realtimepublishers.com™

The Administrator Shortcut Guide™ To

User Management and Provisioning

abridean

Dave Kearns

Copyright Statement

Chapter 2: User Management

Many of you might secretly hope that a chapter about user management will include subheadings such as “Cattle Prods” and “Holding Pens.” In fact, user management really isn’t about users very much at all. Rather, it’s about the accounts of users on enterprise networks. For example, a discussion about a user’s authorizations—the rights and privileges that the user has—is actually an exploration of the rights controlled by a particular account. Should the physical user log on to a different account, it’s likely that the authorizations—the access privileges—would be different. Thus, although user management isn’t a completely accurate term, it is embedded in the Identity Management discussion, so we will use throughout this guide. Simply remember that it isn’t the physical users we’re talking about, but their digital presence on the network.

A Working Definition

User management is sometimes called user lifecycle management, which is sometimes shortened to lifecycle management. It is the existence of the user account on the network that is being defined. There are five aspects to this process:

- Creating an account for a user
- Giving the account the right access roles
- Replacing a forgotten password
- Moving a user from one location to another
- Retiring the account when it is no longer required

In this chapter, we’ll take a look at each aspect in turn, then review the factors presented in Chapter 1 for a successful user management and provisioning project. These considerations should be followed to plan and implement a successful user management project.

For more information about considerations for a successful user management and provisioning project, see Chapter 1.

Creating an Account

You might expect that the process of creating an account for a user is governed by the parameters of the service within which you are creating the account than by any general or "best practice" issues. However, although you cannot perform tasks that are forbidden by the application or service's rules, there is still a great deal of leeway within which you can work. Two major areas of flexibility are the formation of a username for the account and the rules for passwords.

Usernames

Almost all databases are indexed (that is, sorted) according to a field called the primary key, which needs to contain a value unique within that database. Many databases holding identity information, such as network directories like AD (Windows servers) or eDirectory (for example, NetWare servers), require usernames to be unique either for the entire directory or for the namespace within the directory (such as a domain, container, or organizational unit—OU) in which that user's data will be stored. You might think, then, that the username is also the primary key. In fact, the primary key is usually a number, which no one ever sees, that can be called the serial number, the User Identification (UID) number, or some other indicator. A unique name is required to avoid confusion—although it serves a useful purpose when the username is also the email name, which is required to be unique.

To make the account creation task easier, adopt a template for usernames. Two widely used templates are first initial plus last name (for example, John Doe would be known as jdoe) or first name/last name with a separator such as a dot or a dash (for example, John Doe would become john.doe or john-doe).

15 or 20 years ago, the custom was to use only the first seven letters of the last name along with the first initial (rendering Ruth Bresnahan as rbresnah) so that a home directory—what we now call a folder—could be created using the username while holding to the DOS limit of eight characters for a file or folder name. Such is no longer a requirement with modern Microsoft OSs and was never required for UNIX, Linux, and Macintosh systems. Still, some organizations cling to this template. If at all possible, create a template that employs all the letters of a person's name in the username. Doing so will speed searches and be easier for people to remember.

Collisions are caused when a template creates the same username for two or more people. John Smith, Jane Smith, and Joe Smith would all become jsmith in a first initial plus last name scenario. Assume that there will, at some point, be a collision, and that as the organization grows, the number of collisions will increase dramatically. Plan for this scenario by developing a secondary template to differentiate users. Perhaps Jane Smith become jsmith1 and Joe Smith become jsmith2. Alternatively, you might use middle initials (for example, jsmith, jasmith, jrsmith) or even entire middle names. Consider ease of use for the users involved but also minimize the confusion for others within and outside of the organization who might be attempting to find an email address.

☞ Modern network directory systems and email directories, also known as address books, often allow for a user's whole name and other information (for example, "Jane Smith [acctng]") to be shown and searched. This type of *display name*, which is an alias to the username, make life easier for everyone.

Passwords

Passwords should be secret and should be known only to the user. However, forgotten passwords trigger more Help desk calls than any other single problem. Nevertheless, avoid the urge to maintain a list of passwords that would enable you to quickly look up a user's forgotten password. Many of the authentication systems in use today—both those included with servers and applications as well as third-party add-on products—allow you to specify rules for passwords that make the passwords hardened against attacks by intruders. Rules can be designed to cover some or all of these situations: length, character set, case, commonality, reuse, and familiar terms.

Password Length

On one hand, the longer the password, the more difficult it is for an intruder to guess or otherwise discover the password. On the other hand, the longer the password, the easier it is for the user to mistype or forget it. To balance these needs, require a minimum length of 6 to 12 characters. Sites with easy public access or with highly sensitive information should use longer passwords, of course.

