x requires Java Release 1.3.0 or higher.) You may need to install some operating system patches to establish this level of support. Both the installation program and AccuRev itself have a Java user interface.

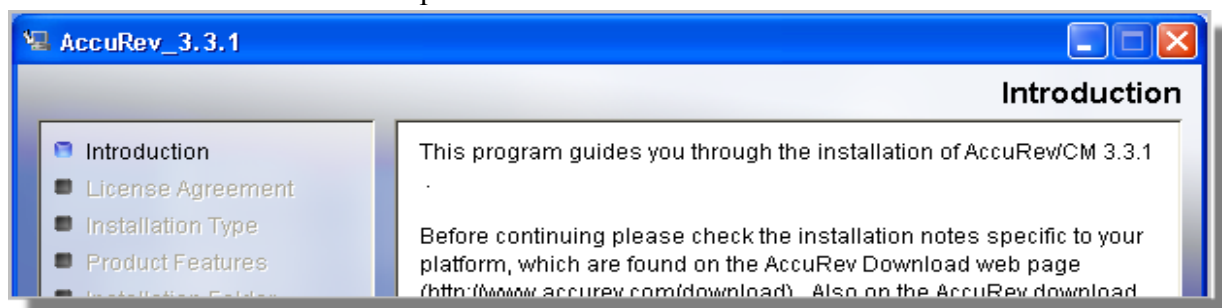
Then, download the installation program, **AccuRevInstall.exe**.

Running the Installation Program

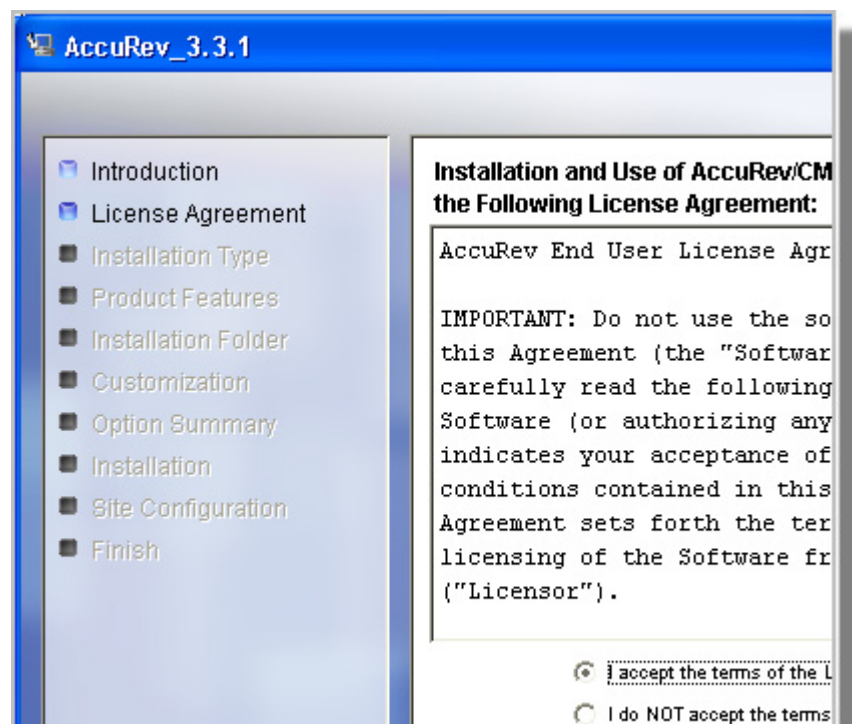
Note: after installation is complete, be sure to continue with the directions in this chapter. We'll guide you through initial depot setup and basic AccuRev usage.

Invoke the **AccuRevInstall** program. It's a "wizard": multiple screens that you navigate by clicking **Next** and **Previous** buttons. The wizard provides lots of helpful information, so you may not even need to consult the following screen-by-screen notes.

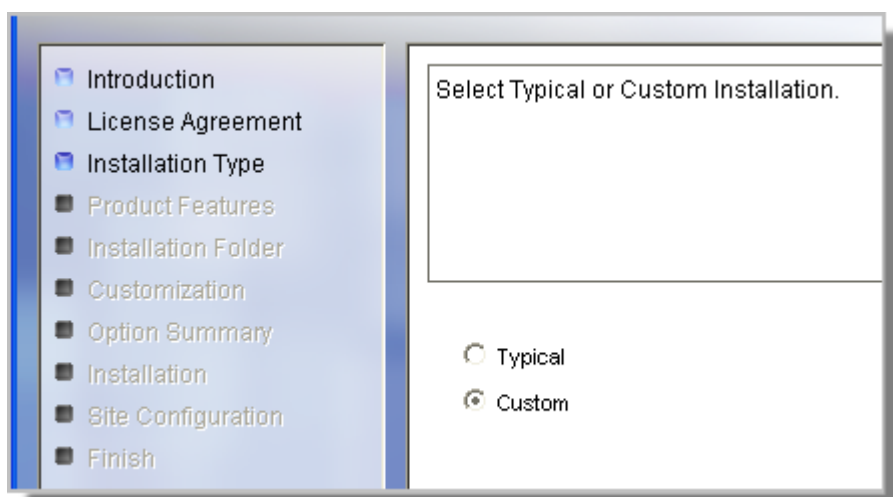
- **Introduction:** The screen explains how to use the wizard. Click **Next**.



- **License Agreement:** Read the agreement, select "I accept", and click **Next**.

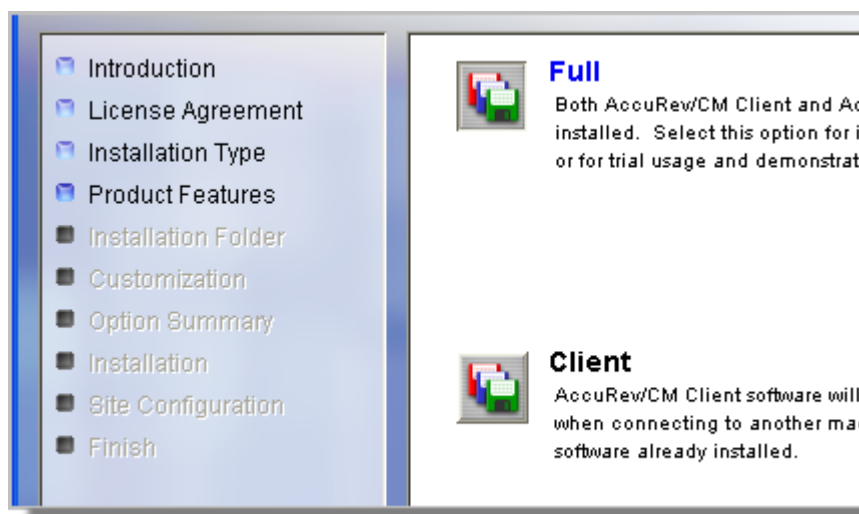


- **Choose Install Type:** If you select **Typical**, the installation program will make a few choices for you, skipping over a few of the subsequent screens.

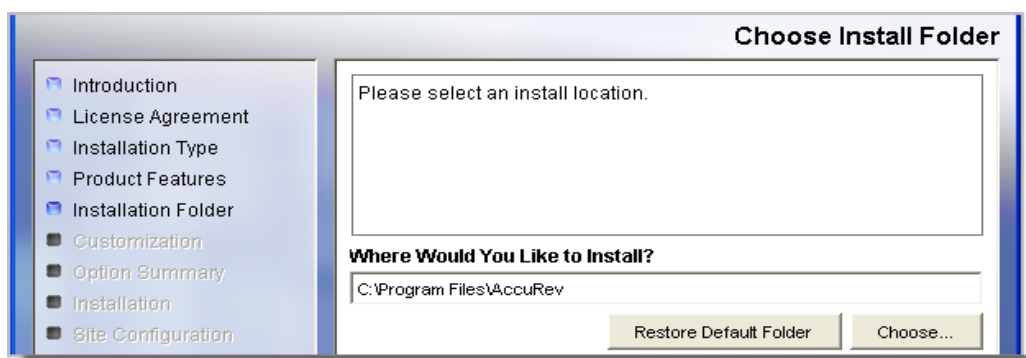


If you are not sure about the **Typical** vs. **Custom** choice, read ahead to the discussion of the various customizations before proceeding.

- **Choose Product Features:** Choose **Client** when installing on each client machine.



- **Choose Install Folder:** The wizard suggests a default location for installing AccuRev. This location will store the product itself (about 27MB of executables and configuration files), along with the repository (one or more “depots”) for your version-controlled files.



The installation program uses a heuristic (an intelligent guess) to determine whether you have Windows administrative privileges. If so, it suggests a public location — for example, within **C:\Program Files**. If not, it suggest a location within your home directory.

To choose an alternate location, type a pathname in the textbox or click **Choose** and navigate to an existing location. Before clicking **Next**, make sure the pathname in the textbox specifies a directory that does not yet exist. If you go the **Choose** route, select a location and then append “\AccuRev” (or maybe “\AccuRev3”) to the pathname before proceeding to the next screen.

- **Optional Customizations:** These customization steps appear if you select a **Custom** install instead of a **Typical** install.
 - **Configuration files:** A set of steps enables you to specify the contents of the client configuration file. If you’re installing on top of an existing AccuRev installation, the wizard first asks if you wish to change the existing settings. The configuration file settings are:
 - **Set Host and Port:** For a product evaluation to be conducted on a single machine, just click **Next** to accept the defaults: the AccuRev Server will run on your machine, using TCP/IP port 5050.

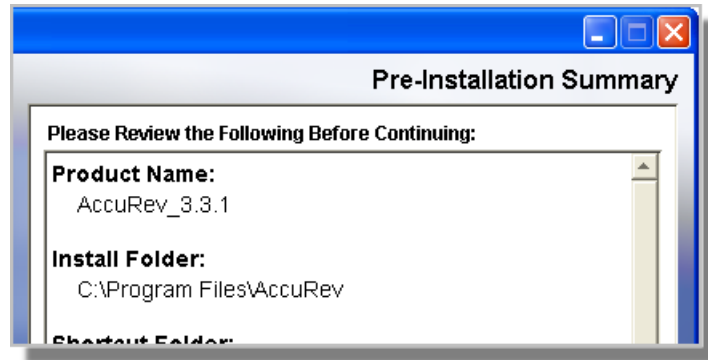
For a **Client** installation, enter the hostname or IP address of the server machine in the **Host** field. You may need to enter a fully-qualified hostname, such as **helios.newmoon.com**. You may also need to specify a different port number (see below).

For a **Full** installation, make sure that no network applications running on this machine use port 5050. If this port is already being used, change the port number to one that is not used by any of the other applications. Then, make sure that all other users perform a Custom Client installation of AccuRev, specifying the same alternative port number.

- **Adjust System PATH Variable:** Adjusting the search path facilitates usage of the AccuRev command-line interface: you’ll be able to invoke the CLI using the simple program name **accurev**, instead of having to type its full pathname. Adjusting the search path is not necessary if you will be using AccuRev’s graphical user interface only, not the command-line interface.
- **Source Control Integration:** Support for IDEs that use the Microsoft SCC interface is always installed. This setting controls whether AccuRev becomes the default provider of SCC services to such IDEs.
- **Java Virtual Machine:** The AccuRev graphical user interface is programmed in the Java™ language. This means that your computer must have a Java virtual machine (Java VM, also known as a “Java runtime environment”, or JRE) installed. The wizard will install a Java VM in the AccuRev installation area if you select “Install a Java VM”. We recommend this; it won’t affect any other Java installation or Java-based applications on your machine.

To enable AccuRev to use a Java virtual machine that is already installed on your machine, the wizard searches for an executable named **java** and displays its pathname if it finds one. You can click **Search for Others** to have the wizard search for additional Java VMs on your computer. Or you can click **Choose Another**, then navigate to another Java VM.

- **Compiled GUI:** Gives you the option of running compiled Java code, rather than the traditional interpreted code. Compiled code rocks!
- **Pre-Installation Summary:**
Click **Install**. The wizard copies files to the installation area on your machine.



An **AccuRev** shortcut is placed on the Windows desktop as part of the installation process. This shortcut starts the AccuRev graphical client user interface.

Using the Client Host to Create and Populate an AccuRev Depot

Note: if you used the Unix server to place an existing source tree under AccuRev's control (as described in *Using the Server Host to Create and Populate an AccuRev Depot* on page 21), you can skip ahead to *Beginning Development Work* on page 32. If you want to use the Quick Setup wizard (new in AccuRev Version 3.1) to set up a depot, use the procedure on page 6 instead of the one in this chapter.

