V I S T A

SENSITIVE INFORMATION

File Manager 22.2

Security and Privacy Manual

March 2013

Fileman/
Lab
Agile
Project

T³TigerTech
Trusted Talented Tireless

vista
Expertise Network

Revision History

Date	Description	Language	Authors
January 2013	Created	English (US)	Eileen Gormly, Kathy Ice, Frederick D.S. Marshall, and Carol Monahan

Contents

Orientation

This manual discusses security- and privacy-related features and options in Fileman 22.2. The manual assumes that Fileman 22.2 is already installed and running on your system.

This manual uses VA conventions for displaying mockups of sensitive data in illustrations and screen captures. The first three digits (prefix) of any social security numbers (SSN) will begin with "000." Patient names will be formatted as FMPATIENT,[N] where "N" represents the first name as a number spelled out.

Screen captures and computer source code are shown in a non-proportional font and enclosed within a box. The user's responses to online prompts will be boldface.

Introduction

What is Fileman?

In discussing what Fileman *is*, it is probably best to begin with what Fileman *isn't*. Fileman isn't a clinical package like Laboratory or Pharmacy, and it isn't an administrative package like IFCAP. Fileman is an infrastructure package, meaning it is part of what makes VISTA run. Programmers use the features in Fileman when creating those other packages: Lab and Pharmacy and IFCAP and so on.

"Fileman" is the shorter, more informal name of the package; File Manager is its formal name. As you may have guessed from the name, the specific part of the infrastructure that File Manager handles is files. That is, Fileman manages VISTA's massive database. Fileman commands are all about data: entering it, editing it, retrieving it, sorting it, printing it, making it into reports.

Although Fileman was written for VISTA, it does not need to be used with VISTA. It can be installed as a standalone database management system.

Fileman Security Environment

When considering Fileman security, it is important to note that it is likely not the only security running on the system. The computer's operating system has its own security. If the computer is on a network, the network includes additional security, both internal (access control) and external (firewall). If Fileman is running as part of VISTA, much of the security will be handled by the Kernel package. For information on Kernel's security features, please see the *Kernel Security Tools Manual*.

To fully understand and take advantage of Fileman's security features, you must also understand the security provided by the operating system, by the network, and by Kernel, if applicable.

What Do You Mean by That?

Understanding Fileman security means understanding a web of closely-related features shared between Fileman and Kernel, many of which have similar names. When dealing with ordinary VISTA packages, the difference between (for example) an "Access Code" and a "Fileman Access Code" is not as important, but when dealing specifically with Fileman security, terminology is crucial.

VISTA programmers and system managers (along with 95% of all human beings on Earth) tend to get a little careless with terminology when speaking about things they understand pretty well. After all, *they* know what they mean. They probably don't even realize that they may be using terminology in ways that might confuse a listener. Don't be afraid to ask for clarification. "When you say 'Access Code,' what do you mean by that?"

This manual was created with the help of VISTA programmers and system

managers. And while we've made every effort to be clear and specific in how we use terminology, we cannot make an absolute promise that some of those careless habits didn't sneak in. Don't be afraid to ask us for clarification either; representatives of the authors of this manual—and often the authors themselves—can be contacted through OSEHRA.

The Glossary of this manual contains definitions of most important terms for Fileman security. Here, we present a few terms that are often confused.

Access Codes

In a VISTA system, users sign in using an Access Code and a Verify Code. These codes control access to VISTA, and indicate to the system which user this is, and what permissions they have. This information is handled by the Kernel.

In Fileman, a part of user security is handled using Fileman Access Codes. Each file, field, and template can be given a Fileman Access Code. Users can be assigned matching Fileman Access Codes, which represent the files, fields, and templates they have permission to use. These codes are part of the user's profile. Unlike the Access and Verify Codes, they are not something the user has to type into the system.

In this manual, we will use "Access/Verify Code" and "Fileman Access Code" to keep these similar terms from being confused.

Files

"File" is one of those lovely words we have in English that can mean a whole lot of different things. We'll just say right at the outset that we won't be using "file" as a verb, and we won't be using it to describe the thing you use on your fingernails, or a column of soldiers. We can even eliminate paper files, the kind that come in file folders and are filed in file cabinets.

Even with all that out of the way, there's still a lot of room for ambiguity. Fileman is a database management system. With Fileman, users can create,

maintain, edit, view, and use files. However, Fileman is not just a producer of files; it is also a consumer. That is, some of the files in a Fileman system are the files that Fileman is actually using. And those files—the ones Fileman uses in order to perform its functions—generally cannot be modified by users.

So: "When you say 'Fileman files,' do you mean files created by users using Fileman, or the files that Fileman uses?" That can sometimes be an important question to ask your IT team.

And it's even a little more complicated than that! Because the things that VISTA and Fileman call "files" are not recognizable as "files" to any other computer language or system. They are not system files. They will not show up on anybody's computer with that adorable little file-folder icon, which says "I am a file. Please copy me for my tasty information!" We don't want to head too far down a technical rabbit-hole here, but essentially a "file" in VISTA or Fileman is a variable with a lot of information stuffed into it. And if that sounds a little bizarre, well, that's because you're paying attention.

But it works. And more importantly, it means that you as a Security officer don't need to worry about Fileman files showing up on the system as copyable objects. The only way to copy Fileman files is by using Fileman's own tools—which are protected by the same security as the files themselves.

Keys

This one isn't specific to Fileman or VISTA. A "key" means something different to a database administrator than it does to a security officer. In a database, a "key" is how the system identifies a specific record. The key may be a specific field, or a combination of two or more fields. Either way, each record in the database has its own unique key. This is how the system can, for example, call up one patient's record out of the thousands of patient records it has stored.

In security, a “key” is an electronic code that allows the user to access certain areas of the system (it is also a metal or electronic device that opens doors, but we’re going to ignore that one). The key may be something that the user has to enter manually, or it may be a code associated with the user profile. Either way, keys are a way of limiting access to certain areas of the system.

In this manual, we will use “database key” or “file key” when speaking about the first kind of key; and “security key” when speaking about the second kind of key. We recommend that you do something similar.

Chapter 1: Access Authorization

The first layer of security on any system running Fileman is external to the Fileman package itself. In an integrated VISTA system, access is initially handled by the Kernel package. In standalone Fileman, this function is handled by the operating system's native security layer. You should familiarize yourself with the underlying security protocols in order to get a complete picture of Fileman security. Please consult your operating system manual, or the Kernel documentation suite, as appropriate.

