

# User Management with Perforce

Tommy Fad  
National Instruments  
May 2003

## Contents

### Introduction

#### What should I manage?

- User data

- Group data

- Protections data

#### Where should I manage this data?

#### Migrating data to Perforce

- Verify that data should be updated

- Migrating user data

- Migrating group data

- Migrating protections data

#### Migrating data carefully

#### What did I gain?

## Introduction

Managing a large group of users for any system can be time consuming, confusing, and costly. A well-defined process for managing the system and its users' requests, however, can automate otherwise repetitive tasks. Groups, users, and permissions are three elements of Perforce that can change on a regular basis. As the number of users increases, so do the number of requests to make changes to these elements. Since Perforce also allows multiple servers with a common user base, the amount of time spent managing these changes can double for each additional server. This paper will explain how a User Management System can help reduce the amount of time spent administering changes in Perforce users, groups and permissions.

## What should I manage?

While all of Perforce's system information can be managed in some manner, users, groups and permissions are the most commonly modified pieces. Typical user management requests for a Perforce Administrator include adding a user to a group, giving a user access to a depot, adding new users and removing old users. In addition, accounting departments may be concerned with reporting how many licenses are being

used by a particular cost center or when the next payment to Perforce for support fees is due. These are concerns that a Perforce Administrator probably would not have. Instituting a system that manages the data for both the Perforce Administrator and the accounting department, while providing a common user interface, can benefit both groups.

## User Data

User data should include the same information contained in the Perforce application, as well as any additional details pertaining to the user, such as cost center, their Perforce group(s), contact information or department. Storing information that would benefit both the accounting department and the Perforce Administrators can be worthwhile. Reports or views can be created using this data, ensuring that various departments are provided with only the information they need. Below is an example of user data that might be contained in the User Management System:

<b>UserName</b>	Jdoe
<b>Name</b>	John Doe
<b>Email</b>	<a href="mailto:John.Doe@mydomain.com">John.Doe@mydomain.com</a>
<b>PerforceGroups</b>	Group1, Group2, Group3
<b>Servers</b>	Perforce1.mydomain.com, Perforce2.mydomain.com
<b>CostCenter</b>	123
<b>Extension</b>	1234
<b>Department</b>	Software
<b>Comments</b>	Forgets that Depot //Foo is on Perforce2.mydomain.com
<b>ActiveUser</b>	Y

## Group Data

Group data consists of the group name, members of the group, max results, max scan rows and subgroups. Optional information stored in group data includes which depot(s) the group has access to and which server the group will be added into. Below is an example of group data that might be contained in the User Management System:

<b>Group</b>	Group1
<b>MaxResults</b>	Fill in later
<b>MaxScanRows</b>	Fill in later
<b>Users</b>	tommyf sevel johnd
<b>DepotAccess</b>	//FooDepot/...
<b>Server</b>	Perforce1.mydomain.com
<b>ActiveGroup</b>	Y

## Protections Data

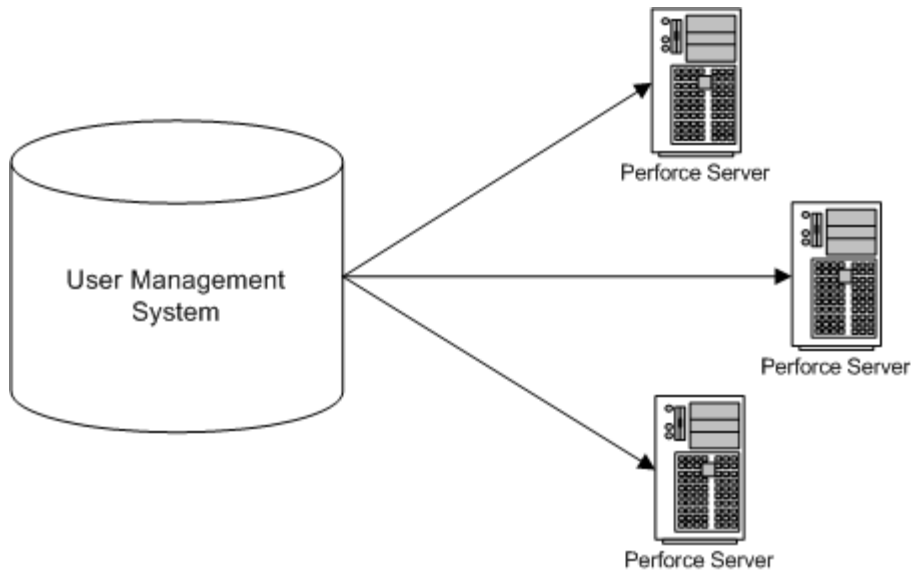
Protections data is sensitive information because it gives users access to specific areas of Perforce. Some databases allow administrators to restrict who can view or modify this information. However, many do not, which might create concerns about storing protections data somewhere other than in Perforce. One benefit to keeping the information in a User Management System is that it can be managed in a central location. When multiple IP addresses are in the Protect file, determining which systems have access to Perforce can be difficult. Storing protections data in the User Management System allows administrators to comment on the Protect file. Below is an example of protections data that might be contained in the User Management System:

```
Protection write group WebGroup 123.456.789.123 //WebGroupDepot/...  
Active Y  
Comments This grants write permission for the WebGroup to their depot  
BelowLine Read group AllUsers 123.456.789.123 //WebGroupDepot/...
```