Character Set

Almost all authentication systems allow the use of the 26 letters of the English alphabet. Most also allow the digits 0 through 9. Some allow certain punctuation or special marks such as dashes, underscores, addition signs, and exclamation points. Generally, the more characters allowed, the stronger the password will be. "Password" is an easily guessed password. But if you use a substitution method, such as the example method that Table 2.1 shows, to change a few characters, "password" becomes p@s5w0rd, which is much more difficult to guess.

Letter	**Substitute**
A	@
B	8
b	6
E	3
g	9
i	!
l	1
o	0
S	5
t	7
y	4
z	%

Table 2.1: Substitute numbers and symbols for letters to help obfuscate passwords.

Case of Characters

Another option is to use a password in mixed case, such as "PasSwORd"; however, many systems ignore case. Given a choice, choose an authentication system (or an add-on) that differentiates between upper and lower case characters in passwords. Doing so immediately doubles the number of characters available for a password. And as we determine earlier, the more characters allowed, the stronger the password will be.

Common Words and Phrases

Add-on security packages for UNIX systems have long had the ability to block users from using common dictionary words (for example, dog, cat, breakfast) as passwords. This feature is gradually becoming available for more systems. You should look for an authentication system that provides this capability. A favorite ploy of intruders attempting to guess passwords is to use what's called a *dictionary attack*, which simply rotates through the words in a standard dictionary attempting to guess passwords. I'm not suggesting you attempt to block the use of the more than 500,000 words in the Oxford English Dictionary, which would, in any case, take far too long to do. However, blocking the 5000 or so most common words would be a big step forward.

☞ If your system allows mixed case as well as an extended character set, common words can be allowed if spelled using mixed case. You could block "cartoon" while allowing "cARtOoN."

Password Changing and Reuse

The longer a password is in use, the easier it becomes to guess. Require that users change their passwords periodically—at least every 90 days in a low-security environment and perhaps weekly in a high-security area. No matter how often they are required to change their passwords, you should also prohibit re-use of a password within a set period of time. Users who must change passwords monthly, for example, should not be allowed to repeat passwords within a 6-month period (or longer). Not all authentication services will allow you to set rules on reuse, but many do and the best of them will provide additional checks to ensure that users aren't subverting the rule (changing "password" to "password1", "password2", and so on).

Familiar Terms

Users feel comfortable choosing easily remembered passwords. If you've blocked common words, users will fall back on names (spouses, kids, pets), words written on their equipment (Dell, Epson, and so on), and similar weak passwords. Although blocking the use of familiar terms is one of the most difficult rules to implement, these passwords offer very fertile ground for intruders. Blocking use of these words requires you to spend time gathering this information, but if effective security is your intent, that research is time well spent. As with previous password rules, you must reach a balance between secure passwords and those that are too complex for users to remember without writing them down on a sticky displayed on their monitors.

Granting Account Access

One of the most security-conscious changes to network accounts over the past 20 years is the move from allowing everything except that which was specifically prohibited to denying everything except what is explicitly permitted. The industry has gone from a situation in which all of the access points were open by default (and stayed that way until administrators remembered to close them) to having most access points closed by default. As Figure 2.1 illustrates, there are three principal ways of assigning access: by individual user, via group assignments, and through role-based access.

> Although discussions generally talk about granting access to the user in terms of the user account, the access is most likely controlled by an access control list (ACL) that is a property of the resource. Granting a user—or a group or a role—access actually means that you're adding the user—or group or role—to the ACL for that resource.

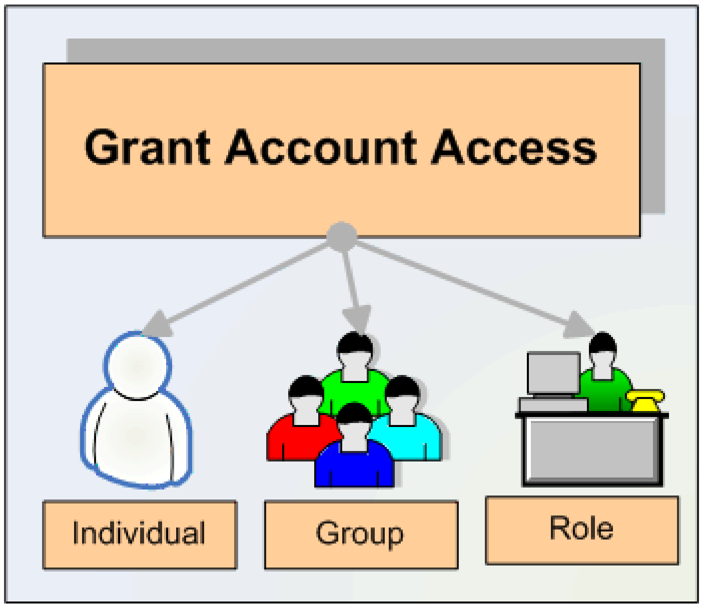

Figure 2.1: Account access can be granted to individual user accounts, based on group membership, or based on organizational role.