Before we plunge into the nuts and bolts of how users access Fileman, it is important to note that *end* users do not access Fileman—at least, not directly. In a VISTA system, the users interact with Fileman through the specific packages they use to perform their job functions. Their Fileman access, therefore, is defined not by Fileman or Kernel, but by Laboratory or Pharmacy or Nursing or IFCAP. Direct access to Fileman in a VISTA system is limited to programmers, system administrators, and super-users (CACs, ADPACs, and so on). And you, of course. This is a key point, and one that we will probably bring up again because it's important to keep in mind. Fileman users are not end users.

Menus, Options, and Security Keys

In a VISTA system, authorization for Fileman (and indeed for all packages) is handled through Kernel. One of the modules of the Kernel packages is called Menuman, and it can show each user a custom menu containing only those options the user is authorized to access. In Kernel, the specific option that handles Fileman access is the DIUSER option.

In standalone Fileman, users either have access to the system or they do not; there is not a system native to Fileman that can allow different types of access for different types of user. There is a basic menu system (essentially a very primitive Menuman) that presents the complete list of Fileman options and responds to user input.

This native menu system does not include the SQL projection options or the Fileman management options. These options are available in standalone Fileman, but programmers must know their entry points in order to access them.

It should be noted that a site running Kernel can opt not to use Menuman's menus for Fileman access. The native Fileman menu system is available to everyone, and programmers in particular often prefer to use it. For the most part, this manual assumes that a site running Kernel is using Menuman for Fileman access. We will make the further distinction—between Kernel sites using Menuman and Kernel sites not using Menuman—if it is important to the feature or option being discussed.

In a system running Kernel, security keys are used in conjunction with Menuman to tailor menu options for users. Menuman menus and options are generally designed for roles, not individuals. But within a given role, there can be individual variations.

For example, one role available in the Laboratory package might be "Lab Tech." But in a hospital lab, there might be junior lab techs, senior lab techs, supervising lab techs, and so on. Some of the options available to the "Lab Tech" Menuman role might not be appropriate to give to the more junior levels of the staff. The solution is to give the "Lab Tech" menu to all the techs, but use security keys to lock certain options so that only the appropriate staff could use them.

When options are locked, Menuman does not even show them to the user. In our lab tech example, the junior techs would see a Lab Tech menu, but it wouldn't list all the Lab Tech options, just the ones that they had access to.

With Fileman, users are often given access with DIUSER. But not all of the options in DIUSER are appropriate for all users of Fileman. Some DIUSER options are locked with the security keys described below.

XUAUDITING This key is needed to access the Auditing submenu, which contains all of the auditing options. These options are restricted

because it's not a good idea to allow users to turn auditing on and off at will. The key should be given to anyone who needs to perform auditing functions, including system administrators, Security officers, Privacy officers, and any programmers directly supporting Fileman.

XUFILEGRAM This key is needed for most of the filegram options. A filegram is a method for extracting a single record in a portable format. This has obvious HIPAA and patient-privacy implications, so most filegram options are located in a filegram submenu, accessible only to users with this key. The exception is the View Filegram option, which is not locked; any user can look at a filegram. That is, any Fileman user can look at a filegram. Remember that Fileman users are not end users; when we say "all users," we're actually talking about a small number of people. The XUFILEGRAM key should be given to system administrators and anyone whose job role includes moving records.

XUMGR This key is needed to access Fileman Management submenu, which includes options for installing and configuring Fileman. It is also needed to access many Kernel options. This key should be given to system administrators, some senior programmers, and whichever programmer or programmers are responsible for patching Fileman and keeping it up-to-date.

XUPROGMODE This key is needed to enter "programmer mode," which most programmers need to be able to do. Therefore, this key ends up being given to more people than will actually be using the specific Fileman options it unlocks. These options are the Regenerate SQLI Projection and Purge SQLI Data options. Both of these options have to do with setting up an interface between Fileman and a SQL query engine.

XUSCREENMAN This key is needed to access the Screenman menu. This is the menu that allows programmers to create and change Screenman forms. This key should be given to anyone authorized to manage Screenman forms.

DDXP-DEFINE This key is needed to access the Export Tool's Define

Foreign File Format option. The Export Tool is used to move Fileman data into another software format: for example, an Excel spreadsheet. Super-users often use the Export Tool to check and verify data. Most of the formats the super-users need have already been defined, so the ability to define new ones should be restricted to programmers who understand how the tool works, and how to define a new format.

DIEXTRACT This key is needed to access the Extract Data to Fileman File submenu. This submenu allows users to extract data from existing Fileman files, and save it in new files that they create. An extracted file is a good option for a user who needs to work with a specific set of data, but does not want to hamper or slow down the main system.

Fileman File Security

The menus and security keys provide access to Fileman's options, but each file also has its own security. It is possible, therefore, for a user to have access to Fileman, or to a set of Fileman options, but not to have access to any files. In other words, they could select an option, but they could not select a file to use the option on. Users hate that.

Therefore, when granting a user access to Fileman, it's important to pair their Kernel/Menuman access with access to at least one file. This will allow the user to at least try out some of their new options, even if they can't do much yet.

Each file in the system has six possible types of access that can be granted to users. Arranged from lowest level of impact on the database to the highest:

> **Read access** allows users to see what is in the file, but not make changes. Related Fileman options include Inquire to File Entries, Print File Entries, and Search File Entries.
>
> **Write access** allows users to edit files, but not add or delete records.

Related Fileman options include Enter or Edit File Entries, and Transfer File Entries.

LAYGO (learn-as-you-go) **access** allows users to add records. Related Fileman options include Enter or Edit File Entries.

Delete access allows users to add and delete records. Related Fileman options include Enter or Edit File Entries, and Transfer File Entries.

Audit access allows users to turn auditing features on or off. Related Fileman options include all of the options on the Auditing submenu.

Data-dictionary access allows users to change the nature of the file: that is, change the file and field attributes. Related Fileman options include Modify File Attributes, Utility Functions, and Data Dictionary Utilities.

Although the six levels of access have a definite hierarchy, they are independent of one another. That is, a higher level of access does not "contain" the lower levels. If a user is given write access to a file, they also need be given read access. Otherwise, they'll be able to edit the file, but not read it, which makes no sense.

These six levels of user access are always available in Fileman; however, there are two possible mechanisms for how the access is actually handled.

Fileman's native mechanism for user access is the Fileman Access Codes. When this system is in use, each file, field, and template can have its own specific Fileman Access Codes. Codes can then be assigned to users as user attributes.

On a system running Kernel, Fileman Access Codes can be replaced by an alternate mechanism called File Access-by-User. In this method, each user is matched with one or more files that they can access. Users and files are matched through Kernel; users are not given Fileman Access Codes. Sites

using this method usually set up standard profiles for different types of users, similar to the role-based Menuman options.