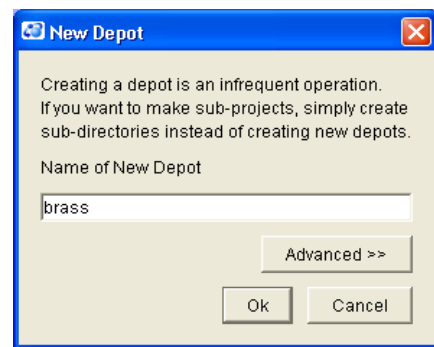
Follow these steps to place an existing source tree on your Windows host under AccuRev's control:

1. Create a depot:

File > New > Depot

Enter a name for the depot (e.g. **brass** for a project code-named “Brass”) and click **OK**. AccuRev will store the depot in a default location. It can easily be moved at a later time.

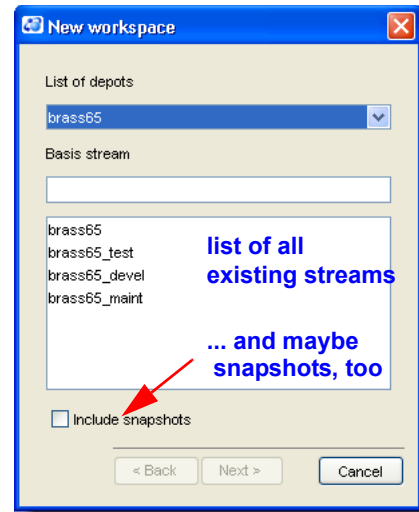
2. Accept the depot-creation command's offer to segue to the creation of a new workspace, a personal area used for editing and testing your work. (You can make a new workspace at any time with the **File > New > Workspace** command.)



Workspace creation is handled by a “wizard”: multiple screens that you navigate by clicking **Next** and **Previous** buttons. The following screen shots show the steps:

- **Backing stream:** The newly created depot has a single stream, with the same name as the depot. The wizard automatically selects this name, so you can just click **Next**.

In general, you can type the name of an existing stream, or click on the name in the list of all existing streams. You can also choose a snapshot (a “static stream”, which never changes) to back a workspace.



- **Name of Workspace to Create:** Click **Next** to accept the wizard’s offer to name the workspace after the backing stream. The suffix *<principal-name>* is appended automatically to the workspace name.

This makes it easy to set up a backing stream and a set of like-named workspaces, one for each user:

backing stream:

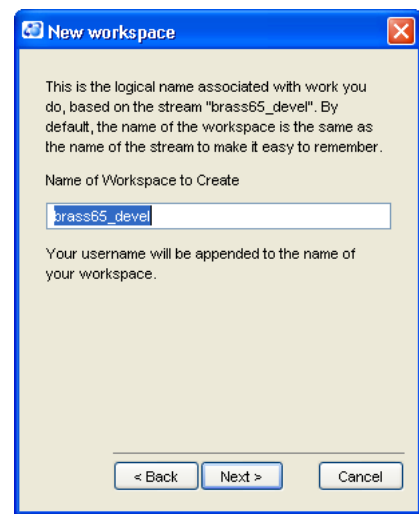
brass

workspaces:

brass_mary

brass_jjp

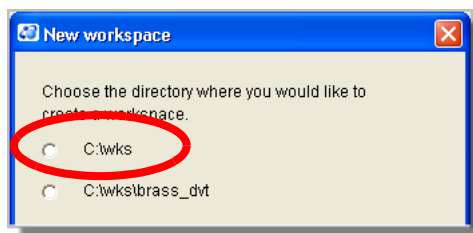
brass_derek



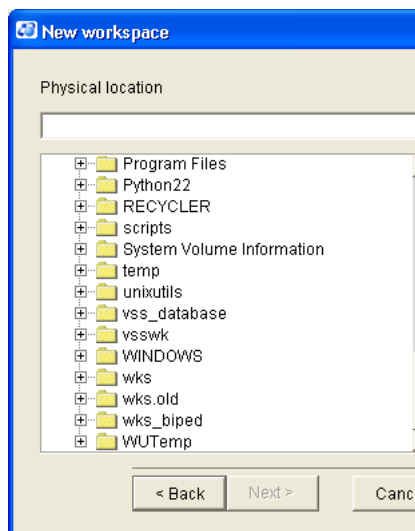
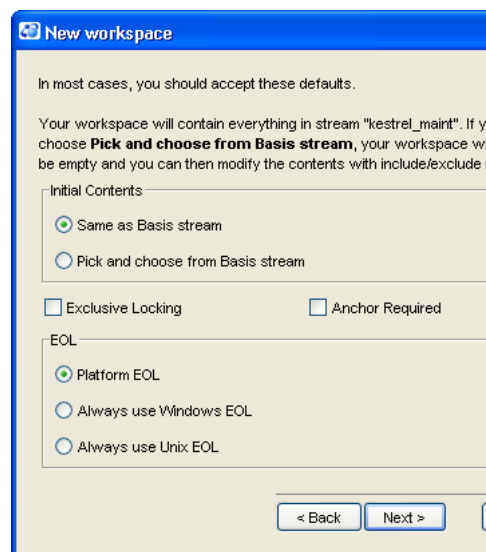
- **Workspace options:** Click **Next** to accept the defaults for these workspace options:

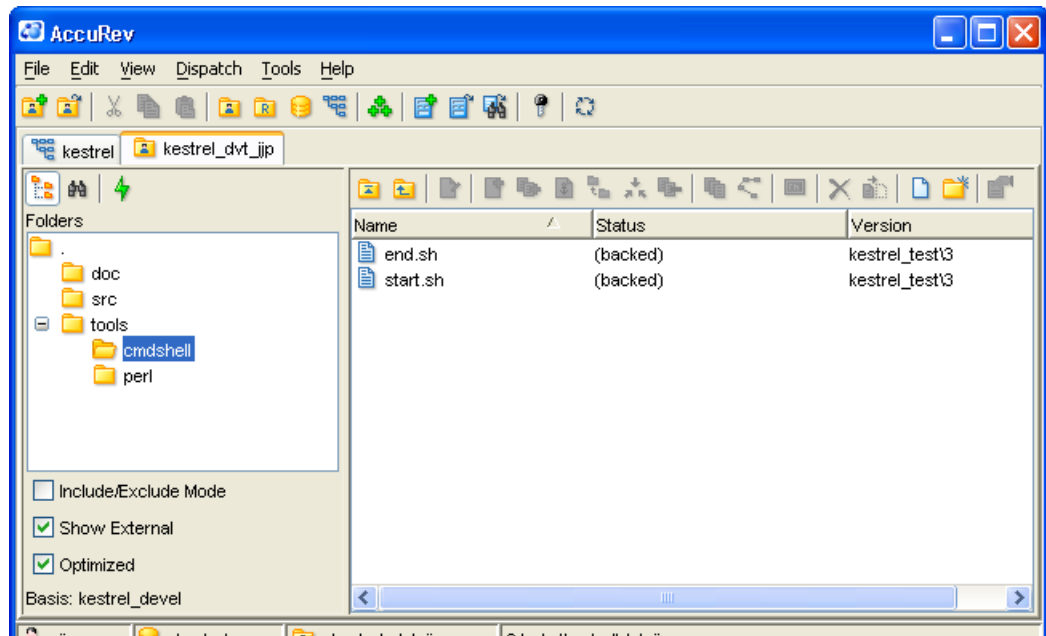
- **Type of Workspace:** A standard workspace contains a copy of each file in the depot. A sparse workspace contains copies of certain files and directories only.
- **Exclusive File Locking:** This feature suppresses the ability of multiple users to work on the same file at the same time. When one user starts working on a file, others are prevented from doing so. By default, this feature is turned off, enabling parallel development in this workspace.
- **Text-file line terminators:** By default, AccuRev uses the line terminator appropriate for the client machine's operating system. You can force Unix or Windows line terminators.
- **Physical Location:** Navigate to the top-level directory of the existing source tree that you wish to place under AccuRev's control. Then, click **Next**.

Choose the use the existing directory, rather than letting the wizard create an empty subdirectory, named after the workspace. Then, click **Finish**.




The existing source tree is converted to an AccuRev workspace. Its contents are presented in an “Explorer-standard” display, with navigation and details panes:

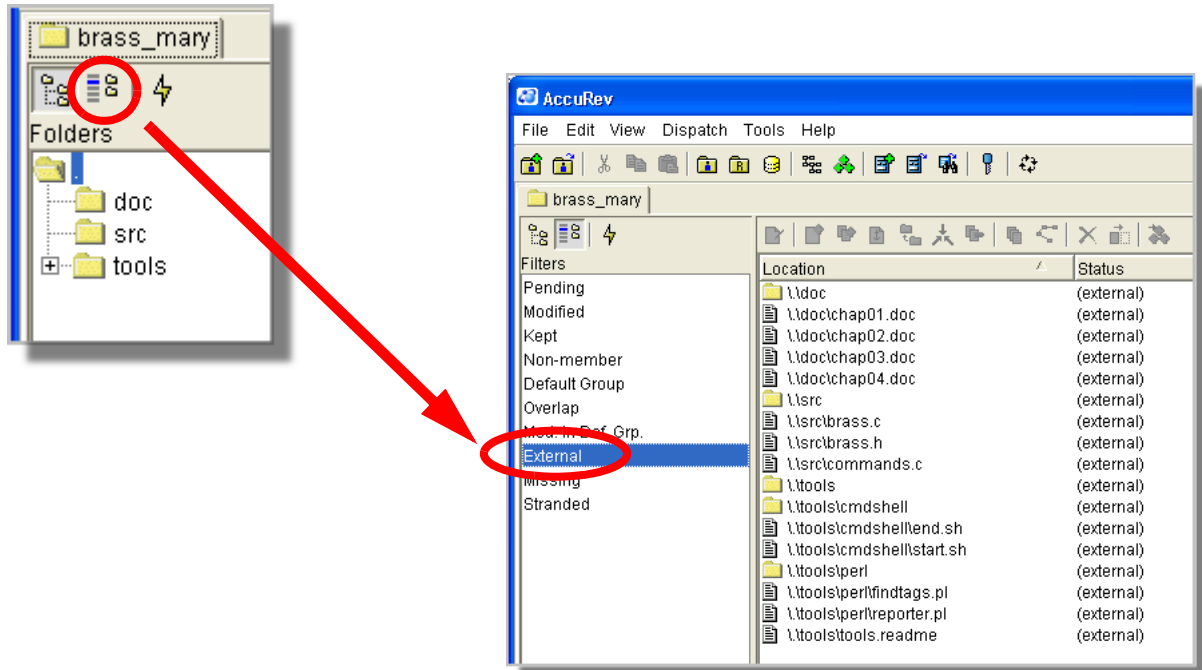





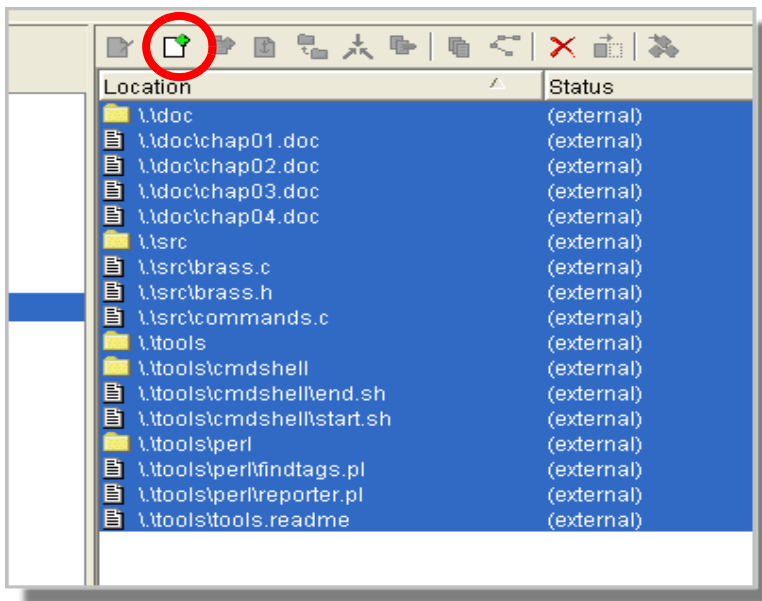
All the files in the source tree are now located within an AccuRev workspace. But the files are “external” (outside the depot): you have not yet told AccuRev that you want to track changes in these files.

The external file concept is an important one. In general, you don’t want to track changes to *every* file in a workspace; think of text-editor backup files, copies of mail messages, etc. This means you need to tell AccuRev exactly which files you want to version-control.