Individual Access

Individual access simply grants access on a user-by-user, resource-by-resource basis. It's tedious—just think of all the resources you have access to, then multiply by the total number of users to determine the number of access grants you would have to perform. It's seems even more tedious when compared with group-based and role-based access because any changes—such as the location of a resource—require you to make changes to every user account needing access to that resource. Because the access is actually a property of the resource, you might need to search through all resources to find every instance of that user. Thus, for authentication systems that offer group-based and role-based access controls, always use these methods rather than granting individual access—except for those items (folders, files, personal desktop computers, directly attached printers or scanners, and so on) that are primarily or solely used by the individual user.

Group Access

If your authentication system allows it, assign users to groups based on a shared need to access a particular resource in the same way. A simple example is a departmental printer: create a group that includes everyone in the department, then grant the group access to the printer. Should any changes be needed, such as a new driver, a new location for the printer, or a new model—one change to the group configuration should be all you need to ensure that everyone still has access. Similarly, applications, such as word processors, that are used by a large number of users can be best administered by using groups to grant access. You can then update the application, move it, or even replace it with only a minimal amount of administrative activity. Using groups also eases the administrative burden when adding users to or removing them from your network: simply putting them into—or removing them from—the requisite groups eliminates the need to remember every right and privilege that needs to be granted (or revoked).

A few authentication systems now support the concept of *dynamic groups*. A dynamic group contains those users that have a common value for a particular attribute. For example, everyone who has the value Accounting for the attribute Department is a member of the ACCT-DYN group. As with a static group, a dynamic group can be assigned access rights. The difference between a static and dynamic group is that when a user attempts to access a resource, the authorization system checks the attribute for the required value and only grants access if the proper value is found. Thus, if you change the user's Department attribute to Marketing, and the user will no longer be able to access resources as part of ACCT-DYN.

Role-Based Access

Granting access based on roles is similar to using groups to control authorization; for systems that don't support the concept of roles, groups can serve a similar purpose. The difference between a role and a group is not great. Usually, a group is organized around a particular resource (for example, a word processor group), business organization (the marketing group), or activity (the carpool group), while a role is typically based on job function (buyer, administrative assistant, Help desk technician). You would rarely set up a group for less than three or four users because the overhead of group creation and maintenance wouldn't save you any time or effort. However, a role-based account is useful even if no one is in the role at a given time. For example, you could create a role called VP of Marketing so that you could rapidly move a new user in (or an existing user out) of all of the access the user would need to do the job.

Most authentication systems that understand the concept of "role" allow you to create one using much the same process as creating a user account. You can't put in location data or personal information (birth date, home address, and so on), but you can indicate groups that the role should be a part of and any access rights the role should automatically acquire.

Which Access is Correct?

Different systems (network OSs, desktop OSs, database applications, Web services) offer a range of rights and privileges. Traditional UNIX systems offer only Read, Write and Execute, while Novell's NetWare allows a rich mix including Supervisor, Read, Write, Create, Erase, Modify, File Scan, and Access Control. Also with NetWare, each of these rights differs in effect depending on whether they refer to a file or a folder. The system you're working with may support as few rights as UNIX supports, as many as NetWare, or somewhere in between. It should be self-evident that a user should have full access to his or her data files, which are stored in folders assigned by the administrator to only that user.

Shared folders, however, present a problem. If all users of the shared folder have full access, one user's file will often overwrite another user's file of the same name. Granting, for example, only the Create, File Scan, and Read rights allows each user to read and place files in the shared folder without overwriting another user's files. Unfortunately, this setup doesn't allow the creator of the file (or anyone else without more rights) to modify the contents or remove a file. One solution to this problem is to grant one user Supervisor rights to the folder and have that user perform all maintenance. Another solution is to employ a *versioning* (also called *librarying*) application that allows users to create, register, and store new documents and read existing ones while allowing the document's owner (or creator) to modify or remove it. In effect, the versioning package acts as the Supervisory user.

Applications deserve their own consideration when granting user rights. After all, applications are only files that have a special purpose stored on the system. In general, you should group applications together in a folder structure away from data files. Grant users only the Read and File Scan rights to the applications (and the Execute right for those systems that require or support it). No ordinary user should need to modify or remove an application file; that is something that viruses, worms, and Trojan Horses try to do. Good security practice demands not allowing your users to inadvertently cause damage or destruction to your systems. For more information about your authentication system's available user rights and their security implications, consult an administrator's guide for the system(s) that you are supporting.

Forgotten Passwords

As I mentioned earlier, Help desk staff spends more time resetting passwords than on any other activity. It's been estimated by the GIGA Group that password resets cost a typical organization $105 per user per year. If you have 1000 users, you're spending more than $100,000 per year on password resets!