The primary difference between these two systems is that under Fileman Access Codes, the default is "fully accessible." That is, if a file does *not* have a Fileman Access Code, the system will allow anyone to access it. Fileman Access-by-User is the exact opposite; the default condition is "no access." If a user has not been matched up with a file, the user will not be able to see or use anything.

If a VISTA system is using File Access-by-User, the Fileman Access Codes are suppressed, with a few specific exceptions.

First, you may have noticed that Fileman Access Codes can be used for files, fields, and templates. But Access-by-User only applies to files. Therefore, if users need to have access to templates, even templates associated to files they've been matched with, they have to be given any applicable Fileman Access Codes.

Similarly, if specific fields within a file have been given Fileman Access Codes, the user will need to be given those codes as well. The Access-By-User permissions will not override the field codes.

Finally, two Fileman Access Codes are not suppressed when Access-By-User is active, just because the codes are so useful. The first of these is the "programmer's at-sign" (@), which gives a user access to every file. Using the @ is easier than manually assigning every single file to the user (and remembering to go back and assign any new files that are created!). The other Fileman Access Code that is not suppressed is the up-arrow (^), which indicates a file that cannot be altered by anyone. This code is typically found in the files that Fileman uses to operate.

The intent is that the ^ overrides the @; that when a file says "nobody can access this," that "nobody" includes programmers. The ^ does work that way for write access at the field level; however, there have been bugs implementing it elsewhere, including at the file level. The Fileman team is

working to fix these, but for now, be aware that a ^ won't necessarily keep programmers out. It will, however, keep everybody else out.

As a Security officer, your biggest challenge with these two systems (Fileman Access Codes and File Access-by-User) will be if your facility decides to switch from one to the other. Such a switchover requires some painstaking work and attention to detail. On the authors' "to-do" list is a set of complete instructions for handling this kind of switchover. Until that is available, we strongly encourage you to reach out to other Security officers, either via OSEHRA or your own contacts, to gain the benefit of their experience.

Fileman's Own Files

The security measures described so far in this chapter apply to all the files created and maintained through Fileman, with one important exception. The files that Fileman needs to operate—Fileman's own files—have their own security and their own rules for access. Some of these files are accessible only to those with programmer access: those who have "@" as their Fileman Access Code. Many of these files are not accessible at all, at least not to humans. For the most part, Fileman trusts only software when it comes to its own files. Programmers can, at times, use specialized data-dictionary options to modify these files, but the normal options for editing data will not usually work.

In terms of file numbers, any file with a number below 2 belongs to Fileman. That is, it is one of the files that Fileman needs in order to function.

Practical Application

In both standalone Fileman, and in Fileman as part of VISTA, access to Fileman files and options is managed through menus (for options) and file permissions (for files). As a Security or Privacy officer, you need to be aware of which employees have access to which options and files. You should also be notified when permissions are changed.

Your facility should have a process in place so that employees can request access to the options and files they need to do their jobs, and so that supervisors and managers have the appropriate oversight. You should probably be a part of this process—but even if you are not, you should ensure that part of the process is notifying you of the change.

When Is File Access Security Checked?

When a user is in Fileman, whether standalone or as part of a VISTA installation, Fileman checks their security permissions every time they access a file. It's pretty straightforward. However, Fileman permissions can also be checked at other times.

As we mentioned at the beginning of the chapter, end users do not typically have direct access to Fileman. This means that most users do not have access permissions to Fileman files. Yet end users read, update, and even add to files all the time, using VISTA options. How does that work?

When programmers are building VISTA options, they can use a variety of MUMPS commands that tell Fileman what is going on. For example, they might use a command that tells Fileman the user will be adding a record. If that command is used for the option, Fileman will not check the user's permissions for the file when the option is invoked. Fileman is expecting the user to add a record, and because VISTA said it was okay for the user to add a record, Fileman doesn't check the permissions. This is how, for example, an admissions clerk can add a new patient to the PATIENT file, even though she doesn't have any kind of Fileman access at all.

For the most part, these options belong to the standard VISTA packages as they are installed. Any security implications have already been identified and described in the documentation for those specific packages, so you should have that information at your disposal.

However, if the programmers at your facility create special VISTA options for users, and if any of those options include the ability to see or change the

database, then you as the Security officer need to be aware of them. If you don't have one in place already, set up a notification process that will give you visibility for any new options your programmers create that will affect user access to the database.

Understanding ^DIC and ^DIE

There are literally thousands of commands, or *calls*, that can be used to create options. Fortunately, you don't need to learn them all. There are two calls, however, that can interact with Fileman's file permissions, so it is a good idea to understand a little about how they work.

The first is ^DIC. This is a lookup call; that is, it allows the user to look up or select records in the database. The second call is ^DIE. This is an edit call, which allows users to edit records in the database.

That's pretty straightforward so far. But VISTA developers discovered fairly early on that when users are looking up records, they often end up wanting to add a record. And when users are editing records, they often end up wanting to delete a record.

Of course, just because users want to do something, that doesn't make it a good idea. Programmers have flexibility when using these calls, so that they can allow users to perform these additional functions, or not, as the situation warrants.

When a programmer builds an option using ^DIC, they can specify that the option is lookup-only, that the user will not be allowed to add records under any circumstances. Or, the programmer can specify a "LAYGO lookup," meaning that a user who has LAYGO access to the file would be able to add records.

In an option built with LAYGO lookup, Fileman allows the user to look at the data without checking their file permissions. Fileman, remember, is working with a ^DIC; it is expecting the user to be looking up and selecting records.

What Fileman isn't really expecting is the user wanting to add records; there is a different call for that. However, because the programmer has set up the option to allow LAYGO lookups, Fileman doesn't automatically refuse a request to add records. Instead, Fileman knows to check the user's file access permissions. If the user does have LAYGO access to the file, Fileman proceeds with the transaction. If the user does not have LAYGO access to the file, the transaction is not allowed.

So far so good. However, programmers can add one more wrinkle into a ^DIC option. A programmer can set the option to allow LAYGO-lookup, and include a special variable called DLAYGO. This variable tells Fileman not to bother checking permissions—that any user who wants to add a record can do so. In effect DLAYGO gives LAYGO access to all users using that particular option.

There is a similar, but not completely identical, situation with ^DIE. Options built with ^DIE allow users to edit. Programmers can also allow deletions from the option, simply by including the file's .01 field as one of the fields that can be edited. As you probably know, deleting a record's .01 field will delete the entire record. Therefore, if an option built with ^DIE includes the .01 field, record deletion is possible.

If a user attempts to delete the .01 field, Fileman knows to check whether the user has delete access to that file. If the user has delete access, Fileman gives the okay and the transaction is allowed. If the user does not have delete access, the transaction is not allowed, and the .01 field is not changed.