The 'BelowLine' field is specified because Perforce reads Protect file from top to bottom. Therefore, if the permissions data must follow a certain sequence in order to be written to the Protect file correctly, that sequence must be specified. An example might be that an administrator wants to grant write access to all groups except one. It is easier to grant write access to a group called AllUsers, and then remove write permissions for the single group, than it is to place a 'write' entry to grant each group write access.

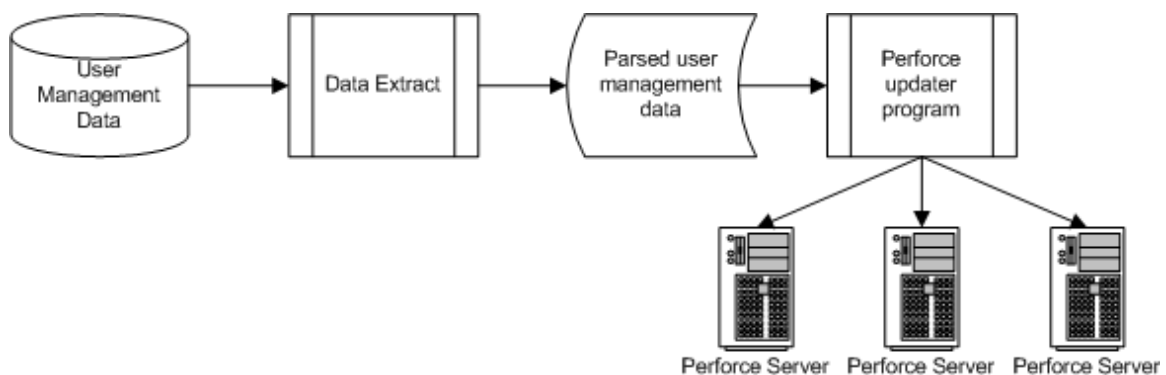
## Where should I manage this data?

Data can be stored using many methods; however, a relational database system is probably the most expandable. When choosing a method for storing the data, retrieving the data should also be considered. There are a number of ways to retrieve data from current database systems. Technologies such as JDBC or PERL modules allow users to connect to any number of common databases. As with any database system, analysis of what data to store can dictate table structures and other architecture. The server that the central User Management System resides on allows replication of information out to multiple Perforce servers. The below diagram depicts how the data would flow from the User Management System to Perforce:



## Migrating data to Perforce

Once the User Management System has all of the necessary data, a method to update Perforce with the information will be needed. As stated above, the data can be extracted many ways. After the data has been extracted from the user management system, it can either be written to a flat file or it can be directly inserted into Perforce. The benefit to using a flat file is that the signature will be checked before the update process begins. If the signature has not changed, then there is no need to update Perforce with any information (since none of the information is new). For this example, it is assumed that data is extracted to a flat file every hour. Below is a more detailed view of how the data will flow from the User Management System to the Perforce server(s):



## Verify that data should be updated

After a file format for the export database information has been chosen, the first part of the Program/Script should verify that the file format is correct. There are unlimited ways to check for a consistent format; however the amount of error checking is up to the individual. After verifying that the file format is correct, ensure that the signature of the file has changed (for instance, a new user could have been added so the file signature

would change). The following example illustrates how to check this using the perl SHA module. The first step is to read in the old checksum:

```
my $sha = new SHA;
$sha->reset();

open CHECKSUMLOG, "<$pathtochecksumFile";
my $oldDigest = <CHECKSUMLOG>;
close CHECKSUMLOG;
```

Next, read the data file to compare its checksum to the previous checksum:

```
open DATAFILE, "<$pathtodataFile" or dieOnErr "Can't open
data file $!\n";
while (<DATAFILE>)
{
    $sha->add($_);
}
close DATAFILE;

my $digest = $sha->hexdigest();
if ($digest eq $oldDigest)
{
    exit;
}
```

## **Migrating user data**

Assuming that the checksums do not match, the data in the User Management System must now be compared to what is already in Perforce. Make two hashes for the user information and the group information. These hashes should come from the formatted file that was extracted from the User Management System. The manner in which the user data file was formatted determines how the information can be stored in the hashes (or data structures). Next, create a ‘users hash’ from Perforce. The hash should contain all the necessary information, plus a list of the clients owned, a list of files the users have opened and a Boolean to specify whether or not those files are on a client owned by the user. The commands to generate the Perforce information listed above are described below (parsing methods are chosen by the individual):

Perforce command to display users	p4 users
Perforce command to display groups	p4 groups
Perforce command to display clients	p4 clients
Perforce command to display opened files	p4 opened

Now that the necessary hashes are populated, users marked as ‘not active’ can be removed (see ‘Group Data’ above). All the information needed to delete users and

clients should be in the user hash. Below is a list of the information to check for before deleting users and clients. Following these rules can help error-checking go smoothly.