One option for saving almost all of this expense is self-service password reset; however, this feature is not yet available for every system. Self-service password reset allows users who forget their passwords to reset the passwords through a Web service from any browser available to them. Typically, the user must answer three to five questions about personal information that should only be known to that user. Elementary systems have stock questions, such as Where were you born?, for which the user needs to supply answers. The best systems also allow the user to create the security questions (for example, What was your nickname in first grade?).

Of course, the typical user has more than one password to remember. There is the network password, the workstation password, the database password (or passwords for multiple applications), all the Intranet Web site passwords, and so on. To address this potential password-reset problem, in Chapter 3, we'll explore SSO services, which hide the multiple passwords and enhance the value of self-service reset. It isn't necessary to implement a full-featured electronic provisioning solution to implement SSO. In fact, many organizations install SSO as the first step towards an electronic provisioning implementation.

Moving a User from One Location to Another

Some vendors and experts refer to user management as lifecycle maintenance. This term reflects the fact that creating an account is far from the end of the work you'll need to do for that user. As people move around the organization, both literally (to a different office, building, city, country) and figuratively (through promotion and transfer), their user accounts will need to be modified. There are three major topics in this area: change in responsibility, change in location, and change in organization. The latter refers to mergers and divestitures and is more of a function of the provisioning technology that we'll delve into in Chapter 3.We'll explore change in responsibility and change in location in the following sections.

Change in Responsibility

Everyone hopes to work their way up the corporate ladder, and even with the high turnover that many organizations face today, there are many people who remain with an enterprise for 10, 20, and even 30 years. Over that time, users' duties and responsibilities will change. The corporate resources they need to access will also change. Your job will be to ensure that users have the access they need to efficiently perform their jobs as well as to see that access they no longer need is removed. This function is often overlooked, but as we become more security conscious, it grows in importance.

Traditionally, the way these changes were handled would be for a user to request access to resources needed for a new position. After getting approval from the user's manager, the resource administrator would grant the access. Invariably, if the administrator attempted to remove access from resources the user should no longer need, the user would claim that a transition period was necessary to finish work with the old resources and train a replacement in working with those resources. The overworked administrator would hope to remember to remove the access in a few week's time, but often wouldn't. Even in the rare instance in which the administrator did remember, *all* of the unneeded rights were very rarely removed.

Role-based access controls can be a real boon to the administrator faced with users who change corporate responsibilities. The use of roles saves the administrator from hunting and pecking to find the correct privileges that the user needs for a new position as well as those that should be removed as a result of the move from the old position.

The transition period can be handled by modern tools that allow you to set an expiration data on the user's use of the role. It's also much easier to spot inappropriate role assignments for a user than it is to spot inappropriate rights assignments to folders, files, and other resources during periodic auditing of your system.

📖 We'll look at auditing needs towards the end of this chapter.

Change in Location

Users also seem to move around a lot—from one cubicle to another, to a different office, a different floor, a different building, a different city, state, or country. The users in Australia shouldn't be accessing the word processor on a server in Switzerland, nor should the users in San Francisco print on a printer in Chicago. Group-based access control can preserve your sanity in dealing with user location changes.

A large part of group assignment is based on geography. All the users on the third floor of building 27 on the South San Francisco campus use the same printer. A group called, perhaps, SSF27-3pr would handle all the rights needed to use that printer as well as be useful in distributing drivers and online manuals for that printer. If a user moves to the fifth floor of the building at 227 State Street in Chicago, you would simply move the user from the SSF27-3pr group to one perhaps called CHI227SS-5pr. You could just as easily call the groups PR-SSF27-3 and PR- CHI227SS-5 if the resource, rather than the location, is the primary way you want to sort the groups.

Automating Change

Using roles and groups certainly speeds the application of changes; scripting will automate the process even more. Imagine clicking on a user object, then dragging it to a different location. Based on the user's roles and groups, the service automatically makes the necessary changes so that not only is the user productive from his or her first day in the new position or location but also the no-longer-needed access rights and privileges are removed.

📖 We'll explore this type of functionality in more detail in Chapter 3.

Retiring the Account When it Is No Longer Required

I've mentioned that it's important to remove no-longer-needed access rights when a user moves to a different position or location. But even more important, from a security perspective, is the need to remove all access rights from a user who has left the organization.

A Cautionary Tale

In the spring of 1993, I walked into the monthly staff meeting at the company (a manufacturer of computer products) of which I was IT director. About halfway through the meeting, I realized that the person who usually represented the design department wasn't there. When I asked about him later, I was told that he had left the company 3 weeks earlier and was now working for a major competitor. This person had access to all of our proprietary design information and, as a senior staff member, had dial-in access to the network. I immediately disabled his account and sent out memos warning everyone about the danger of not disabling the accounts of terminated staff. Two years later, when I left the company, my account (which had Supervisory access to all of the company's resources) remained active for a month—until I phoned my successor and told him to disable it!