There is, as you may have guessed, a special variable that programmers can use with ^DIE options that tells Fileman not to check a user's file permissions. The variable is DIDEL, and it essentially gives delete access to all users using that particular option.

Because ^DIC and ^DIE interact with file access security, it is important to understand how they work.

Access from Another File

As you have probably seen, the data files in Fileman are highly interconnected. Files point to other files, which point to other files, which point to yet more files. Computed fields can pull data from all over the place. This is part of what makes VISTA such a useful tool in medical informatics.

It can be difficult, however, to track Fileman permissions through all those pointers. So at the present time, Fileman does not even attempt it. When a user has Fileman access to a file, they have access only to that file. If they try to "follow" a pointer to edit or even read another file, Fileman will not allow it unless the user specifically has access to that file as well.

Because of this, you may occasionally need to give users access to pointed-to files in order for them to fully manage the files they control. Such access is usually granted on a case-by-case basis.

Security and the Line Editor

The Line Editor is one of VISTA's two native word-processing tools. Most users prefer the newer Screen Editor.

However, the Line Editor includes an ability that the Screen Editor does not have. Using the Line Editor, a user can copy lines from another "document" (that is, any other word-processing field) into the document currently being edited. Basically, a user can use the Line Editor to grab text from anywhere and paste it into their document.

When a user does this, however, Fileman does check their file permissions to ensure that they have read access to the file they're trying to use.

Chapter 2: Fileman Access Codes

If your system has not implemented File Access-by-User, your access to files, fields, and templates will be regulated by which Fileman Access Codes each user has, and what codes have been assigned to those files, fields, or templates. Specifically, each user's Fileman Access Code is stored in a special variable, which is never supposed to be erased. (If it is, you get to yell at somebody, although you may have to get in line.) The name of the variable is DUZ(0); you will sometimes hear programmers refer to it by name.

DUZ(0), where the user's Fileman Access Code is stored, is associated with another variable, called simply DUZ. The DUZ variable contains the user's identification code, and, like DUZ(0), it is never supposed to be erased.

Security at the File Level

At the file level, user access is managed either through Fileman Access Codes or through File Access-by-User. This section discusses the specifics of Fileman Access Codes. For a discussion of File Access-by-User, please see Chapter 3.

In a Fileman Access Code, each individual character has a specific meaning. For example, a specific file might have a "p" for its write access. Any user who had a "p" in their Fileman Access Code would be able to edit the file. Another file might have "Pp" for its write access. In this case, any user who had a "P" or a "p" in their Fileman Access Code would be able to edit the file.

A file might have one character as its Fileman Access Code, or it might have more than one. If it has more than one, then any one of the codes listed will provide access. A user may also have one or more characters in their Fileman Access Code, which represent all the types of files, fields, and templates they have access to.

A list of Fileman Access Codes, and which packages they are typically used for, can be found in Appendix A.

When a user signs into VISTA, the Kernel package verifies the user's identity. It then sets the value of the DUZ variable to the user's identification code, and the DUZ(0) variable to the user's Fileman Access Code. Fileman then uses the value of DUZ(0) to determine which files the user can access, and at which level.

Standalone Fileman is accessed from MUMPS programmer mode. The programmer has a choice of commands they can enter in order to launch Fileman. The commands are summarized in the table below:

Command	value of DUZ(0)	Symbol table erased?
D ^DI	unchanged*	Yes
D C^DI	unchanged*	Yes
D D^DI	unchanged*	No
D P^DI	@	Yes
D Q^DI	@	No

The first two commands, D ^DI and D C^DI are functionally identical. Essentially, if a programmer uses "^DI" by itself, C^DI is the default.

*For the first three commands, the value of DUZ(0) is unchanged. That is, if the programmer has previously set DUZ(0) to a specific value, that value is retained. If the programmer has not set DUZ(0), it has a null value.

The symbol table is the part of MUMPS where the programmer's local variables have been stored. Sometimes the programmer wants to keep these when going into Fileman, and sometimes they want to get rid of them.

In MUMPS, local variables are used only by the programmer who created them. Erasing them will not affect the system, and will not affect any other

users. It may affect the task the programmer is working on, but they can usually fix their own mistakes.

As we mentioned in Chapter 1, there is an "all-access" Fileman Access Code: the programmers' at-sign, @. A user with @ as their Fileman Access Code doesn't need any others; @ will allow the user full access to all files, except the ones specifically locked with ^. On the other hand, a file, field, or template that has @ as its Fileman Access Code is accessible only to programmers.

We mentioned this in Chapter 1, but it bears repeating: if a file has no Fileman Access Code at all, it is available to everyone. There are times when this is perfectly appropriate. Everyone, for example, could have read access to the STATE file; there's nothing secret or confidential about the names and abbreviations of states.

However, sometimes an absence of Fileman Access Codes is the result of a mistake. If your facility is using Fileman Access Codes for file security, we recommend that you check your files to make sure that they all have appropriate Fileman Access Codes.

Changing Fileman Access Codes on Files

When a user creates a new file in Fileman, the new file is automatically given Fileman Access Codes that match the user's. This may or may not be what the user wants. For example, if a programmer creates a new file, it will have a Fileman Access Code of "@," meaning only other programmers can see it.

We recommend that, before creating a new Fileman file, a user should give some thought to what the new file is for, and who ought to be able to use it. After creating the file, and adjusting the Fileman Access Code, the user should document the new file, including its purpose, its Fileman Access Code, and the reason for how the code was set up.

If the user who created the file was not a programmer, they may need to

contact the IT staff for assistance in modifying the Fileman Access Code. This should not be seen as a reason to skip this step, or to try to make do without the new file. If the file permissions need to be changed, then the IT staff needs to find the time to change them.

Fileman Access Codes can be set and changed for files created locally. In addition, the Fileman Access Codes that come "pre-installed" on the core VISTA files can be changed, and sometimes should be changed, to meet the needs of your facility. If your organization does make the decision to change these codes, be sure to document what files were changed, and why.

If your facility has changed Fileman Access Codes on the core VISTA files, then the programmer responsible for patching Fileman needs to be careful when installing patches. Some patches may include new versions of these files, and they may re-set your Fileman Access Codes to their old levels. If this happens, your documentation will assist the programmer in getting the files back to the settings your organization has chosen.

Rather than going through all that, there is an option where Fileman can ask, as the files are being installed, whether you want to override the default Fileman Access Codes. However, that option has to be set by whoever is distributing the patch—whether it is National VA, National IHS, OSEHRA, or some other organization. If your facility often uses its own Fileman Access Codes, you might want to ask for this option to be set up. Even if you ask, you might not get it—but if you don't ask, you *certainly* won't get it.