3. To see all of the files in the workspace that are external, switch to the external-files-only view of the workspace. Click the  **Searches View** button in the navigation pane, then select the **External** search:



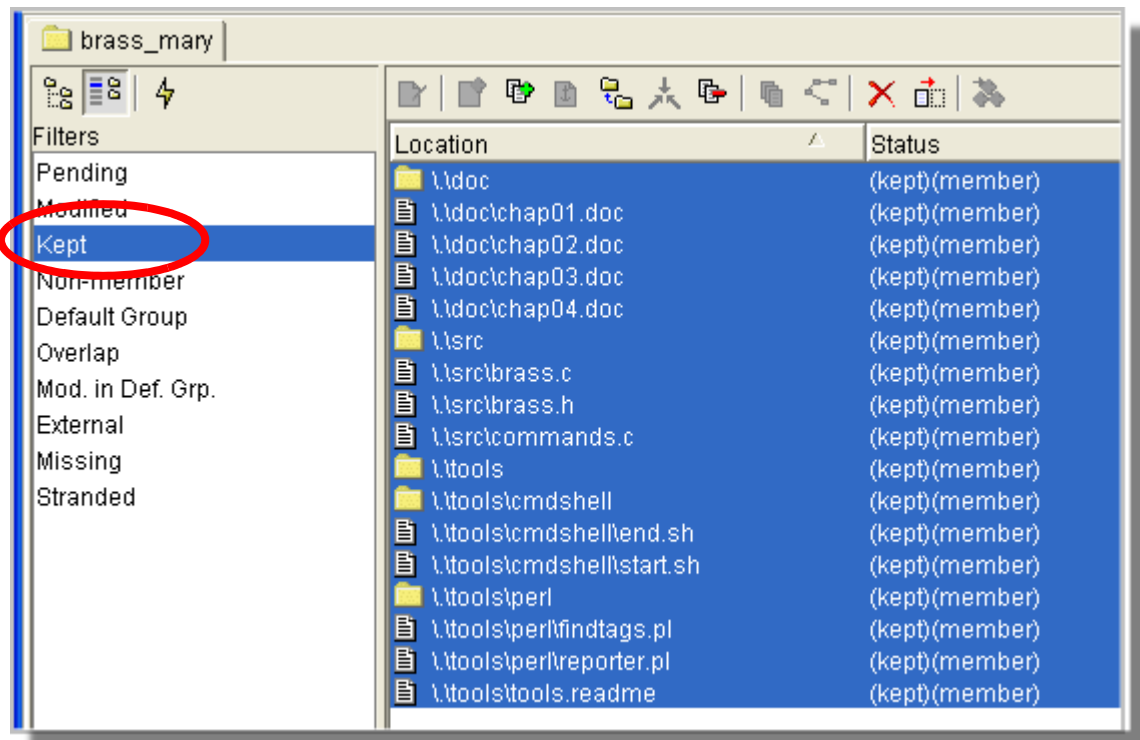
4. Placing files under version control is called “adding to the depot”. To do so, highlight all the files and click the  **Add elements to Depot** toolbar button:




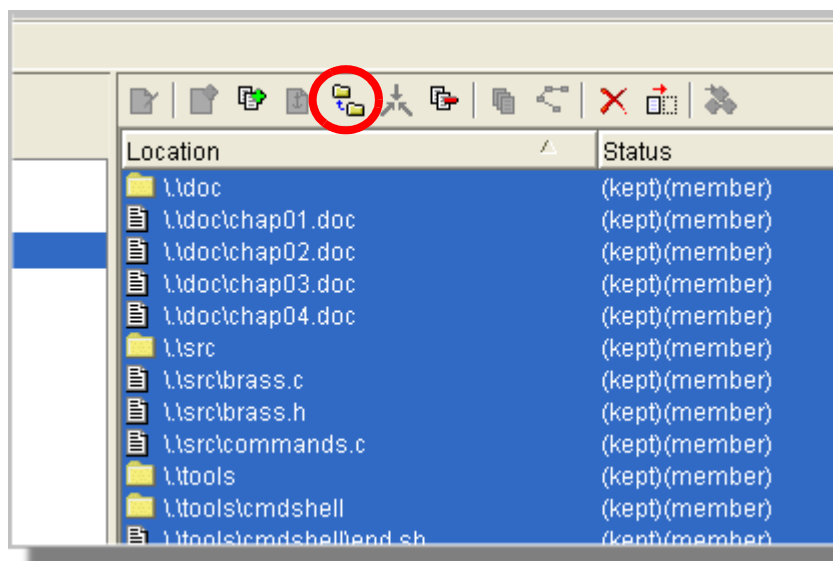
The “Log Message” is a comment string that will be stored in the add-to-depot transaction in the AccuRev database. It’s optional.

When the add-to-depot transaction completes, the external-files-only view becomes empty, because all the external files have been converted to AccuRev elements.

- When adding the files to the depot, AccuRev keeps an initial version of each one. You can verify this by switching to the kept-files-only view of the workspace:



- “Kept” versions are accessible only in your workspace — they’re “private”. To make these files available to other workspaces (“public”), you promote the kept versions. To do so, highlight all the files and click the  **Promote** toolbar button.




Beginning Development Work

Now that you've created a depot and have populated it with the files in your workspace, you're ready to begin making changes to the files you've added. There's no need to check out files; simply edit them. To edit a file, double-click it. Alternatively, right-click it and select **Edit** from the context menu.

If you wish to keep some work that you have done on a file, select the elements you want to keep, right-click on the selection and choose **Keep**. You are prompted for a comment, which is optional. This creates a new version in the repository in your private workspace stream. You can keep a file as often as you like.

When your changes are ready for other people to see, first keep the file, then promote it. To promote elements, select them, right-click on them, and select **Promote**. Promote will place the most recently kept version of the file in the backing stream for others to use. The next time they update their workspace they will get a local copy of your promoted file.

Seeing Other People's Changes

To update your workspace with other people's changes, click the  **Update** button in the navigation pane's toolbar.

Adding a New File

Simply edit a file with your favorite editor. When you're ready for AccuRev to add the initial version of your new file, right-click on it and select **Add to Depot**. Enter a comment (optional, just as with **Keep**).

You can promote the new file immediately. Alternatively, you can continue editing, and then keep and promote.

What's Next?

This chapter has covered only the most basic installation options and features of AccuRev. See the *AccuRev User's Guide* to learn more about AccuRev's capabilities. There are both CLI and GUI editions of this manual.

Converting to AccuRev from Directories Containing Baselevels

One popular method of supporting multiple releases without a versioning tool is to create parallel directory trees containing complete copies of the source code. The top-level directories of these trees are named for the releases. We'll examine a scenario in which existing baselevels are stored in directory trees whose top-level directories are:

```
d:\baselevels\gizmo1.0
d:\baselevels\gizmo2.0
d:\baselevels\gizmo2.5
```

It is fairly easy to incorporate the multiple baselevels into AccuRev. Make sure you read these instruction all the way through before trying it out.

Creating a Depot

First of all, you need to create an AccuRev repository, called a depot, with the **mkdepot** command:

```
accurev mkdepot -p <name-of-depot> -l <where you want the repository>
```

The depot is where AccuRev stores all of its information and hides it from public view. For instance, you might use

```
accurev mkdepot -p gizmo -l d:\acprojects\gizmo
```

This will create a project called **gizmo** and a stream called **gizmo**.

Processing the First Baselevel

Next you will need to populate **gizmo** with files from the first baselevel.

cd to the first baselevel directory:

```
cd \baselevels\gizmo1.0
```

Create a workspace to be used for importing files from the baselevel into AccuRev:

```
accurev mkws -w import -b gizmo -l .
```

(Note that the '.' is important here.)

This creates a workspace called **import** which is based on stream **gizmo** (currently empty) in the current location.

Get a list of all of the files that AccuRev doesn't know about (all of them):

```
accurev stat -x > xfiles
```

The **-x** stands for external.

You may see lots of files that you don't want to put under version control: object files, executables, text-editor backup files. etc. Exclude everything that you don't want AccuRev to keep track of. Note that case is important. You may need to include both upper and lower case versions of the following:

```
set ACCUREV_IGNORE_ELEMS=*.exe *.obj *.lnk *.err *.map      (and so on)
```

(The syntax for setting environment variables varies among operating systems and command-line processors. For more on ACCUREV_IGNORE_ELEMS, see *Using the ACCUREV_IGNORE_ELEMS Environment Variable* on page 39 in *AccuRev Technical Notes*.)

Try the **accurev stat -x** again.

Keep repeating this until **stat** lists exactly the set of files that you want AccuRev to keep track of.

Create initial versions of these files in the depot:

```
accurev add -x
```

This creates versions in the **import** workspace, but has not yet made them available to others working on the **gizmo** project.

To make these new files public, promote them:

```
accurev promote -k
```

Processing Subsequent Baselevels

Now, you need to “layer” the files in the next baselevel on top of the files that you’ve already placed under AccuRev control. First, enter a command that changes the definition of the **import** workspace:

```
cd d:\baselevels\gizmo2.0
accurev chws -w import -l .      (note that the '.' is important here)
```

In effect, you’ve moved the workspace to where the files are, instead of moving the files into the workspace! The files fall into these categories:

- Files that changed from **gizmo1.0** to **gizmo2.0**

To AccuRev, these files appear to be modified versions of the **gizmo1.0** files that you **added** and **promoted** in the preceding section. You can list all these files:

```
accurev stat -m
```

And you can keep the new versions of the files:

```
accurev keep -m -c "my comment"
```

- Files that didn't change from **gizmo1.0** to **gizmo2.0**

You don't need to do anything about this files. In particular, you don't need to **keep** new versions.

- Files that didn't exist in **gizmo1.0**, but do exist in **gizmo2.0**

These files are external, because AccuRev hasn't seen them before. (Just as *all* the files were external when you placed the first baselevel under version control.) Add the external files to the depot, just as you did in the preceding section:

```
accurev add -x
```

As above, you may want to use **stat -x** and the `ACCUREV_IGNORE_ELEMS` environment variable to filter out unwanted files before entering the **add** command.

Now, promote the new files and changed files:

```
accurev promote -k
```

You've now placed two baselevels under AccuRev control. Layering the third baselevel, **gizmo2.5**, on top of the second one is exactly the same as layering the second one on top of the first. Just repeat the steps in this section.

Handling Additional Baselevel-to-Baselevel Differences

In the discussion above, we broke a baselevel's "new layer" of files into three categories. This was a bit oversimplified — there are additional categories to consider.

- Files that existed in one baselevel, but were deleted in the next baselevel.

You can make such files disappear from the new baselevel by defuncting them:

```
accurev defunct <filenames>
```

- Files that were renamed from one baselevel to the next.

This will appear to be (1) a file that existed in one baselevel, but was deleted from the next baselevel, along with (2) a new file that didn't exist in the preceding baselevel. If you know that file **oldname.c** in the preceding baselevel was renamed to **newname.c** in the next baselevel, use this series of commands to make the connection:

```
ren newname.c SAVEME (Unix: use the mv command)
accurev move oldname.c newname.c
ren SAVEME newname.c
```

Now, AccuRev knows that the element formerly known as **oldname.c** is henceforth to be known as **newname.c** (until the next name change, that is!).

Cleaning Up

Finally, deactivate the **import** workspace:

```
accurev remove wspace import
```


Creating a Maintenance Stream

Many software development organizations have two main streams of development: work towards the next release, and maintenance of the previous release. Other SCM systems use “a branch based on a label” to accomplish this. AccuRev uses snapshots (static streams).

At the time of the release (say, “Gizmo Release 1.0”), create a snapshot:

```
accurev mksnap -s gizmo1.0 -b gizmo -t now
```

This creates a new snapshot called **gizmo1.0**. The snapshot contains whatever versions the **gizmo** stream contained at the time the **mksnap** command is executed. Subsequently, the **gizmo** stream can change as new versions are promoted to it, but the **gizmo1.0** snapshot never changes. Instead of **now**, you can specify any time in the past, such as **2001/05/18 10:10:24**.

For maintenance work on this release, create a new dynamic stream based on the snapshot:

```
accurev mkstream -s gizmo_maint -b gizmo1.0
```

Initially, **gizmo_maint** will be identical to **gizmo1.0**, but it will change as people promote changes to it.

Maintenance developers use workspaces based on the **gizmo_maint** stream. For instance, to make a maintenance fix, Mary might create a workspace like this:

```
accurev mkws -w gizmo_maint_mary -b gizmo_maint -l <wherever>
```

When Mary promotes her maintenance work, the changes will go to **gizmo_maint**.