You should periodically—on a weekly basis, for example—review the login/logout logs for any anomalies: people not logging on at all or only accessing resources at odd times or from non-typical workstations or locations. Investigate the anomalies to be sure they aren't simply a result of someone on vacation, temporary assignment, or sick leave, but be ready to disable the account if you don't discover a satisfactory reason for the changed behavior.

Better still, cultivate the Human Resources department. Create an automatic tool that performs the function or formalize a way for Human Resources staff to inform you when someone is:

- On leave
- In legal dispute
- No longer with the company

Many of the Human Resources applications used today are built on standard relational database platforms that can be programmed to take an action whenever the contents of a data field changes. Most have a data field for "last day" or "final day" or "employee status" or an alternative that indicates that the employee has left the company. If at all possible, have a change in that field trigger a message to the appropriate administrator (or administrator's group) to remove access rights for that user.

Rather than remove the individual, group, or role-based rights, however, the preferred action is to disable the account. It's possible that at some future time, someone will need to know what the user had access to, which files and folders were created by that user, and which resources the user accessed and when. If the user account is deleted, this information might be lost.

However, even removing the access rights (as well as removing the user from groups and roles) can lead to a loss of information because you can never be sure which resources the user might have been able to reach. The increasing government interest in and regulation of digital data and access—such as the Health Insurance Portability and Accountability Act (HIPAA)—often requires that you be able to produce records showing when, where, and by whom a bit of data was created, retrieved, modified, or removed. Disabling the account and logging the data and time you do so should protect you and your organization if legal proceedings were to ensue (see Figure 2.2).

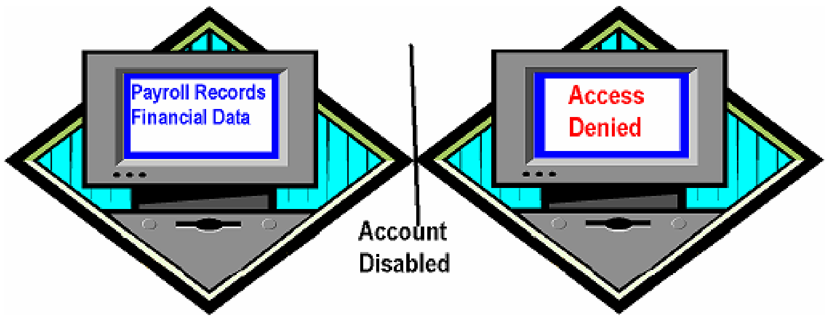

Figure 2.2: With many authentication systems, the user is immediately denied access when the account is disabled.

Considerations for a Successful User Management Project

In Chapter 1, I introduced considerations for a successful user management and provisioning project. In the following sections, we'll explore which of these factors apply specifically to a user management project:

- Define how you will measure ROI
- Take a phased, modular approach
- Choose an appropriate administrative approach
- Secure the user management process
- Ensure that all activity is logged and audited
- Consider Human Resources as the "trigger" for your system

Define How You Will Measure ROI

According to the Gartner Group, "A broad range of factors—including the demands of enterprise resource planning implementation, regulatory compliance issues, and the pressure to contain costs—are intensifying the focus on how enterprises manage the processes associated with granting users access to business information. Identity and access management solutions, which can offer 3-year return on investment in the triple-digit-percent range, are becoming essential tools for effective management of user account and access rights information across heterogeneous IT environments, for Web and non-Web applications" (Source: "ROI Drives Identity and Access Management Implementation," Gartner Group). But how will you measure ROI?

Reduction in Help desk calls—through self-service password reset, for example—is quantifiable. So too is the reduction in administrator time spent in assigning access rights when group-based and role-based access is implemented. Determining the ROI on security improvements, however, is more problematic. The following list highlights suggestions for doing so:

- Perform a manual audit of accounts and access rights, then tally the number that are in need of change or removal. Calculate the amount of time necessary to perform the changes one at a time compared with using roles and groups.
- Carry out a thorough risk assessment (or hire a security consultant to do so) that includes costs for damage and repairs. Perform both a before and after scenario that highlights the changes your user management project will implement.
- Remember that a security breach can have a multiplier effect: the costs of a breach include the costs of validating the integrity of other data—data that is based or relies on the data that has been compromised, the cost of adding security, and the loss of reputation.

Take a Phased, Modular Approach

In the first chapter, I mentioned that a phased approach is a good practice for any Identity Management project. The basic idea is that your total project might take months and even years to implement, but people want to see results in a shorter period of time. By breaking the total project into smaller sub-projects and implementing them one at a time over time, you can keep the implementation team's interest from flagging and provide opportunities for non-team members to be aware of and re-commit their approval of the entire project.