Security at the Field Level

As we discussed in Chapter 1, Fileman Access Codes can be applied to individual fields, as well as entire files. Even if a facility is using the Kernel-based File Access-by-User option, fields that have their own Fileman Access Codes will still be inaccessible to users without the corresponding codes in their user profile, even if they have access to the file.

This system allows a facility greater flexibility when it comes to giving permissions to users. It may be, for example, that some fields of a patient file should be accessible only to users from Billing, and that other fields only be accessible to physicians. Fileman Access Codes at the field level can allow a facility to fine-tune its user permissions.

At the field level, the possible permissions are read, write, and delete. You may have a programmer try to explain to you that LAYGO access is possible, under certain circumstances, for the .01 field, except it isn't really for the field, except it's stored in the field.... Just smile and nod. Read, write, and delete are the only permissions you need to worry about at the field level.

Keep in mind that field-level Fileman Access Codes will only apply to people who can access the file in the first place. They will not apply to end users (who, remember, are not Fileman users). They will not apply to users from another department or service, who shouldn't have access to the file in the first place. Field-level Fileman Access Codes are a way of refining access for the subset of users who already have access to the file.

The most common use for Fileman Access Codes at the field level is to designate some fields with ^ in their write and delete access. That is, to make these fields updatable only by the computer, and not by any human users.

As with files, your facility can change pre-set Fileman Access Codes for fields. And, as with files, your facility will need to document these changes and take care when patching.

Security at the Template Level

Like files and fields, Fileman templates can have their own Fileman Access Codes. As with fields, templates' Fileman Access Codes override the Access-by-User permissions, so users who need access to a template will need the corresponding code.

Like new files, new templates are created with the user's Fileman Access Code as a default. And again, this may or may not be the desired outcome, so the codes should be adjusted if necessary.

Templates can only be given read or write access. The ability to create and delete templates is managed through Fileman options.

Chapter 3: File Access-by-User

If your system is running Kernel, your organization has the option of replacing Fileman Access Codes with the Kernel-based File Access-by-User. As we have already discussed, File Access-by-User replaces Fileman Access Codes only for files; field and template access is not overridden.

Technically, File Access-by-User is managed through the Kernel package, not the Fileman package. So technically, we should be sending you to the Kernel documentation suite to learn how to use it. However, File Access-by-User is an important tool in Fileman security, so we are giving you a basic overview of how it works. If you would like to learn more than what we have here, please consult the Kernel documentation.

Granting Users Access to Files

You can grant file access to users in the File Access Security submenu of the User Management option in Kernel.

```
Select Systems Manager Menu Option: USER Management
Select User Management Option: FILE Access Security
Select File Access Security Option: ?

          Grant Users' Access to a Set of Files
          Copy One User's File Access to Others
          Single file add/delete for a user
          Inquiry to a User's File Access
          List Access to Files by File number
          Print Users Files
          Delete Users' Access to a Set of Files
          Remove All Access from a Single User
          Take away All access to a File
          Assign/Delete a File Range

Select File Access Security Option:
```

Yikes! That's a lot of options. Fortunately, we can group them into categories that make it easier to remember what they do.

Three of the options deal with **listing** permissions—that is, with just looking at what kind of access people have. These options are: Inquiry to a User's File Access, List Access to Files by File Number, and Print Users Files.

Three options deal with **editing** permissions. These options are: Grant Users' Access to a Set of Files, Copy One User's File Access to Others, and Single File Add/Delete for a User. Most sites begin by using the Grant option, but as more users are set up, the Copy option becomes a better choice. You can select a few "model" users whose permissions are appropriate for a certain job category, then just copy their permissions any time you add an employee to that job category.

Three options deal with **restricting** or **removing** permissions. These options are: Delete Users' Access to a Set of Files, Remove All Access from a Single User, and Take Away All Access to a File.

The only option left on the list is **the weird one**. Strictly speaking, it probably doesn't belong on this menu, although it would be difficult to say where it would belong. You can use the Assign/Delete a File Range option to give a user a limited ability to create new files. You do this by assigning the user a range of file numbers they can use—obviously, choosing file numbers that are not in use and not likely to be used anytime soon. This option does have security implications; if the user already has read access to a confidential file, they could create a new file, and copy confidential information into it.

For more information about how to use each of these options, please consult the Kernel documentation suite.

In File Access-by-User, each of the six access levels of a file (read, write, LAYGO, delete, audit, and data-dictionary) is associated with a YES or NO in the user's profile. As we mentioned in Chapter 1, the default is a NO. That is, no users have access to any files unless that access has been granted by somebody.

As with Fileman Access Codes, the access levels in File Access-by-User do not "contain" the lower levels of access. If a user has write access to a file, they will also need to be given read access, or they won't be able to look at the file they can edit!

Recommendations

Some facilities and organizations who have switched over to File Access-by-User seem to be assuming that Fileman Access Codes are no longer needed. They have stopped giving their users Fileman Access Codes, even though those codes may be needed to access certain fields and templates.

Worse (from a security standpoint), they have begun removing Fileman Access Codes from files. This doesn't affect them; they still have File Access-by-User to control access to the files. If they share these files, however, or distribute them as part of a patch, then they are sending completely unsecured files to organizations that may not have File Access-by-User in place.

This is a dangerous practice, and we recommend that it stop. We recommend that you, as a Security officer, do what you can to ensure that Fileman Access Codes remain in place. They are single characters; they use almost no computer memory; and they're not hurting anything by being there. And some people still need them!

Chapter 4: Auditing

Fileman includes built-in auditing tools that can provide you with valuable information. Fileman "audits" a file or field by making a note in the file's audit log every time a change is made, or in some cases, every time the information is accessed. The ability to audit fields and files is a very important tool for any Security officer to have. However, the tool needs to be applied with common sense.

In a typically busy clinical environment, fields and files are being updated constantly. It would be impractical to try to audit them all. First, such large-scale auditing would cause the audit logs to grow rapidly—rapidly enough that it might even affect the speed of the overall VISTA system. Second, once all that audit data was collected, who would have time to look at it all? You'd burn yourself out trying.

Fileman's auditing features are best focused on critical and sensitive data, such as data dictionaries and patient records. If your facility is experiencing problems, it may be necessary to temporarily audit other files, but you should avoid a situation where you are auditing many files on a long-term basis. It is crucial to manage your audit logs.