All maintenance work is isolated from the main development stream, and vice-versa. Developers working on the next release create their workspaces like this:

```
accurev mkws -w gizmo_justine -b gizmo -l <wherever>
```

Changes promoted from **gizmo_justine** will go to the main development stream, **gizmo**. The changes won't appear in the **gizmo_maint** stream.

The Change Palette in the AccuRev GUI makes it easy to migrate changes back and forth between a main development stream (**gizmo**) and a maintenance stream (**gizmo_maint**).

Using the ACCUREV_IGNORE_ELEMS Environment Variable

Like most command-line programs, AccuRev's CLI tool, **accurev**, accepts one or more filename/pathname specifications as command arguments:

```
accurev keep base.h
accurev promote intro.doc chap*.doc
```

Such arguments cause the command to be invoked on a particular set of files.

But several **accurev** commands are capable of operating on *all* the files in the current workspace. For example, this command searches the entire workspace containing the current working directory, and lists the files that have not been placed under version control (external files):

```
accurev stat -x
```

Such “whole-workspace” commands are very powerful and useful, but they can be time-consuming. If your workspace contains many hundreds or thousands of files, you must wait while all the names are transmitted to the server machine, the AccuRev Server process determines the status of each file, and information on the matching files is returned to the client machine.

This large list of files to be processed may contain a significant number of “don't care” files. For example, a search for external files is probably intended to locate source files (with suffixes like **.c** or **.cc** or **.java** or **.bas**) that you've forgotten to place under version control. You probably don't care about files with suffixes like **.exe** (executables built in the source directory), **.bak** (editor backup files), **.msg** (copies of mail messages, and so on — because you don't intend to place them under version control.

You can use the environment variable **ACCUREV_IGNORE_ELEMS** to specify one or more patterns (or even individual filenames/pathnames). When it executes certain “whole-workspace” commands, the **accurev** tool ignores all external files that match this specification. For example, setting **ACCUREV_IGNORE_ELEMS** to the following value causes the **stat -x** command to ignore all **.exe** and **.bak** files:

```
*.exe *.bak
```

ACCUREV_IGNORE_ELEMS is also used — in a slightly different way — by certain commands that process a particular set of files, instead of the whole workspace. The following sections explain the details of using this environment variable.

Eligible “Whole-Workspace” Commands

The following **accurev** commands use the value of `ACCUREV_IGNORE_ELEMS` to filter names:

Command	Description
stat -x	List external (non-version-controlled) files
stat -m	List files that have been modified
stat -n	List files that have been modified, but are not <u>active</u> from AccuRev’s viewpoint (that is, are not in the workspace’s <u>default group</u>)
stat -p	List files that are pending promotion — files that have been modified, along with files with (kept) status
add -x	Place external files under version control

These commands apply to the entire workspace if you *don’t* specify any filenames/pathnames or wildcard patterns on the command line:

```
accurev stat -n          (“whole-workspace” command)
accurev stat -n *.doc    (pattern specified; not a “whole-workspace” command)
```

When applying these commands to an entire workspace, the **accurev** tool, running on the client machine, uses `ACCUREV_IGNORE_ELEMS` to filter the list of filenames *before* sending the list to the AccuRev Server process. This can significantly reduce the amount of network traffic, and also reduce the amount of file-status computation the Server process must perform.

Note: by default, the **stat** command also uses a timestamp-based optimization to reduce the number of files it must process. For details, see [Optimized Search for Modified Files](#) on page 155 in the *AccuRev User’s Guide (CLI)*.

GUI Counterparts to the “Whole-Workspace” Commands

The AccuRev GUI also uses the `ACCUREV_IGNORE_ELEMS` environment variable. The Searches view of the File Browser corresponds to the various “whole-workspace” forms of the **accurev stat** command. Thus, `ACCUREV_IGNORE_ELEMS` is used by these search criteria:

- External (**stat -x**)
- Modified (**stat -m**)
- Non-member (**stat -n**)
- Pending (**stat -p**)

Commands that Don’t Apply to the Whole Workspace

The **stat** and **add** commands above accept filename/pathname specifications, in several forms:

- Individual filename: **chap01.doc**
- Individual pathname: **doc/chap01.doc** or **./widgets/doc/chap01.doc**
- Wildcard pattern: ***.doc** or **docs/*.doc**

- list-file: **-l my_list_of_files**

Such specifications restrict the command to processing a certain set of files, not the whole workspace (even if you also specify **-x**, **-m**, **-n**, or **-p**). Similarly, the **files** command introduced in AccuRev Version 3.2 processes a certain set of files, not the whole workspace.

For these commands, the **accurev** tool still uses `ACCUREV_IGNORE_ELEMS` — but in a way that is less efficient than for whole-workspace commands:

1. Send the list of files that match the filename/pathname specification to the AccuRev Server.
2. Use `ACCUREV_IGNORE_ELEMS` to filter the data that the Server returns. Only names of external files are filtered out; files that are AccuRev elements remain in the listing, even if they match the value of `ACCUREV_IGNORE_ELEMS`.

This may produce similar, or even identical results as a whole-workspace command, but the fact that an unfiltered list is sent to, and processed by, the Server means that overall performance won't be as good.

Values for `ACCUREV_IGNORE_ELEMS`

The value of the `ACCUREV_IGNORE_ELEMS` environment variable must be a SPACE-separated list of filenames, pathnames, and wildcard patterns. Some examples:

```
*.exe
*.exe *.doc
manuals/*.doc
*.doc README.html
```

You can use either or both of the “standard” wildcards:

- **?** matches any one character
- ***** matches any sequence of characters (including a zero-length sequence)

Be careful — the asterisk (*****) wildcard works a bit differently here than in standard Unix and Windows command processors. It matches any number of characters, including the directory separator (**/** or ****). With standard Unix and Windows command processors, the scope of ***** is restricted to a single pathname component.

So ***.doc** matches any of these names:

```
chap01.doc
manuals/chap01.doc
widgetproj/src/manuals/usergd/chap01.doc
```

The pattern **manuals/*.doc** matches any of these names:

```
manuals/chap01.doc
manuals/usergd/chap01.doc
```

... but not this name:

```
widgetproj/src/manuals/usergd/chap01.doc
```

The pattern `*/manuals/*.doc` (or `*manuals/*.doc`) *does* match the last name.

You can use the following pattern to ignore all items in the directory tree(s) named **usergd**:

```
*/usergd*
```

Specifying Directories and Their Contents

A typical application of `ACCUREV_IGNORE_ELEMS` is to have **stat -x** (“list all external files”) ignore temporary build directories. That is, you want the listing to exclude both the directories themselves and all the files within those directories. If the build directories are named **build_001**, **build_002**, etc., you might be tempted to use this pattern:

```
*/build_??*/*
```

But this pattern matches only the *contents* of the directories, not the directories themselves. Instead, use the following value for `ACCUREV_IGNORE_ELEMS`:

```
*/build_??? */build_??*/*
```

(The single pattern `*/build_??*` would match both directories and their contents. But it also might coincidentally match names of some source files, such as **lib/build_end.c**.)

Setting ACCUREV_IGNORE_ELEMS on a Unix System

When setting the `ACCUREV_IGNORE_ELEMS` environment variable on a Unix or Linux system, be sure to single-quote or double-quote the value, in order to protect any wildcard characters from being expanded by the shell:

```
export ACCUREV_IGNORE_ELEMS="*.exe *.doc"    (Bourne shell family)
setenv ACCUREV_IGNORE_ELEMS "/*.exe *.doc"    (C shell family)
```

To determine the current value of `ACCUREV_IGNORE_ELEMS`, use either of these commands:

```
env | grep ACCUREV_IGNORE_ELEMS    (or a shorter 'grep' pattern)
echo "$ACCUREV_IGNORE_ELEMS"        (don't forget the quotes!)
```

Setting ACCUREV_IGNORE_ELEMS on a Windows System

On a Windows system, you can set the `ACCUREV_IGNORE_ELEMS` environment variable in the System applet (on the Control Panel). Alternatively, use the **set** command in a Command Prompt window:

```
set ACCUREV_IGNORE_ELEMS=*.exe *.doc    (no quotes!)
```

Don't use quote characters, even if the value includes SPACES.

To determine the current value of `ACCUREV_IGNORE_ELEMS` in a Command Prompt window, use either of these commands:

```
set
echo %ACCUREV_IGNORE_ELEMS%
```


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 perfectly 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.

Setting Up Multiple Server Machines

There are no special considerations, and no special procedures, for setting up multiple machines to run AccuRev Servers. You can perform a server installation on as many hosts as desired 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

Performing a client installation — or a combined client/server installation — 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
```

Be sure to select a *custom* client installation, not a *typical* one. In a custom installation, you are prompted to supply the information to be stored in the **acclient.cnf** file — both the “host” (name of server machine) and “port” (IP port number). In a *typical* installation, the configuration file automatically gets a line specifying port 5050 on the local machine.

If more than one server has already been set up, you'll need to pick one of them during a custom 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
...
```

Configuring a Client Machine to Use Multiple Servers

The client configuration file, **acclient.cnf**, is an editable text file. You can modify it 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"). If an existing AccuRev server isn't listed in a machine's **acclient.cnf** file, AccuRev client programs running on that machine won't be able to communicate with that server.

Note: as of AccuRev Version 3.0.2, the file must not contain any empty lines; be sure to check the end of the file.

In what order should the servers be listed? For users of the AccuRev GUI (**acgui**), it doesn't matter: users can switch among all the servers with the GUI's **Tools > Change Active Server** command. 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.

Note: the GUI's **Tools > Change Active Server** command revises the **acclient.cnf** file, moving the line for the new active server to the beginning of the file.

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.)

Note: It's 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.

Suppose you want to create one workspace for use with a depot on server **venus**, and another workspace for use with a depot on server **pluto**. Here's how:

1. Make sure the **acclient.cnf** file lists **venus** first.
2. Create a workspace with **mkws**, specifying a stream in any of **venus**'s depots as the backing stream.
3. Edit the **acclient.cnf** file, so that the **pluto** entry is first.

4. Create a workspace with **mkws**, specifying a stream in any of **pluto**'s depots as the backing stream.

Many AccuRev CLI commands (examples: **keep**, **promote**, **merge**, **hist**) require that your current working directory be within an existing workspace. By default, such commands succeed only if the server associated with the workspace is the first one listed in the machine's **acclient.cnf** file. So, for example, an error would occur if we continued the example above as follows:

5. Go to the workspace created in Step 2.
6. Execute the command **accurev hist**.

An error would occur because the workspace is associated with the **venus** depot, but the first line in the **acclient.cnf** file still lists the server on **pluto** (changed in Step 3). Are we saying that every time you want to use the **venus** workspace, you need to make sure the **acclient.cnf** file lists the **venus** server first? Well, that's the way AccuRev works by default; but fortunately, there's a better solution to the "switching servers" problem.

The 'wspaces' File

If the workspaces you've created on your client machine are not all associated with the same AccuRev server, then you'll probably want to use a **wspaces** configuration file. This file records the workspace/server association for one or more workspaces. The AccuRev CLI client program, **accurev**, uses this file to decide which AccuRev server to use:

- If the current working directory is within one of the workspace-root-directory locations listed in the **wspaces** file, the CLI command is directed to the associated server listed on the same line of the file.
- Otherwise (including the case in which no **wspaces** file exists), the CLI command is directed to first server listed in the machine's **acclient.cnf** file. This is the default behavior that we've mentioned in the sections above.

If your site has a single AccuRev server (or if your workspaces are all associated with a single server), there is no need for a **wspaces** file: the default behavior directs each CLI command to the one and only server listed in your machine's **acclient.cnf** file.

Format of the 'wspaces' File

The **wspaces** file is a text file, each line containing four whitespace-separated fields:

- workspace name
- full pathname of workspace's root directory
- server machine name (or IP address)
- IP port number on which AccuRev server program listens

Thus, each line records the association between one workspace and its AccuRev server.

For example, here's a **wspaces** file recording workspaces associated with servers **venus** and **pluto**:

```
gizmo_dvt_jjp      /usr/jjp/gizmo_dvt      venus  5050
frammi_2.3maint_jjp /usr/jjp/frammi_2.3_maint pluto  6678
```

Some notes:

- The workspace pathname (2nd field) must match the pathname listed by the **accurev show wspaces** command. The pathname always uses forward slashes (/), not backslashes (\), even on Windows machines.
- The last two fields must exactly match an existing entry in the machine's **acclient.cnf** file — except for the fact that **acclient.cnf** uses a colon (:) to separate the machine name and the IP port number, whereas **wspaces** uses whitespace to separate them.
- Workspace names and pathnames that contain SPACE characters are not handled correctly by this facility.