For example, implementing self-service password reset could be fairly easily accomplished as long as the authentication system you use supports it either directly or through third-party services. This implementation will be visible to almost everyone in the organization. In addition, it is a sub-project that has an easily demonstrable ROI that results from the reduction in Help desk calls.

Creating an automated notification system from the Human Resources software's database to your Administrators Group whenever someone leaves the company is a project that won't be visible to most people but is a sure winner with upper management—especially those most concerned with security. Depending on the Human Resources system you use as well as your in-house programming expertise, this task could also be an easy sub-project to implement.

There's also the sub-project I suggested in Chapter 1: a GAL. If your identity storage system allows you to access its data through LDAP, there are many applications and services available (some at no cost) that will allow your users to query the name, location, phone number, and email address of others in the system (provided, of course, that you have that data in the system). This implementation has proven time and again to be a big hit with most users and can usually be implemented in less than a day at little or no cost. This type of initiative is ideal in helping you to sell the larger user management project.

Choose an Appropriate Administrative Approach

Although moving to group- and role-based access methods will reduce the amount of time you and other administrators must spend creating, maintaining, and removing user accounts, managing users still remains one of the least desirable activities in your job description.

At one time, user administration was an all or nothing activity. Either you had full control of the user object or you had none. Today's authentication, authorization, and user management tools allow for much greater granularity of control. It is possible in most systems, for example, to create a role of Password Administrator—a person who can reset a user's password but otherwise has no control over the user's account or access privileges. Of course, the ultimate delegation is self-service password reset—you delegate the administration to the user! Almost all of the duties associated with user management can now be delegated along with controls to ensure that the management is not abused nor causes unwanted security or privacy risks.

The ideal situation is to delegate as much as possible to automated services. In a recent report, the Burton Group noted that "…manually administered environments require a Full Time Equivalent (FTE) for approximately every 500 to 1,000 users, while automated environments can manage 5,000 or more users per administrative FTE" (Source: "User Management," the Burton Group). A reduction in administrative costs by a factor of 10 to 1 could certainly improve the ROI of your project.

Also consider moving parts of the process to non-IT departments. If your identity system supports rules-based administration so that you can implement rules governing the creation of user accounts, for example, you should be able to have the Human Resources department, through the action of their Human Resources software, initiate the creation of user accounts. Although this type of delegation is more prevalent in provisioning services (see Chapter 3), it could be done by having the Human Resources database start a process that both creates the user account and notifies an IT administrator who can then tweak or modify that account. Delegation of these responsibilities doesn't decrease your control; it frees your time to do those parts of your job that you might consider more interesting.

Secure the User Management Process

I encourage you to delegate as much of the user management process as possible, but I must also urge you to ensure the security and integrity of the process at the same time. The more people who have access to a resource either as users or administrators, the greater the risk that the resource will be compromised in some way (either purposely or accidentally). There are three very important aspects to the security implications of your user management project:

- Restricting and authenticating access to the user management system
- Restricting administrators' purview to only those users, groups, and objects for which they are responsible
- Automating and simplifying the process of user management

Restricting and Authenticating Access to the User Management System

In the previous section, I encouraged you to enlist as many people as possible as delegated administrators in order to ease the burden on you and your immediate staff. This step doesn't contradict that recommendation, but it is important to consider in the light of delegated administration. You'll need to institute rules governing the use of the user management tools that restrict them to only authorized users. To enforce those rules, you'll need to be vigilant and audit all access to the user management system. Any breach of the rules should be dealt with quickly and appropriately. Anyone deliberately misusing the system should at least have those privileges revoked. If there is evidence of a security breach, the offending person should be subject to immediate dismissal—a chain of events that should be included in a corporate policy. The security and integrity of your user management system should be paramount.

Restricting Administrators Access

People are curious. People will look at, probe, poke, and manipulate anything they can get their hands (or eyes) on. Telling delegated administrators that although they can easily manipulate a user's access privileges that doing so is forbidden, might encourage them to give in to the temptation to see what might happen. Fortunately, most identity data repositories have rights and privileges associated with the objects they contain.

Just as you can assign rights to folders and files, you can also assign rights to the identity system's objects, which also have access control lists (ACLs). It is this function of having an ACL for each object and property that allows you to create the Password Administrators I suggested in the delegation section earlier.

If your authentication system allows, create delegated administrators for particular functions (account creation, printer administration, password reset, and so on). Be sure that these administrators have access only to the objects they need to see and have the ability to make only the modifications you want them to be able to do.