The Audit Menu

Most of the options you will use to manage auditing at your site are in the Audit menu, which is found under Other Options in the main Fileman menu.

```
Select FILEMAN option: OTHER Options
Select OTHER option: AUDITing
Select AUDIT option: ?

   Choose from:
   1            LIST FIELDS BEING AUDITED
   2            TURN DATA AUDIT ON/OFF
   3            DATA AUDIT TRAIL PURGE
```

```
   4              SHOW DD AUDIT TRAIL
   5              DD AUDIT TRAIL PURGE
   6              MONITOR A USER

Select AUDIT option:
```

In VISTA, DD refers to the data dictionary. The data dictionary contains the set of attributes that defines a field. For example, the data dictionary for a NAME field might specify that it is a free text field, and contains between 5 and 30 characters.

As you can imagine, changing the data dictionary of a field could have far-reaching consequences, so in Fileman 22.2, data dictionaries are always audited. The option to turn this auditing on and off has been removed from the menu.

Let's look at each of the audit menu options in greater detail.

List Fields Being Audited

You can use this option to show which specific fields are being audited in a file, or in a range of files. You can select the files by name or by number.

```
Select AUDIT option: LIST fields being audited
Start with what file: PATIENT// OPTION
Go to what file: OPTION//
```

If you just want to look in one file, enter the same file at the "start with" and "go to" prompts. If you want to look at several files, you will need to list the file with the smaller file number first. Otherwise, Fileman gives you an error message:

```
Start with what file: PATIENT// OPTION
Go to what file: OPTION// RELIGION
The START WITH File Number must be less than the GO TO File
Number.
START WITH What File: RELIGION//
```

Notice that if you do enter your files in the wrong order, Fileman helpfully

gives you the smaller-numbered file as your new default, so you can enter your files in the correct order.

If the files you designate do contain audited fields, Fileman next gives you a Device prompt. If you do not see a Device prompt, you know that there were no audited fields in the files you selected.

Once you choose a device, Fileman gives you a report with the file number, field number, field name, type of field being audited, and the type of audit being performed. Currently, this option only lists audited fields at the file's top level; fields audited in a multiple don't appear in this listing. The Fileman development team is aware of this shortcoming, and is working to add audited multiples in a future release.

The listing below shows all fields flagged for auditing in the PATIENT file.

```
AUDITED FIELDS              JAN 30,2013@16:13              PAGE 1
FILE       NUMBER    LABEL              TYPE           AUDIT
------------------------------------------------------------------

2          .01       NAME               FREE TEXT      YES, ALWAYS
2          .02       SEX                SET            YES, ALWAYS
2          .03       DATE OF BIRTH      DATE/TIME      YES, ALWAYS
2          .05       MARITAL STATUS     POINTER        YES, ALWAYS
```

This list would continue for a while, because as you can imagine, quite a few fields in the PATIENT file are audited by default.

Turn Data Audit On/Off

As the name implies, this option allows you to set up or cancel data auditing for specific fields.

```
Select AUDIT option: TURN data audit on/off
Audit from what file: PATIENT// ORDER
Select FIELD:
```

As with the "list files" option, you can select files by name or file number. Once you have selected a file, Fileman asks you which field you would like

to work with. You can enter a question mark at this prompt to get a list of fields. As with files, you can choose fields by number or by name.

```
Select AUDIT option: TURN data audit on/off
Audit from what file: PATIENT// ORDER
Select FIELD: .01 ORDER #
Audit:
```

Next, Fileman asks you for a value to place in the audit field. There are three possibilities. Well, four. Okay, three and a half. Let's explain.

One option is **Yes, Always**. This means that Fileman will make a note in the audit log anytime somebody touches this field (in this case, the ORDER # field). It doesn't matter whether the user is editing, deleting, or even looking at the field; if the field is touched at all, Fileman makes a note.

Another option is **Edited or Deleted**. This means that Fileman will make a note in the audit log only when a user edits or deletes the value in the field. It will not make a note if the user only looks at the field.

The third option is **No**, which means that no auditing will take place.

Those are the three possibilities. But the audit field can also be left blank. If it's blank, no auditing will take place. Functionally, leaving the field blank is the same as entering No.

A blank field and a No may be functionally the same, but they look different. One is an empty value, and the other is not. You can leverage this difference to add clarity to your auditing choices.

We recommend that you reserve No for times when you are overriding audit policies or guidelines set for your facility. If you simply delete whatever value is in the field, it could appear as if the field had been deleted accidentally; somebody could "helpfully" re-set the field to its old value. If you enter a No, on the other hand, it is clear that you did it on purpose; it was not an oversight. Of course, you may need to defend your

position, but you won't have to constantly undo the efforts of people "helping."

We recommend leaving the audit field blank in all other cases where you don't want auditing. And of course, the vast majority of the fields in your Fileman installation will not be audited; the audit field is usually blank.

Data Audit Trail Purge

You can use this option to purge the audit trail—that is, to delete the log entries created from auditing data fields for a specified file. This may sound terrifying, but it needs to be done, or else your audit logs will grow too large and start slowing down the system. The logs are not purged automatically. You will need to schedule and perform purges in accordance with your organization's overall data-retention strategy.

If your organization's overall data-retention strategy is "retain all audit data all the time forever and never delete anything," then it needs to be re-written. Hanging onto audit data forever is an understandable impulse, but it is not practical and not necessary.

Before purging an audit trail, you should turn auditing off for the field(s) whose audit trail is being purged, and then turn it back on once the purge is complete.

If that is not possible, you can leave auditing on during the purge. If you do it this way, however, Fileman will be trying to audit and purge at the same time, and the audit logs recorded during the purge may end up being incomplete. If you need to leave auditing on, it is best to run the purge when there aren't many users in the system.

Purged audit records cannot be recovered!

When you select this option, Fileman asks you which file you would like to purge. The next prompt asks whether you want to purge all audit records for the file.

```
Select AUDIT option: DATA audit trail purge
Audit from what file: PATIENT// PROTOCOL
Do you want to purge all data audit records?  NO// <Enter>
```

Usually, you will not be purging all data audit records from a file, unless it's a file where you're only auditing a couple of fields. If you answer No at this prompt, Fileman next asks you how to select the records to be purged. If you type a question mark at this prompt, you will see a list of possible answers.

```
Do you want to purge all data audit records?  NO// <Enter>
Purge audit records by: INTERNAL ENTRY NUMBER// ?

 Answer with FIELD NUMBER, or LABEL
 Do you want the entire FIELD List? Y <Enter>
  Choose from:
  .001    NUMBER
  .01     INTERNAL ENTRY NUMBER
  .02     DATE/TIME RECORDED
  .03     FIELD NUMBER
  .04     USER
  .05     RECORD ADDED
  .06     ACCESSED
  1       ENTRY NAME
  1.1     FIELD NAME
  2       OLD VALUE
  2.1     OLD INTERNAL VALUE
  2.2     DATATYPE OF OLD VALUE
  2.9     PATIENT
  3       NEW VALUE
  3.1     NEW INTERNAL VALUE
  3.2     DATATYPE OF NEW VALUE
  4.1     MENU OPTION USED
  4.2     PROTOCOL or OPTION USED

Type '-' in front of numeric-valued field name to sort from high
to low.
Type '+' in front of field name to get SUBTOTALS by that field's
values.
   '#' to PAGE-FEED on each field value, '!' to get RANKING
NUMBER
   '@' to SUPPRESS SUB-HEADER,      ']' to force SAVING TEMPLATE
Type ';TXT' after free-text fields to SORT NUMBERS AS TEXT
Type [TEMPLATE NAME] in brackets to SORT BY PREVIOUS SEARCH
RESULTS
```

```
Type 'BY(0)' to define record selection and sort order
```

Selecting which fields and which records to purge is similar to the process for sorting records, as described in the *Fileman 22.2 Getting Started Manual*. In fact, you can even use a Sort template to select records to purge.