**Rule1:**

Do not delete userA, who owns clientA, when userB has files opened using clientA. These open files can be listed and an email alert can be sent to the appropriate parties.

**Rule2:**

Do not delete userA, who is using clientB, which is not owned by userA, if any other user has files opened using clientB. Here again the open files can be listed and the appropriate parties can be alerted.

Finally, the user and client can be deleted using the following commands:

```
Perforce command to forcefully delete user  p4 user -fd <user>
Perforce command to forcefully delete client p4 client -fd <client>
```

Before simply wiping out a user or a client it can be helpful to create a text file of the deleted client. Therefore, if the user was accidentally deleted and had his/her clients deleted as well, it will be easy to recover and put their clients back to their original state. When replicating data to multiple servers, a folder for each server can be created (since clients can have the same name for different servers). A simple way to do this is to redirect a command, which specifies the client settings to a text file:

```
Perforce command to write client  p4 client -o <client> > client.txt
data (standard out and redirect)
```

Now that all ‘inactive users’ have been deleted, find out if there are any existing users who have modified any of their data in the User Management System. There are a number of ways to figure out if any of the data shared between the User Management System and Perforce (username, full name, email address, etc.) have changed. One way is to create a hash within a hash; another is to use a regular expression to match the data from the hash to the data from the `p4 user` command. To get the user’s information from Perforce, type the following command:

```
Perforce command to write user data (standard out)  p4 user -o <user>
```

Once all of the data for existing users has been updated, new users can now be added to Perforce. The final step in migrating user data to Perforce is to check for any users defined in the hash containing `p4 users` output as any not listed in the output file from the database management system. If there are any users, they can be deleted using the methods described above, or by sending an email alert.

## **Migrating group data**

Just as in migrating user data, a hash of group data should be created. This hash should contain all of the users in a specific group. Once again, determine the format in which to store data in the hash or data structure. Some parsing methods may be more useful than others. First, find any of the existing groups and add remove users as they are specified in the output file from the User Management System. While adding/removing existing members to existing groups, check for users that may not have been defined in the output log from the User Management System and delete them from the group accordingly. Upon arriving at a new group specified in the User Management System's output file, add that group and its appropriate members. Any other specifications (subgroups, maxscanrows, etc.) should be added to this new group as well. If the User Management System specifies a group as 'inactive,' remove that group or notify those users that the group will be removed. As with undefined users, either delete or send an email notification for undefined groups in the User Management System's output file.

## **Migrating protections data**

After all of the user and group data has been migrated, it is time to move the protections data. The protections data should be fairly simple to update. First, get all of lines of the protections file into a hash or another data structure. This can be done using the following Perforce command (remember to parse the output):

Perforce command to display protect file (standard output) `p4 -o protect`

The next step is to remove any protect lines that have been marked as inactive or are not listed in the output file from the User Management System. After that, find any lines that were listed as 'BelowLine' (see Protections Data), and place the active protect line below this line. Using the following Perforce command will allow data to be pushed back into the protect file:

Perforce command to direct standard input to the protect file `p4 -i protect`

## **Migrating data carefully**

There are many ways to parse data from a User Management System and update Perforce with this information. It is important to perform **error checking** during migration. Since this system is automated, various scenarios should be tested to ensure that the migration is error-free. Allow for several use cases while testing. It is also essential to note **efficiency**. This whole system boils down to building a good diff utility. Make sure that the appropriate data structures, search routines and sort algorithms are used. For example, in Perl, arrays are indexed by integers, while hashes are indexed by strings. It is much faster to find something that has a key in a hash rather than searching through a whole array. Finally use **email alerts**. The system does not do any good unless a user is notified of errors, updates, or simply things that should not be happening. Email alerts are the key to monitoring a User Management System.

## **What did I gain?**

After all this time has been spent creating a User Management System, what was gained? Perforce administrators saved time and reduced their workload. Keeping all of the Perforce data in a central location also helps increase security by allowing for different levels of user access. Maintaining information in a central system also enables monitoring of not only local servers, but also of servers halfway around the world. A centralized User Management System also provides an interface between user accounts for non-Perforce Administrators.

The more data stored in the User Management System, the greater the number of queries and reports that can be run against the system. The licensing department can determine which groups or cost centers own specific licenses, as well as when it is time to allocate more money to buy extra licenses. Having an automated User Management System can save time, increase security, and provide a friendly user interface for companies with a large user base.