Automating and Simplifying the Process of User Management

Many of the problems that occur are not caused by someone with malicious intent but by an administrator who isn't giving full attention to the activity he or she is involved with. The creation and maintenance of user accounts is generally a tedious operation. Delegating this task doesn't make it less tedious for those who are performing the process—mistakes will happen. Automation is a solution.

Scripted processes that are initiated by a change—for example, in the Human Resources' database—are the best way to mitigate the problems of administrator error. Of course, if the Human Resources clerk spells a new employee's name wrong or wrongfully indicates someone is being terminated, the result might be embarrassment but rarely will it engender a security breach.

Where you can't fully automate a process, try to eliminate as many data entry steps as possible by having the application or service read that data from an existing computer-based source. If your administrators will need to handle user data in a number of services or applications, try to minimize the differences between the administrative tools they will need to use. The goal is to eliminate the avoidable mistakes that an administrator might make.

Ensure that all Activity is Logged and Audited

Perhaps the most important consideration: whenever the user identity storage is accessed; whenever a user account is created, modified, or removed; and whenever a user's access is granted or revoked an entry should be automatically made in a log file. The log file itself should only be able to be modified (or removed) by an auditor who is not the same person as the administrator whose activities are being logged. Ideally, the auditor will be someone with no connection to the user management process.

Auditing is now required—or at least strongly recommended—by many, if not all, of the recent government-imposed regulatory environments (such as HIPAA and the Sarbanes-Oxley Act). The auditing process itself should be under the strictest security controls while following the recommendation to automate and simplify the process. Most modern user management services allow for a completely automated and transparent audit process. You should implement it immediately when you begin using the service. For older systems, there are third-party auditing packages that can help you ensure that full audit trails are available for all user management activities.

I encouraged you to delegate administration to others and to automated processes as much as possible. Frequently, the ability to make the changes you've delegated rests not with the individual user to whom you've given the administrative role, but to a service that performs the actual task. The delegation simply involves giving the user access rights to the service. When the activity is logged, then, it is the service that is identified as doing the activity and not the user who is running the service. If your delegation works in this manner, be sure to associate the user management functions performed by the service with the identification information for the user who is running the service at that particular time.

Thoroughly test in a non-production environment the auditing functions and capabilities of the system you are using. The time to discover problems is before going into production, not after you've received a subpoena from court.

Consider Human Resources as the "Trigger" for Your System

In the section on delegating administration, I suggested that you cultivate contacts in the Human Resources department and that, where possible, you automate the user management function through the use of the Human Resources database and applications. Doing so will almost be a requirement when you want to move to a full provisioning project (as we will discuss in Chapter 3) and is very desirable for your user management project. By having the user account creation, maintenance, and removal initiated by activity within the Human Resources database, you go a long way towards delegating administration as well as automating and simplifying the process (and, as I mentioned earlier, preventing manual mistakes).

The best way to enlist the help and buy-in of the Human Resources department depends on your environment—office politics often play a significant role in IT implementations. To aid in the justification of this type of implementation, highlight how the process will minimize the work that must be done by the Human Resources staff—either through automation or simplification. If, for example, the trigger you place in the Human Resources database to initiate creation of the user account could also notify the payroll department—with, say, the user name, social security number, employee number, and pay rate—will save a step for the Human Resources staff, your project will be looked upon favorably (see Figure 2.3).