In the following example, we purge all audit data related to Flappy's protocols. (Flappy is the FLAP project mascot, pictured on the title page of this manual.)

```
Purge audit records by: INTERNAL ENTRY NUMBER// USER
Start with USER:  FIRST// FLAPPY
Go to USER: LAST// FLAPPY
  Within USER, purge audit records by: <Enter>
DEVICE: <Enter>
PURGE OF AUDIT DATA: PROTOCOL FILE   FEB 4,2013@16:10    PAGE 1
-------------------------------------------------------------------


...
0 POINTERS FIXED.

2 RECORDS PURGED.
```

As we mentioned earlier, it is not typical to purge all data audit records associated with a file. In fact, if you answer Yes at this prompt, Fileman asks if you are sure.

```
Audit from what file: PATIENT// <Enter>
Do you want to purge all data audit records? NO// YES
Are you sure? NO// YES
DELETED
```

The above dialog represents a really bad idea. Do not purge all data audit records from the PATIENT file unless you have a truly dire emergency. Purges from the PATIENT file, in particular, need to be done with care.

Show Data Dictionary Audit Trail

Beginning with Fileman 22.2 all data dictionaries are automatically audited. This function cannot be disabled. There are a couple of reasons for

this: first, changing the way files are defined can have a profound effect on the integrity of the data. And second, since the data dictionary isn't (or shouldn't be) modified that often, its audit trail isn't going to grow too large too soon.

Also new in Fileman 22.2 is this option to view the audit trail for the entire data dictionary.

```
Select AUDIT option: SHOW DD audit trail
Show Data Dictionary changes since: First// <Enter>
DEVICE: HOME//
```

If you do begin with the first change as shown above, you probably will end up with quite a few entries to look at. You may want to send this to the Browser, if it is available. Of course, if you have an idea of the time frame you want to review, you can cut the list down by entering a date at the "show changes since" prompt.

If you would like to see the data-dictionary audits for just one or two files, you can use the Inquire to File Entries or Print File Entries option, as described in the "Other Auditing Options" section of this chapter.

Data Dictionary Audit Trail Purge

This option allows you to purge the data-dictionary audit trail, much as the Data Audit Trail Purge allows you to purge the audit trail from fields. Unlike the Data Audit Trail Purge, however, this option does not need to be run regularly. Unless your site is doing something really unusual, your programmers will not be messing with the data dictionary all that much. Purging the data-dictionary audit trail won't free up much space on the system, so there's little reason to do it very often.

If you do decide to purge this audit trail, here's what it will look like:

```
Select AUDIT option: DD AUDIT trail purge
Audit from what File: PROTOCOL// LANGUAGE
Select LANGUAGE SUB-FILE: <Enter>
DO YOU WANT TO PURGE ALL DD AUDIT RECORDS? NO//
```

If you answer Yes to this final prompt, the specified data dictionary's audit trail will be purged. Remember that a purge is permanent; there is no way to undo it.

Monitoring a User

This option is new to Fileman 22.2, and was created to address the need for greater oversight of patient-privacy issues. Through this option, you can use Fileman's audit trails to review the activities of a single user. Keep in mind that you will only be able to see the user's activities in fields and files that have auditing turned on; if a user has been accessing or editing an unaudited file, you will have no way of seeing that activity.

Let's see what Flappy's been up to. (For those of you skipping around in the manual, Flappy is the FLAP project mascot, pictured on the title page.)

```
Select AUDIT Option: MONITOR A USER
Select a USER who has signed on to this system: FLAPPY
Select AUDITED File: PATIENT
Start with DATE: FIRST// <Enter>
DEVICE: HOME//
```

Once we select the user, Fileman prompts us for the file, and where we want to start. Note that if you are auditing someone whose job role includes making frequent changes to the database, starting with the FIRST date as in the example above will give you a lot of data. It's probably better to narrow your search a little, and you may want to send results to the Browser.

Here is what the results look like:

```
PATIENT RECORDS ACCESSED BY FLAPPY (DUZ=00)              Page 1
                      EARLIEST ACCESS        LATEST ACCESS
-------------------------------------------------------------------
FMPATIENT,EIGHT       DEC 6,2012@14:24:21    DEC 6,2012@14:24:35
FMPATIENT,FIFTEEN     DEC 6,2012@14:22:45    DEC 6,2012@14:23:02
FMPATIENT,NINETEEN    DEC 6,2012@14:20:49    DEC 6,2012@14:21:10
FMPATIENT,TWENTY      DEC 6,2012@14:18:41    DEC 6,2012@14:19:04
```

Other Auditing Options

Although most auditing options are accessible through the Audit submenu, some are not. In this section, we outline auditing-related features accessible through other menus.

Viewing a Data Audit Trail

You can use the Inquire to File Entries or Print File Entries options on the main Fileman menu to query the AUDIT file to obtain audit information. A general discussion of how to use these options can be found in the *Fileman 22.2 Getting Started Manual*.

When you choose AUDIT as the file to be printed (or inquired to), the next prompt asks, again, for the name of a file. In this case it is the file for which you wish to view audit data.

Let's see what happens when we use Inquire to File Entries:

```
Select option: INQUIRE TO FILE ENTRIES
Output from what File: PATIENT// AUDIT
Audit from what File: PATIENT// <Enter>
Select PATIENT AUDIT: ?
Answer with PATIENT AUDIT NUMBER, or INTERNAL ENTRY NUMBER, or
    DATE/TIME RECORDED, or USER
 Do you want the entire 103-Entry PATIENT AUDIT List?
```

Wow. If we want to use Inquire to File Entries, we need to know the exact audit number, or date and time, or record number, of the audit we want to see. We could also inquire by user, but remember that there is already a "monitor a user" option in the Audit submenu, which is probably a better way to do that kind of inquiry. If we don't have the information we want, we have to pick it out of the entire audit list. And it is probably worth noting that the dialog captured above is from a development environment, not an actual clinical environment. Odds are, your PATIENT audit file has a lot more than 103 entries.