Location of the 'wspaces' File

Each user has his own **wspaces** file, recording information about his own workspaces. The **wspaces** file must be located under your home directory, within subdirectory **.accurev**. On Windows systems, your home directory is determined by concatenating the values of environment variables HOMEDRIVE and HOMEPATH. On some versions of Windows, you must set these environment variables yourself; they are not set automatically by the operating system.

The -H Command-Line 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
```

Note that 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.

Techniques for Sharing Workspaces

This note describes two techniques for accessing the same workspace from multiple machines.

Accessing a Windows-Based Workspace From Multiple Clients

It is easy to have multiple AccuRev users share a workspace that is physically located on an MS-Windows machine. Designate the top-level directory of the workspace tree as a Windows “shared directory”. Alternatively, share a directory that is above the workspace directory.

Each user who wants to use the workspace on his Windows machine gains access to the workspace location through the **Map Network Drive** command (or the **net use** command in a Command Prompt window). Even on the machine where the workspace resides, a user must access the storage through a shared directory.

For example, suppose a workspace tree is located at **C:\netshare\brass_jjp**, where **C:\netshare** is a shared directory. To access the workspace from that machine or another Windows machine in the network:

1. Map **C:\netshare** to a drive letter, say **X:**. (Users on different machine can map **C:\netshare** to different drive letters.)
2. Set the current working directory to **X:\brass_jjp**. (A user on another machine might go to **Y:\brass_jjp** or **R:\brass_jjp**.)

Accessing a Unix-Based Workspace from Windows Clients

Users sometimes need to access a workspace from both Windows and Unix clients. Starting in AccuRev 3.5.5, a workspace physically located on a Unix machine (in a network-accessible exported file system) can be accessed from both Unix and Windows client machines.

The key is a simple text file, **share_map.txt**, located in the AccuRev **site_slice** directory. Each line of this file consists of three TAB-separated fields. Here’s a sample entry:

```
jupiter      accwks      /public05/accurev_workspaces
```

The first field (**jupiter**) names a Unix machine where one or more workspaces are (or will be) physically stored. The third field (**/public05/accurev_workspaces**) indicates the Unix file system where those workspaces are located. This file system must be suitably exported for network access.

The second field (**/accwks**) specifies the Windows share name that a Windows machine will use to access the exported Unix file system. All Windows machines must access the file system using the same share name.

Note: the separator between fields must be single TAB character — don’t use SPACES. If a specification (e.g. a share name) includes a SPACE character, do not enclose the specification in quotes.

Example: As indicated by the sample entry above, Windows machines in the network use the share name **accwks** to access the file system **/public05/accurev_workspaces** located on Unix host **jupiter**. User John works on a Unix machine that mounts the file system as **/mnt/accurev_workspaces**. He can create a universally-accessible workspace with this command:

```
accurev mkws -w wks_john -b dvt_stream -l /mnt/accurev_workspaces/wks_john
```

User Mary works on a Windows machine. She accesses the file system through the share name **accwks**, which is mapped to network drive **W:**. She can create a universally-accessible workspace with this command:

```
accurev mkws -w wks_mary -b dvt_stream -l W:\wks_mary
```

To enable sharing for an existing workspace, first make an entry in the **share_map.txt** file. Then use the **chws** command on the Unix system where the workspace resides, specifying the existing location with the **-l** (“dash-ell”) option. For example:

```
accurev chws -w wks_john -l /mnt/accurev_workspaces/wks_john
```

What's the Difference between Populate and Update?

AccuRev users sometimes confuse the two commands **Populate** and **Update**. These commands seem similar because they both bring new data into your workspace. But they are quite different, both in their usage pattern — most people use **Update** far more often — and in what they accomplish. Understanding the difference between these two commands will enable you to choose the right command at the right time (always useful!), and will deepen your knowledge of how AccuRev really works.

Note: the **accurev** command-line program invokes the **Populate** command using the nickname **pop**.

This note starts with a brief statement of the difference between **Populate** and **Update**, along with a few examples. Then, we present a full discussion of the data structures and mechanisms involved in these commands.

In a Nutshell ...

The essential difference between **Populate** and **Update** concerns time. Roughly speaking, your workspace contains an informal “baseline” (the contents of the shared backing stream, at a particular moment) plus “changes” (the modifications that you make to some of the files). The **Update** command advances the workspace’s baseline from the time of the workspace’s last update to the present moment. This incorporates into the workspace data recently placed in the backing stream by other team members.

Note: Remember, this is rough description. **Update** actually tracks the workspace’s baseline in terms of AccuRev transactions, not timestamps.

The **Populate** command doesn’t advance a workspace’s baseline at all, but leaves it “stuck in the past”. Instead, **Populate** simply restores the appropriate “old” version of one or more elements that are currently missing from the workspace.

The two commands also differ in their scope: **Update** always processes the entire workspace; **Populate** processes just a selected set of elements or directory subtrees.

The capsule description above uses imprecise language, such as “advancing the workspace’s baseline” and “old version”. The following description is more precise, using AccuRev-specific terminology. The terms are explained fully in section *Data Structures Used by Populate and Update* on page 51 below.

The **Update** command changes both the workspace stream and the workspace tree:

- It advances the workspace stream’s update level to the depot’s most recent transaction — say, from current update level 32155 to new update level 34002. This allows a new set of versions — in this case, some or all the versions created by transactions 32156 through 34002 — to flow into the workspace stream from the backing stream.
- It copies the contents of the workspace stream’s new versions from the repository’s file-storage area to the workspace tree. When processing a sparse workspace, **Update**

bypasses elements with **(missing)** status. (It assumes that you have chosen not to load these elements into the workspace tree.)

By contrast, the **Populate** command changes the workspace tree only, not the workspace stream. In particular, it doesn't change the workspace stream's update level. **Populate** merely fixes a discrepancy between the workspace stream and the workspace tree: a certain version of a file is in the workspace stream, but there is no actual file in the workspace tree — that is, the file's status is **(missing)**. To fix this situation, you invoke **Populate**, which copies the version currently in the workspace stream to the workspace tree.

Note: it would be incorrect to conclude that **Update** *never* processes **(missing)** elements, and that **Populate** *only* processes **(missing)** elements. Examples 3 and 4 below show that exceptions exist for both these “rules”.

Example 1: Standard Update Scenario

You've just finished a coding project, so you're not actively working on any files in your workspace. Other team members create new versions of files **red**, **white**, and **blue** in their workspaces, then promote those versions to the team's backing stream. You invoke the **Update** command, which copies the most recent versions of **red**, **white**, and **blue** from the backing stream to your workspace.

Example 2: Restoring a Deleted File (“missing” by accident)

Since you have complete control over the files in the workspace tree, it's easy to accidentally delete a version-controlled file with an operating-system command or a third-party tool. If you do this, AccuRev knows that the file *should* be there, because a version of the element still exists in the workspace stream. Thus, the File Browser continues to list the element, but shows it as **(missing)** from the workspace tree. You select the element and invoke **Populate** to fix the accidental deletion.

Example 3: Adding Data to a Sparse Workspace (“missing” by design)

Note: this example becomes less likely in Version 3.5, which no longer creates sparse workspaces. The new include/exclude facility replaces the functionality of sparse workspaces. See *Working in Include/Exclude Mode* on page 61 of the *AccuRev User's Guide (GUI)*.

A sparse workspace is one in which only some of the elements in the workspace stream are loaded into the workspace tree. That is, some of the files are missing by design — because you don't want them in this particular work area. But it's all the same to the File Browser: it shows the non-loaded elements as **(missing)**. As above, you select the elements and invoke **Populate**. In this case, instead of fixing an accidental deletion, it loads previously non-loaded elements into the workspace tree.

What would **Update** do in the situations of Examples 2 and 3?

- If you accidentally delete a file in a standard workspace, a subsequent **Update** *might* copy a new version of the file to your workspace tree; it does so only if someone else promoted a new version of the file to the backing stream since your last **Update**.

- But in a sparse workspace, **Update** *never* processes an element with **(missing)** status.

Example 4: Handling Active Elements

Update and **Populate** differ in how they handle elements that are active (are in the workspace's default group). The **Update** story is simple: it *never* overwrites the file in the workspace tree.

Populate usually doesn't overwrite the file, but there are a couple of cases to consider.

- It doesn't *need* to overwrite a file that you've kept and *not* subsequently edited, because the active version in the workspace stream is identical to the file in the workspace tree.
- But if you have subsequently edited the file in the workspace tree, so that the element status is **(modified)(member)**, then you can order **Populate** to overwrite the file and clobber those subsequent edits. This can also happen with a file that you've edited, but never kept, so that its status is **(modified)**.

Be careful — **(modified)** files will also be overwritten if you invoke **Populate** with both the **Recursive** and **Overwrite** options on a directory that directly or indirectly contains the active element.



Example 5: A Tale of Two Files

Let's see how **Update** and **Populate** differ in this situation:

You have a standard (non-sparse) workspace that is completely up to date. You delete two files, named **blue** and **green**. Someone creates a new version of **blue** in another workspace, and then promotes it to your workspace's backing stream.

If you select both **blue** and **green** in the File Browser, then invoke **Populate**, the two files that you deleted are restored to the workspace tree. This does *not* bring in the new backing-stream version of **blue**, because that version is not in the workspace stream — it's too new, having been created after your workspace's most recent update.

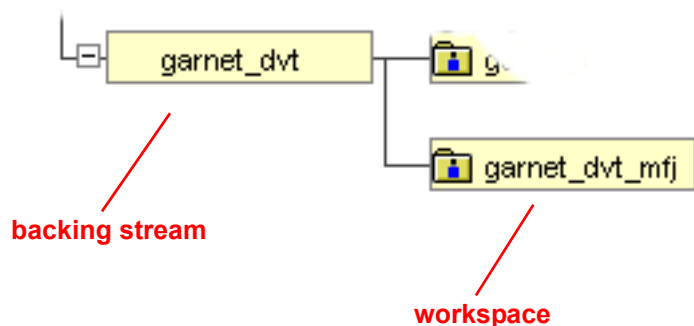
If you invoke **Update** instead of **Populate**, the workspace tree gets the new version of **blue**. No version of **green** is copied to the workspace tree, because **Update** only handles new versions — ones that enter your workspace stream as a result of advancing its update level.

Data Structures Used by Populate and Update

The simplicity of AccuRev's day-to-day usage model stems, in large part, from the fact that you don't need to worry about the "big picture" of your organization's development scheme and process. Instead, you only need to concern yourself with:

- the workspace in which you maintain your private copies of version-controlled files
- the workspace's backing stream, a "data switchboard" that organizes the sharing of files' changes with other members of your development team

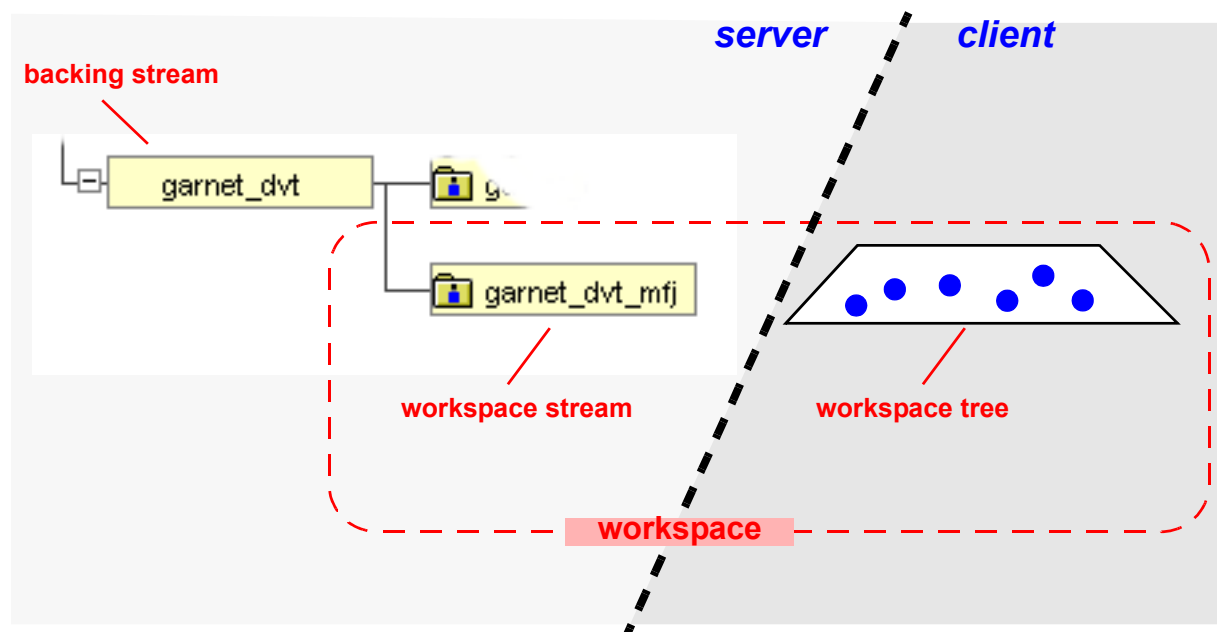
The illustration below shows how a workspace and its backing stream typically appear in the AccuRev StreamBrowser:



But a workspace actually consists of two parts:

- the workspace tree, an ordinary directory tree (“just a bunch of files”)
- the workspace stream, which contains all of the workspace’s configuration management information

So the picture looks more like this:



The above illustration shows one important difference between a workspace’s two parts: the workspace tree lives in “client space”, while the workspace stream lives in “server space”. The following table summarizes all the important differences.

Workspace Stream	Workspace Tree
Resides in the database of an AccuRev depot, located on the AccuRev server machine	A standard directory tree, located on your machine (or in other user-accessible storage)

Workspace Stream	Workspace Tree
Managed by the AccuRev Server process	Managed by you, the individual user
Contains all version control and configuration management information for the workspace, such as version-IDs	Contains no version control or configuration management information
Contains no actual files, just pointers to files in permanent storage	Contains <i>only</i> files and directories, which you can edit, compile, copy, etc.
Operating system commands and tools never change data here	Operating system commands and tools can change data here

The sections below expand on these differences.

How the Data Structures Get Their Data

Each of the data structures introduced above — backing stream, workspace stream, and workspace tree — is different in the way it gets changes (i.e. new data) from other parts of the development environment.

Backing Stream

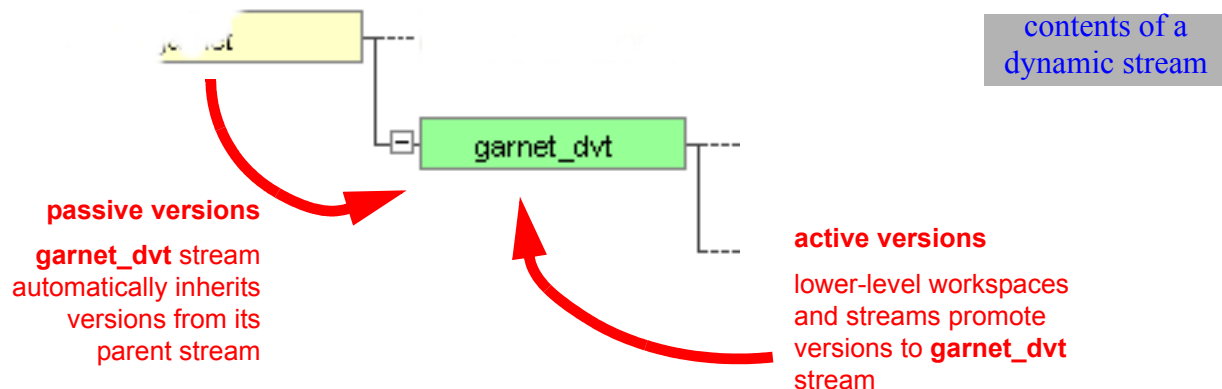
The backing stream is, in most cases, a dynamic stream. (It can also be a snapshot or a time-based stream.) A dynamic stream is a changing configuration of its depot. At any given moment, it (logically) contains a simple table that indicates particular versions of a set of elements. For example:

Element	Version-ID
doc	garnet_dvt\1
doc\chap01.doc	garnet_dvt\5
doc\chap02.doc	garnet\3
doc\chap03.doc	garnet_dvt\2
src	garnet\1
src\garnet.c	garnet_dvt\12
src\commands.c	garnet_dvt\7
tools	garnet\3
tools\start.sh	garnet_dvt\6
tools\end.sh	garnet\2

At any given moment, a dynamic stream's versions fall into two categories:

- **passive versions:** versions that the stream inherits from its parent stream. Inheritance is automatic and instantaneous: as soon as a new version enters the parent stream, it is inherited at once by the child stream.

- **active versions:** versions that have been **Promoted** to the stream, (usually) from lower-level workspaces and substreams. This set of versions constitutes the stream's default group.



In the configuration table above, all the **garnet_dvt\...** version-IDs indicate active versions in the **garnet_dvt** stream. All the **garnet\...** version-IDs indicate passive versions, inherited from the parent stream, named **garnet**.

Workspace Stream

The workspace stream is the “behind the scenes” part of your workspace. It lives inside the AccuRev repository, located on the AccuRev server machine. In many ways, it resembles a dynamic stream:

- It’s a changing configuration of one of the repository’s depots.
- It logically contains a particular version of some or all of the depot’s elements.
- It doesn’t contain actual files, but is logically just a table of elements and version-IDs.

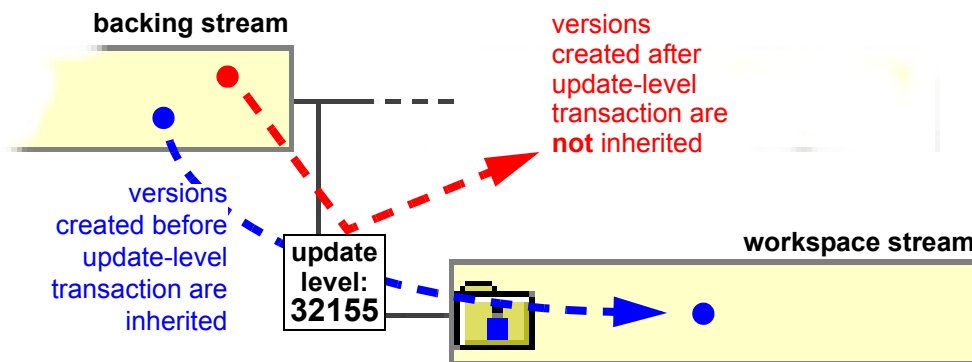
Note: when creating a new version of a file, the **Keep** command copies the file to the repository’s file-storage area, not to the workspace stream itself. The workspace stream just gets a version-ID for the new version; the version-ID serves as a pointer to the file in the file-storage area.

- It contains active versions, created by explicit user commands: **Add**, **Keep**, **Rename**, **Defunct**, etc. The new versions preserve in the repository the changes that you’ve made to files in your workspace tree. (There’s only one way to create an active version in a dynamic stream: the **Promote** command.)
- It also contains passive versions, inherited from its parent stream, the workspace’s backing stream.

The last item is where the crucial difference between workspace streams and dynamic streams comes into play. The versions inherited by the workspace stream are not the ones *currently* in the backing stream, but the versions that *were* in the backing stream when the workspace was last updated. This is called the workspace’s update time.

More precisely, AccuRev records the number of the depot's most recent transaction — say, transaction #32155 — as the new update level of the workspace stream. So we can rephrase the principle:

*The workspace stream inherits from the backing stream versions that were created in transactions up to and including the workspace's **update level**.*

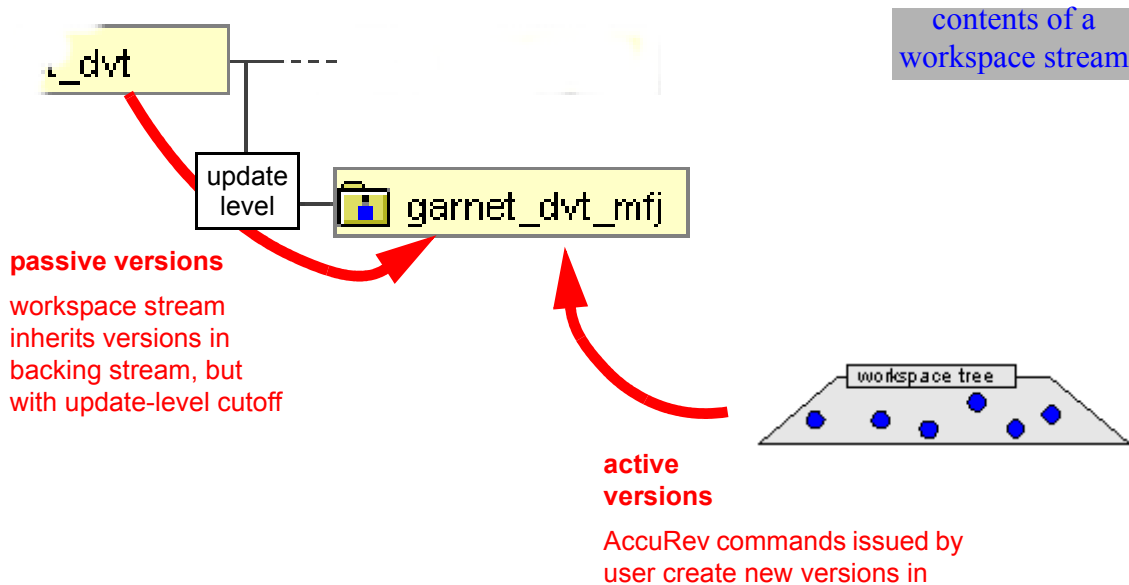


Note: the update level of a workspace stream is very much like the optional basis time of a dynamic stream. Both mechanisms restrict the flow of versions to a child stream from its parent stream, based on a point in the depot's development history.

Thus, a workspace stream is not updated dynamically when changes occur to its parent stream. It gets new versions from the backing stream only when you issue an **Update** command. This is how AccuRev implements the workspace's user-controlled “privateness”, isolating it from the changes regularly being recorded in the backing stream by other team members.

To summarize: at any given moment, a workspace stream contains:

- a set of passive, inherited versions, created in transactions that do not exceed the workspace's update level. (A file that you've **Promoted** since the last update is an exception. The version is passive, but was created after the workspace's update time.)
- a set of active versions, which you've created in that workspace with such AccuRev user commands as **Add**, **Keep**, **Rename**, and **Defunct**.



Roughly speaking, the set of versions in the workspace stream indicates what data currently *should* be in your workspace tree. Examples:

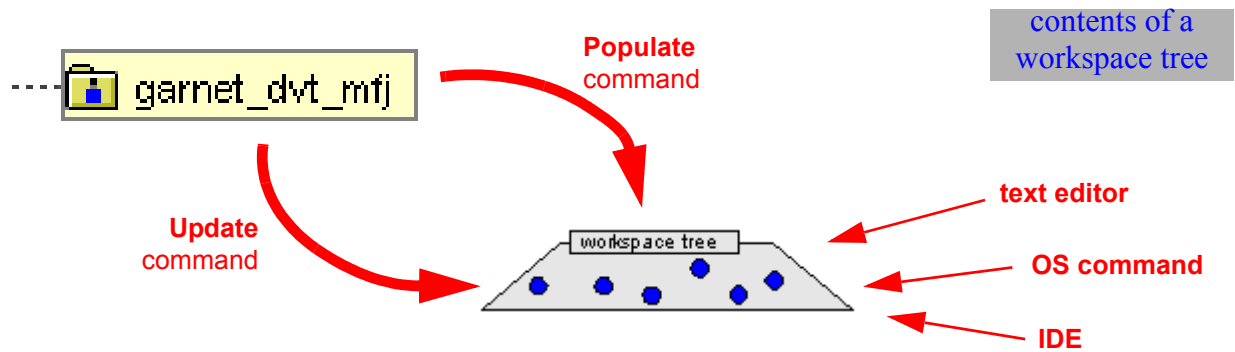
- The workspace stream contains active version **garnet_dvt_mfj\9** of file **commands.c**, which you created with the **Keep** command. This means your workspace tree should contain a file **commands.c** that matches the repository file referenced by version-ID **garnet_dvt_mfj\9**.
- The workspace stream contains passive version **garnet_dvt\6** of file **start.sh**. This means your workspace tree should contain a file **start.sh** that matches the repository file referenced by the version in the backing stream, **garnet_dvt\6**.

If you modify a file without **Keeping** it (or modify it again after **Keeping** it), the file in the workspace tree does not exactly match the version in the workspace stream. AccuRev signals this by reporting the file's status as **(modified)**.

Workspace Tree

The workspace tree is an ordinary directory tree, located in your personal disk storage. You can modify the contents of the workspace tree in two basic ways:

- By invoking operating system commands, text-editing tools, IDEs, etc.
- By invoking AccuRev commands to copy existing versions from the repository to the workspace tree. This can either overwrite existing files in the workspace tree or add new files. Both the **Populate** and **Update** commands copy versions from the workspace stream to the workspace tree. (So do a couple of other commands, such as **Send to Workspace**.)



The Update Algorithm

This section describes the processing of the **Update** command in detail. It's not as simple as “get all the new versions”, because AccuRev takes care not to overwrite version-controlled files that you have changed, but whose changes have not yet been preserved in the repository.

1. AccuRev first searches the workspace tree for such “at-risk” files, by performing a CLI **stat** (file status) command on your workspace:
 - It uses the **-n** option to **stat**, which restricts the search to version-controlled files that you have modified but are not in the workspace's default group. (It's safe to ignore default-group files, because these are never affected by an **Update**.)
 - It uses timestamp optimization to speed the file search: it ignores files whose timestamps precede the workspace's update time (the time that the workspace was most recently updated). If you modify a file by overwriting it with a file with an old timestamp, the file will be ignored in this step. This can cause problems in Step 5 below.
 - To make sure that a file with a recent timestamp has actually been modified, it compares the file with the version currently in the workspace stream by performing a checksum. (This means that simply modifying a file's timestamp with a **touch** command won't prevent the file from being overwritten by **Update**. You have to make a real change to the file.)
2. If the preceding step found any at-risk files — modified, by not in the default group — the **Update** command is immediately cancelled.
3. AccuRev notes the number of the repository's latest transaction, and sets that number as the workspace's target update level.
4. AccuRev decides which recently created versions in the repository should be delivered to the workspace tree. A version is a candidate for delivery if it became visible in the workspace's backing stream by one of the transactions between the workspace's *current* update level and the newly set *target* update level.
5. AccuRev attempts to deliver all those versions to the workspace tree. Each time it is about to overwrite a file in the workspace tree, AccuRev first makes sure it won't be “clobbering”

unpreserved changes: it checksums the existing file to confirm that it matches the current version in the workspace stream (the version at the current update transaction level).

Note: it's sometimes OK for the workspace tree to already contain the new version (the version at the target update level). See *Incomplete Updates* below for an explanation.

If a file about to be overwritten fails this checksum step, AccuRev reports a “crc mismatch”, skips the file, and continues to the next file. The most common cause of this error is your having overwritten the file in such a way that it gets an old timestamp. Such a file escapes detection in Step 1, but gets caught here — just in the nick of time to avoid being clobbered.

If the checksum succeeds, the file is safe to overwrite, so AccuRev updates it — finally! The update can be a replacement of the file's contents, a change in its pathname, or both.

6. If *all* the recently created versions identified in Step 4 were successfully delivered to the workspace:
 - The target update level becomes the workspace's current update level, indicating a successful, complete update.
 - The workspace's update time, used for the **stat -n** timestamp optimization (Step 1), is set to the time of this successful update.

Incomplete Updates

The **Update** command is not implemented as an atomic operation, and it is not recorded as an AccuRev transaction. Transactions are used to organize and serialize changes to the central repository, not to the workspace trees that implement user “sandboxes”. An **Update** can take a significant amount of time, and it can be interrupted before it completes — by user action, by network failure, by loss of telephone connection, etc.

For purposes of discussion, assume that your workspace, **talon_dvt_mary**, has a current update level of 84, that the highest transaction in the repository is 109, and that 45 files in your workspace would be involved in a complete **Update**. You can monitor transaction levels using the CLI command **show wspaces**:

- Before the update the output of **show wspaces** might include:

```
talon_dvt_mary      c:\wks\talon_dvt      xlnr      13 84 84 1 0
```

The first **84** indicates the workspace's target update level; the second **84** indicates the current update level.

- The **Update** command sets the target update level to **109**, then proceeds. If **Update** processes all files successfully, it raises the current update level to the target:

```
talon_dvt_mary      c:\wks\talon_dvt      xlnr      13 109 109 1 0
```

But if the **Update** does not complete successfully, the target update level remains unchanged. A subsequent **show wspaces** reveals that the target update level differs from current update level:

```
talon_dvt_mary      c:\wks\talon_dvt      xlnr      13 109 84 1 0
```


The differing update levels — target vs. current — is the telltale sign that the most recent **Update** did not complete successfully.

Incomplete Update: Command Interrupted

Consider the case in which **Update** was interrupted — say, after it had processed 29 out of the 45 files to be updated. When another **Update** command is issued:

- The checksum of the 29 files will succeed, because those files are already at the target transaction level.
- The checksum of the 16 files will succeed, because those files are still at the current update transaction level.

(See Step 5 above for a discussion of the checksum process.)

Incomplete Update: Checksum Failure

Now consider the case of an incomplete **Update**, due to one or more “crc mismatch” errors. Suppose that only 42 out of 45 files are updated, because 3 files fail the checksum match. You must fix the problem before issuing another **Update**.

If those three files had been overwritten by mistake, you can restore the proper versions using the **Revert to Backed Version** command (CLI: **purge**). Then, a second **Update** brings the new versions of those 3 files into the workspace.

Performing the “Fixup” Update

When it begins executing an Update command, AccuRev determines whether the preceding update of the workspace completed successfully or not:

- The Update completed successfully if the target and current update levels are the same.
- The Update was incomplete if the target and current update levels differ.

If the preceding update was incomplete, AccuRev performs two updates at once. First, it performs a “fixup” update that completes the preceding update; then it performs an additional update (if necessary), to process changes made to the backing stream after the incomplete update.

Example:

- Your workspace’s current update level is 84, and the highest transaction in the repository is 109.
- You issue an Update, but it fails to complete. At this point, the workspace’s target update level is 109, but its current update level is still 84.
- You wait until after lunch break to reissue the Update command. At this point, the highest transaction in the repository is 137.
- AccuRev performs a “fixup” update, which brings the current update level to the original target update level, 109. Then, it advances the target update level to 137 and performs another update. If this update succeeds, it advances the current update level to 137.

Using a Trigger to Maintain a Reference Tree

Reference trees allow you to have a physical copy of the most recent sources for a stream. They are available for reference, thus the name reference tree. Snapshots never change, so they only need to be updated once using **update -r** and then you can forget about them.

To create a reference tree, use the **mkref** command. To keep a reference tree up to date with its associated stream, you need to run **update** on the reference tree every time versions are promoted to the stream.

AccuRev supplies the following trigger scripts to automate this procedure:

server_post_promote.pl

A general-purpose script, which can be used to perform various tasks after completion of every **promote** command. In this case, we're going to have it call the **update_ref.pl** script.

update_ref.pl

A script that invokes the **update** command to update the files in a reference tree. On a Unix machine, this script must be setUID-root.

The indirection is necessary for security purposes.

To enable the automatic updating of one or more reference trees, follow these steps:

1. Make sure the following Perl scripts are installed in some directory on the search path of the AccuRev Server process's user identity:

```
server_post_promote.pl
update_ref.pl
```

See *User Identity of the Server Process* on page 9 of the *AccuRev Administrator's Guide*.

2. Edit both the **server_post_promote.pl** and **update_ref.pl** scripts, and follow the step-by-step instructions contained within them.
3. Windows only: convert the Perl scripts to Windows batch files:

```
pl2bat server_post_promote.pl
pl2bat update_ref.pl
```

4. Tell AccuRev to run the **server_post_promote** script after every **promote** command:

```
accurev mktrig server-post-promote-trig server_post_promote.pl (Unix)
accurev mktrig server-post-promote-trig server_post_promote (Windows)
```

For more information, see the descriptions of **mkref**, **mktrig**, and **show triggers** commands.

Dispatch Command-Line Interface

This note describes aspects of the command-line interface to the Dispatch issue management system.

Note: this interface is not an officially-supported part of the AccuRev software. The information herein is accurate as of Version 3.5, but this interface might change or be discontinued in a future release.

Overview

The Dispatch CLI is implemented through a single command, **accurev xml**. The **xml** command is a non-interactive general-purpose command dispatcher; it reads a specified file to determine the AccuRev operation — in this case, a Dispatch command — to be invoked. For example:

```
accurev xml -l mycmd.xml           ("dash-ell" not "dash-one")
```

Here, the **xml** command's input comes from a file, **mycmd.xml**, which must contain an XML document. (The filename is irrelevant, and need not have a **.xml** suffix.) The XML document might contain this specification of a Dispatch query:

```
<queryIssue
  issueDB="UserReportedBugs"
  useAlt="false"
  expandUsers="true">
21 == "rel2.0"
</queryIssue>
```

This example specifies the command, “find all issue records in depot **UserReportedBugs** whose value in field #21 (the **targetRelease** field) is the string **rel2.0**”. The results of an **xml** command are sent to standard output, also in the form of an XML document. For example, this query might retrieve two issue records, producing this output:

```
<issues>
  <issue>
    <issueNum
      fid="1">2</issueNum>
    <transNum
      fid="2">3</transNum>
    <targetRelease
      fid="21">rel2.0</targetRelease>
    <type
      fid="7">defect</type>

    ... additional fields ...

  <platform
    fid="12">All</platform>
```

```

</issue>
<issue>
  <issueNum
    fid="1">3</issueNum>
  <transNum
    fid="2">26</transNum>
  <targetRelease
    fid="21">rel2.0</targetRelease>
  <type
    fid="7">enhancement</type>

  ... additional fields ...

  <platform
    fid="12">Linux</platform>
</issue>
</issues>

```

This output provides the correspondence between field-ID numbers (e.g. `fid="21"`) and field-names (e.g. `targetRelease`). This correspondence is important, since you must specify a query using field-IDs, not field-names (e.g. `21 == "rel2.0"`, not `targetRelease == "rel2.0"`).

Dispatch CLI Operations

You can perform the following Dispatch operations through the command-line interface:

- Query an issues database (`<queryIssue>` document) — Retrieve the contents of all issue records in a particular depot (issues database) that match a specified query.
- Create a new issue record (`<newIssue>` document) — Enter a single new issue record in a particular depot.
- Modify an existing issue record (`<modifyIssue>` document) — Change the contents of a single issue record that already exists in a particular depot.

The sections below provide guidelines for performing each of these operations. But there's an important prerequisite step to perform first

Determining the Field-ID / Field-Name Correspondence

In the Dispatch CLI, you identify a field by its field-ID, not by its field-name. Thus, before doing any real Dispatch CLI work, you must determine the correspondence between field-IDs and field-names in your depot (issues database). This information is stored in the schema configuration file on the AccuRev Server host:

```
<AccuRev-inst-dir>/storage/depots/<depot-name>/dispatch/config/schema.xml
```

You can retrieve the contents of the **schema.xml** file from an AccuRev client machine with this command:

```
accurev getconfig -p <depot-name> -r schema.xml
```

Extract the **name=** and **fid=** text lines from this data, and store them for future reference. You'll need to refer to this information often as you work with the Dispatch CLI. Let's call this extracted data the field-ID definitions.

For example, the field-ID definitions for the default schema look like this:

```
name="issueNum"
fid="1">
name="transNum"
fid="2">
name="status"
fid="3">
name="shortDescription"
fid="4">
name="state"
fid="5">
name="productType"
fid="6">
name="type"
fid="7">
name="severity"
fid="8">
name="priority"
fid="9">
name="submittedBy"
fid="10">
name="dateSubmitted"
fid="11">
name="platform"
fid="12">
...
```

This data shows that field **submittedBy** has field-ID **10**, field **productType** has field-ID **6**, etc.

Note: all examples in the remainder of this document will use the field-ID/field-name correspondence in the above example.

The contents of these XML elements are the field values for issue record #1. A couple of them, **submittedBy** and **dateSubmitted**, have values that might be a bit surprising — numbers instead of strings. We'll discuss these kinds of values in section *Selecting Issue Records with a Query* below.

Selecting Issue Records with a Query

The simplest kind of query selects one or more issue records by comparing the value of one field with a literal value (a constant) — for example, “is the value of the issueNum field equal to 415?” or “does the shortDescription field contain the string ‘busted’?”.

For such simple queries, you can adapt the one we used in section *Determining the Field-ID / Field-Name Correspondence* above. This query finds all issue records whose **productType** value is **Frammis**.