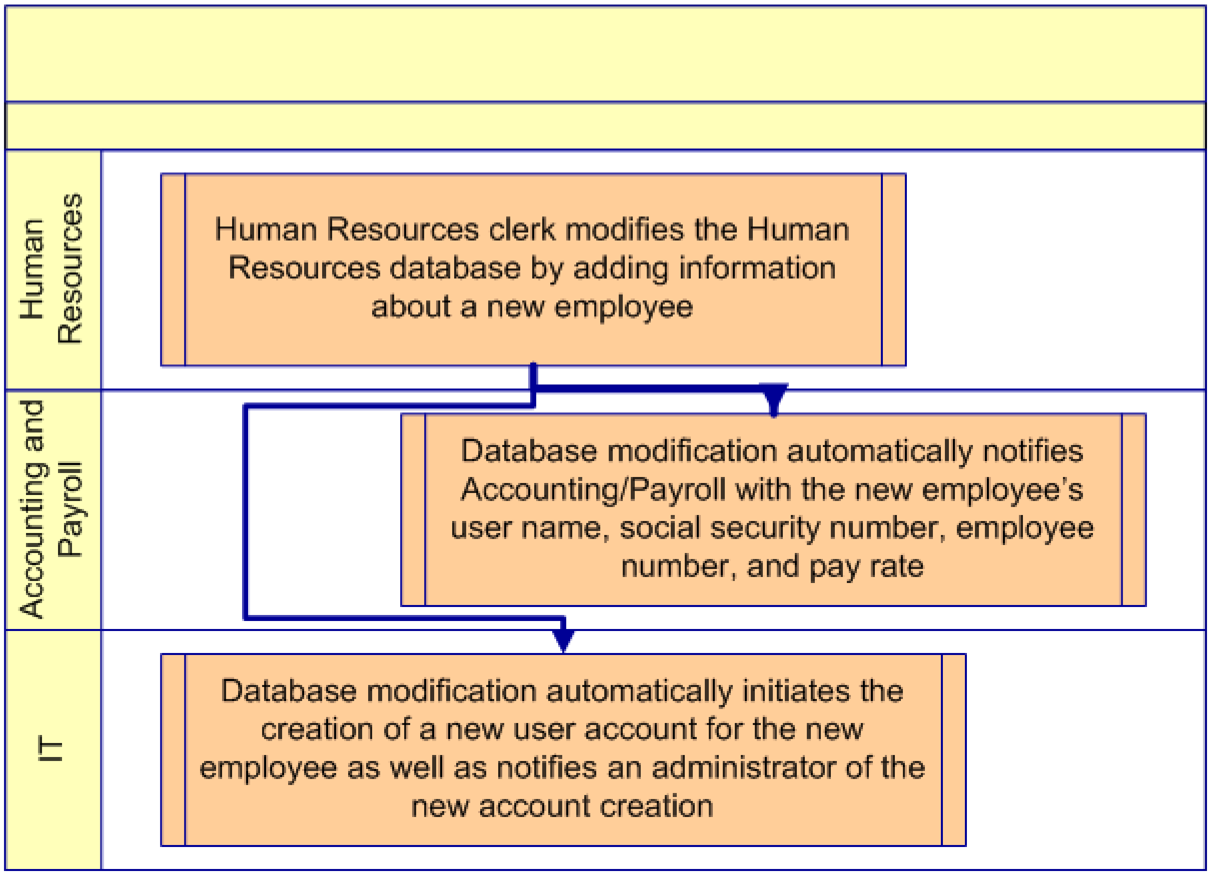
Human Resources
Human Resources clerk modifies the Human Resources database by adding information about a new employee
Accounting and Payroll
Database modification automatically notifies Accounting/Payroll with the new employee's user name, social security number, employee number, and pay rate
IT
Database modification automatically initiates the creation of a new user account for the new employee as well as notifies an administrator of the new account creation

Figure 2.3: Implementing an automated Human Resources "trigger" eases user management as well as the workload of the Human Resources staff.

Outsourcing

The ultimate in delegated administration is to move the entire process out of your company to some third-party resource that will handle all of the tedious details for you. As with just about every choice you must make, there are pros and cons to be considered.

Small organizations—certainly those that employ less than 100 users (but more than 20 users) and those that employ as many as 500 users—should seriously consider outsourcing. These are organizations with only a one- or two-person IT department (when the entire department isn't outsourced) who must be generalists rather than specialists in security or user management. Care must be taken by small organizations in regulated industries—healthcare, securities, banking, and law— to ensure that all required safeguards are taken to protect the security and integrity of the user data even if the user management is outsourced.

Larger organizations might well have all the required expertise on hand, but might choose to outsource parts of the user management process. Password resets as part of an outsourced Help desk are one example of outsourcing by service. Alternatively, you might outsource by geography. It might be cost advantageous for smaller, more remote or isolated offices to have their users managed independently by a third-party organization. When dealing with multi-national organizations, the expertise of a local outsourcing company might be very useful in determining the correct procedures to comply with local government requirements. Privacy requirements, for example, vary widely by country but need to be enforced for the workers in that country. Using a user management provider in that location could be necessary in order to be sure of compliance.

No matter what reason you have to choose a third-party organization to provide outsourced user management, you need to have detailed, binding, written service agreements in place. When a user needs access but the user management provider isn't answering the phone, it is you and the IT department that will be held accountable. There is nothing inherently wrong with using a small organization for your outsourcing partner—it is, after all, specialized knowledge that you're licensing—as long as you have guarantees that they will do the job in an accurate and timely manner.

Summary

A user management project can be a tremendous asset to both your IT organization and to the enterprise as a whole. Improved security, better access control, lower cost of account maintenance, and greater user satisfaction can go a long way towards improving the image of the IT department.

In this chapter, we explored all aspects of user management categorized by the basic aspects of the user account lifecycle:

- Creating an account
- Granting the account the correct access
- Resetting forgotten passwords
- Moving a user and maintaining the account
- Retiring the account when it is no longer required

We've also seen which of the successful user management and provisioning project considerations apply specifically to user management and examined the best practices they suggest.

In the next chapter, we'll take user management, which we've been looking at in a system-by-system method, and extrapolate it to the entire organization. Electronic provisioning, in theory, allows you to automate every aspect of user management and the user account lifecycle.