```
<queryIssue issueDB="XXXXX" useAlt="false">
  6 == "Frammis"
</queryIssue>
```

The output of a query is an XML document whose top-level <issues> element contains zero or more <issue> sub-elements:

```
>>> accurev xml -l query-filename
<issues>
  <issue>
    ... individual field-name elements ...
  </issue>
  <issue>
    ... individual field-name elements ...
  </issue>
  ...
</issues>
```

More Complex Queries

You can compose and run arbitrarily complex queries. If the query goes just a bit beyond the simplest, you can probably compose it manually. For example, this query finds all issue records whose **productType** value is **Frammis** or **Widget**:

```
<queryIssue issueDB = "dpt38" useAlt = "false" useAltQuery = "false">
  <OR>
    <condition>6 == "Frammis"</condition>
    <condition>6 == "Widget"</condition>
  </OR>
</queryIssue>
```

With more complex queries, be sure to include the useAltQuery = "false" attribute in the <queryIssue> start-tag.

But to minimize the chance of getting lost in the syntax of more complex queries, we strongly recommend that you compose the complex query graphically, then “export” the query to a text file:

1. In the AccuRev GUI, enter the command **Dispatch > Queries** to enter the Query Editor.

2. Create a new query, give it a name, and specify the query logic.
3. Click the **Save All Queries** button.
4. Place an XML-format dump of *all* your queries in a text file:

```
accurev getconfig -p depot-name -u user-name -r query.xml > myqueries.txt
```

This **getconfig** command retrieves the contents of the configuration file that stores your queries:

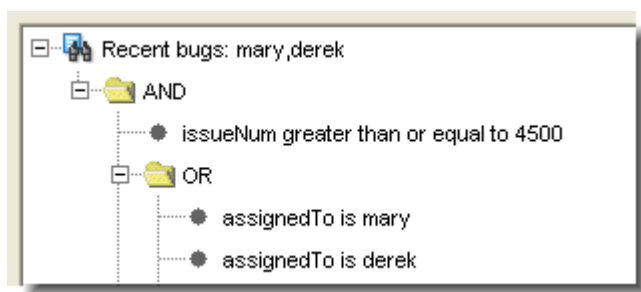
AccuRev-install-dir/storage/depots/depot-name/dispatch/config/user/user-name/query.xml

5. Use a text editor to extract the XML `<queryIssue>` element that defines the query. (Don't extract the entire `<query>` element, which includes the query's name and output field definitions.)
6. Store the `<queryIssue>` element code in a separate file, say **frammis_or_widget.xml**.

You can now invoke the query with the CLI:

```
accurev xml -l frammis_or_widget.xml
```

Example: This GUI-composed query ...



... is represented by this XML code:

```
<queryIssue issueDB = "dpt38" useAlt = "false" useAltQuery = "false">
<AND>

<condition>
1 >= &quot;4500&quot;;

</condition>
<OR>

<condition>
14 == &quot;2&quot;;

</condition>

<condition>
14 == &quot;24&quot;;
```

```

</condition>
</OR>
</AND>
</queryIssue>

```

This code is perfectly good XML, even though it's not "pretty-printed". Also note the use of the XML entity reference **"**; which is equivalent to a double-quote character.

Special Field Types

Each field in a Dispatch issues database has a field type: Text, Choose, List, etc. For a field of type User (such as the **submittedBy** field in the example in section *Determining the Field-ID / Field-Name Correspondence* above), a query defaults to reporting users by their integer user-IDs, rather than by their usernames (principal-names):

```

...
<submittedBy
  fid="10">40</submittedBy>
...

```

In this case, you could use the output of the **accurev show users** command to determine that user **jjp** has user-ID **40**. Alternatively, you can modify the query to set the attribute `expandUsers` in the `<queryIssue>` start-tag:

```

<queryIssue issueDB = "dpt38"
  useAlt = "false"
  useAltQuery = "false"
  expandUsers = "true">
...

```

Setting `expandUsers` causes the query to report the values of User fields with usernames instead of user-IDs:

```

...
<submittedBy
  fid="10">jjp</submittedBy>
...

```

For a field of type Timestamp (such as the **dateSubmitted** field in the example in section *Determining the Field-ID / Field-Name Correspondence* above), a query always reports the timestamp as a large integer, representing the number of seconds since Jan 1, 1970 UTC:

```

...
<dateSubmitted
  fid="11">1083606273</dateSubmitted>
...

```

(This is the standard Unix timestamp scheme.) You can use Perl to convert the integer into a human-readable string:

```
>>> perl -e "print scalar localtime(1083606273)"
Mon May  3 13:44:33 2004
```

Creating a New Issue Record

To create a new issue record in a particular issues database, execute the command

```
accurev xml -l my_datafile
```

... where **my_datafile** contains an XML document in this format:

```
<newIssue issueDB="XXXXX">
  <issue>
    ... individual field-value specifications ...
  </issue>
</newIssue>
```

As always, replace **XXXXX** with the name of the depot that stores the issues database. For the individual field-value specifications, adapt the output of a query that retrieves a single issue record. The complete contents of **my_datafile** might be:

```
<newIssue issueDB="UserReportedBugs">
  <issue>
    <type
      fid="7">defect</type>
    <submittedBy
      fid="10">5</submittedBy>
    <foundInRelease
      fid="20">rel2.0</foundInRelease>
    <productType
      fid="6">Widget</productType>
    <shortDescription
      fid="4">Names are sometimes trunca</shortDescription>
    <dateSubmitted
      fid="11">1083606275</dateSubmitted>  </issue>
  </newIssue>
```

Some DOs and DON'Ts:

- You must specify the value of a User field with a user-ID (**5**), not a username (**derek**).
- You must specify the value of a Timestamp field with a number-of-seconds integer, not a string. You can use the **timelocal()** function in the Perl module **Time::Local** to generate these integers:

```
use Time::Local;
$sec = 35;    # range = 0 .. 59
$min = 22;    # range = 0 .. 59
```

```
$hr = 14; # range = 0 .. 23
$dte = 8; # range = 1 .. 31
$moth = 5; # range = 0 .. 11 (January is the zero'th month!)
$yr = 2004; # play it safe: use a 4-digit number
$numseconds = timelocal($sec, $min, $hr, $dte, $moth, $yr);
print $numseconds, "\n";
```

- Don't include specifications for the **issueNum** and **transNum** fields. Dispatch assigns these values automatically.
- The field initialization and validation code defined in the issues database schema will not be executed when the issue record is created. It's up to you to specify the appropriate values for the appropriate fields.

The validations *will* be invoked when the issue record is subsequently opened in the Dispatch GUI. In particular, you can create an issue record with a List field whose value is not in the field's list of possible values. But when the Dispatch GUI opens the issue record, it replaces the bogus value with `<none selected>`.

- Be sure that the top-level and second-level XML elements are named `<newIssue>` and `<issue>`. The names for the individual-field elements are irrelevant — only the **fid** attributes count. The following specifications are equivalent:

```
<status fid="20">New</status>
<myfield fid="20">rel2.0</myfield>
```

When you submit the `<newIssue>` data structure to the Dispatch CLI, it creates the record and reports the new record's contents. This report includes the automatically assigned **issueNum** and **transNum** values:

```
>>> accurev xml -l my_datafile
<issue>
  <issueNum
    fid="1">11</issueNum>
  <transNum
    fid="2">154</transNum>
  <type
    fid="7">defect</type>
  <submittedBy
    fid="10">24</submittedBy>
  <foundInRelease
    fid="20">rel2.0</foundInRelease>
  <productType
    fid="6">Frammis</productType>
  <shortDescription
    fid="4">Refuses to fram</shortDescription>
  <dateSubmitted
    fid="11">1062787292</dateSubmitted>
</issue>
```

See also *Using 'modifyIssue' to Create New Issue Records* on page 71 below.

Modifying an Existing Issue Record

Use the following procedure to modify an existing issue record:

1. Create a query to select the desired issue record, as described in *Selecting Issue Records with a Query* on page 66. For example, to select issue record #472 from the **Problems** issues database, create this XML document:

```
<queryIssue issueDB="Problems" useAlt="false">
  1 == "472"
</queryIssue>
```

2. Run the query, storing the results in a text file:

```
accurev xml -l myquery.xml > issue472.xml
```

3. Edit the text file, making these changes:

- Change the top-level XML **<issues>** start-tag to **<modifyIssue>**. Include **issueDB** and **useAlt** attributes in the start-tag:

```
<modifyIssue issueDB = "Problems" useAlt="false">
```

- At the end of the file, change the **</issues>** end-tag to **</modifyIssue>**.
- Remove the entire **<transNum>** element.
- Change the values of one or more existing fields.
- If you wish, add new field values, using the discussion in *Creating a New Issue Record* on page 69 as a guide.

4. Submit the edited text file:

```
accurev xml -l issue472.xml
```

When you submit the **<modifyIssue>** data structure to the Dispatch CLI, it modifies the specified issue record. As when you create a new issue record with the Dispatch CLI, field validations are not applied.

To verify the new record's contents, submit the original query again:

```
accurev xml -l myquery.xml
```

Using 'modifyIssue' to Create New Issue Records

Instead of using a **<modifyIssue>** document to change an existing issue record, you can use it to create a new issue record. This is useful for copying issue records from one issues database to another. For example, you might use a **<queryIssue>** document, as described earlier, to retrieve the contents of an issue record from database **Problems**, in the form of an **<issues>** document. As

described in Step 3 above, when you change the `<issues>` tag to a `<modifyIssue>` tag, specify a different database:

```
<modifyIssue issueDB = "Problems_Public" useAlt="false">
```

In this example, the issue record will effectively be copied from the **Problems** issues database to the **Problems_Public** issues database. Make sure the target database exist and has the same schema as the source database.

The `<modifyIssue>` technique is also useful for making copies of the issue records that act as change packages, and for replicating issue records between different AccuRev sites.

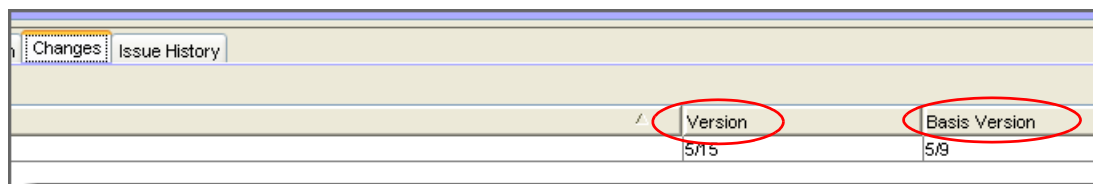
Note that a new issue record created with `<newIssue>` always gets assigned the next available issue number in the database; by contrast, a new issue record created with `<modifyIssue>` gets the issue number specified by the `<issueNum fid="1">` subelement:

```
<issueNum  
  fid="1">4197</issueNum>
```

An issue record with this number must not already exist in the target database, but there is no other restriction. It's perfectly OK to have a “sparse” issues database, in which most issue numbers are unallocated.

Interface to the Change Package Facility

Dispatch issue records are used to implement the change package facility, introduced in AccuRev 3.5 Enterprise Edition. The set of changes in a change package is indicated by a set of “Versions”, listed on the Changes subtab of an issue record, each with a corresponding “Basis Version”:



Version	Basis Version
5/15	5/9

Each Version / Basis Version pair defines a set of changes to the element: the changes made since the Basis Version was created, up to and including the Version. The change package consists of such Version / Basis Version “change intervals” for any number of elements.

Various user commands and triggers maintain the contents of a change package: adding new versions and removing existing versions. There are also commands for comparing the contents of a change package to the contents of a stream, enabling you to easily answer the question, “Have all the changes made for Task A been propagated to Stream B?”

The following sections describe the CLI to the change package facility.

Adding Entries to a Change Package

The following XML document requests the adding of two entries to the change package of issue record #433: for the element with element-ID 3, record the series of versions between Basis

Version 4/3 and Version 4/6; for the element with element-ID 7, record the series of versions between Basis Version 4/2 and Version 4/9.

```
<acRequest>
  <cpkadd>
    <user>jjp</user>
    <depot>etna</depot>
    <stream1>etna_dvt</stream1>
    <issues>
      <issue>
        <issueNum>433</issueNum>
        <elements>
          <element
            id="3"
            real_version="4/6" basis_version="4/3">
          <element
            id="7"
            real_version="4/9" basis_version="4/2">
          </element>
        </elements>
      </issue>
    </issues>
  </cpkadd>
</acRequest>
```

Removing Entries from a Change Package

The following XML document requests the removal of the change-package entry for the element with element-ID 3 from issue record #433.

```
<acRequest>
  <cpkremove>
    <user>jjp</user>
    <depot>etna</depot>
    <stream1>etna_dvt</stream1>
    <issues>
      <issue>
        <issueNum>433</issueNum>
        <elements>
          <element
            id="3"
            real_version="4/6">
          </element>
        </elements>
      </issue>
    </issues>
  </cpkremove>
</acRequest>
```

```
</cpkremove>  
</acRequest>
```

Listing the Contents of a Change Package

The following XML document requests the listing of the change packages of issue records #433 and #512.

```
<acRequest>  
  <cpkdescribe>  
    <user>jjp</user>  
    <depot>etna</depot>  
    <stream1>etna_dvt</stream1>  
    <issues>  
      <issueNum>433</issueNum>  
      <issueNum>512</issueNum>  
    </issues>  
  </cpkdescribe>  
</acRequest>
```