So, unless you know exactly what you're looking for, Inquire to File Entries is probably not the option you want.

Let's try Print File Entries instead:

```
Select option: PRINT FILE ENTRIES
Output from what File: PATIENT// AUDIT
Audit from what File: PATIENT// <Enter>
Sort by: INTERNAL ENTRY NUMBER// DATE/TIME RECORDED
Start with DATE/TIME RECORDED: FIRST// 12/3/2012
Go to DATE/TIME RECORDED: LAST// 12/5/2012
  Within DATE/TIME RECORDED, Sort by: <Enter>
First Print FIELD: [CAPTIONED
Include COMPUTED fields: (N/Y/R/B): NO// <Enter>
Heading (S/C): AUDIT List// <Enter>
START at PAGE: 1// <Enter>
DEVICE: HOME//
```

This is probably closer to the kinds of audit searches you'll be doing. Using Print File Entries, we are able to sort the audits by date, and narrow our search to a three-day period. (For a general discussion of how to use the Print File Entries option, please consult the *Fileman 22.2 Getting Started Manual*). If we choose the CAPTIONED template, as shown above, our results will look something like this:

```
AUDIT List                FEB 19,2013@16:06              PAGE 1
-------------------------------------------------------------------

NUMBER: 1                                 INTERNAL ENTRY NUMBER: 1
  DATE/TIME RECORDED: DEC 4,2012@12:56:39
  FIELD NUMBER: .01                       USER: TESTMASTER,USER
  RECORD ADDED: Added Record              MENU OPTION USED: DG LOAD
PATIENT DATA

NUMBER: 2                                 INTERNAL ENTRY NUMBER: 1
  DATE/TIME RECORDED: DEC 4,2012@12:56:40
  FIELD NUMBER: .02                       USER: TESTMASTER,USER
  NEW INTERNAL VALUE: M                   DATATYPE OF NEW VALUE: RSa
  MENU OPTION USED: DG LOAD PATIENT DATA
```

The data would go on for some time, of course, but that is how it would look. Remember that you can send your results to the Browser (if it is

enabled), which may be preferable if you think you're going to get a lot of data.

Viewing a Data Dictionary Audit Trail

The Audit submenu includes an option for viewing the entire data dictionary audit trail, for all files. If you are interested in the data dictionary audit for only one or two files, however, you can use Inquire to File Entries or Print File Entries to see the information you want:

```
Select option: INQUIRE TO FILE ENTRIES
Output from what File: DD AUDIT
Audit from what File: PATIENT
Select PATIENT sub-file:
```

We didn't see the "sub-file" prompt when looking at the data audit trail, but we do see it for the data dictionary audit trail. If we specify a subfile, for example ALIAS, then Fileman will show us the data dictionary audit for just the ALIAS subfile. If we do *not* specify a subfile, the Fileman will only show us the data-dictionary audit trail for the top level of the PATIENT file; no subfiles will be included. If we want to inquire on the PATIENT file *and* all the subfiles, we will need to do those as separate queries.

As with the data audit trail, we would next be prompted for which specific audit we would like to see. If you aren't sure which audit you want to look at, then Print File Entries is probably a better option.

Setting a Data Field Audit–Modify File Attributes

We have already seen how to turn data auditing on and off by using the aptly-named Turn Data Audit On/Off option. There is another Fileman option that can be used to turn data audit on and off, although you may not have permission to use it. The option is Modify File Attributes, and it allows a user (usually a programmer) to change the data dictionary settings for a specific file and field. One of those settings is "auditing," so programmers could use this option to change the auditing value in a

particular field. For example, a programmer could change the auditing from “yes, always” to “edited or deleted.”

Presumably, a programmer would have a good reason for making such a change. Your facility should have a procedure for when and how programmers can turn data auditing off and on using this option.

Chapter 5: Other Security Issues

In the first four chapters, we discussed the major issues that arise when discussing security and patient privacy in VISTA: user access, file access, and auditing. In this chapter, we briefly describe some other issues to be aware of.

Archiving

To keep the database at a reasonable size, your facility may periodically archive some of the records. Archived records are still available in case they are needed, but they are stored in another computer system, or on disk, or printed out in physical files, so that they aren't taking up space in the system.

An archive is not a backup. A backup is a copy of the entire system, while an archive is a collection of specific records—usually older records—that will probably not be needed anytime soon. Your facility should run backups frequently, but should only archive records a few times a year.

Archiving should be done in accordance with your facility's overall data-retention strategy. And, unless your facility is exceptionally small, you as the Security officer will not be doing it yourself. You therefore need to have a general idea of how archiving works, and you should be notified whenever it takes place, but you do not need to know in detail how to archive.

In brief, archiving records means extracting a set of records, saving them elsewhere, then deleting the records from the main system. This has security implications in that the archived information needs to be stored securely. Your facility should have policies in place that describe how this is to be managed. If you do not have these policies, you need to draft some and get them approved. Once you have policies in place, part of your role as Security officer is to ensure that the policies are followed.

At present, there is no de-archive tool available for Fileman. Because of the way the archived data is structured, it is possible for a Fileman programmer to write a program for the specific data that needs to be re-inserted into the database. However, it is best to limit your archiving activity to records you're pretty sure you aren't going to need again.

Complete information on how to archive records can be found in the *Fileman 22.2 Advanced User Manual*.

Extracting

Fileman includes an extract tool, which allows super-users to copy a specific set of data from one file into another. This tool is often used to create sets of stable data for studies or analyses. For example, a super-user may want to run some kind of statistical analysis on patient data. To run her analysis, she may decide to extract copies of the records she is interested in, and save the copies to a new file. That way, she can run her analysis on the copied information, rather than trying to run it in the main database, which could tie up the system.

Often, when super-users extract data, they want to store the extracted data in a new file, created for the project they are working on. This can be handled in one of two ways.

You can give your super-users the ability to create their own files, using file numbers not currently in use by your VISTA system. You do this from the File Access Security submenu, as discussed in Chapter 3. (For a complete description of how to use the File Access Security options, please see the Kernel documentation suite.) When super-users create their own files, they automatically have access to the new files, and can proceed with their project. This method has the advantage of minimizing the time that the IT department has to spend creating and modifying files. Its disadvantages is that it's more difficult to monitor what the super-users are doing.

The alternative method is to require super-users to ask the IT department

to create files for them. When files are created in this way, the IT programmer also needs to remember to give the super-user access to the new file. The advantage for this method is that it is easier to oversee the super-users, and ensure that they are only creating files that are needed. The main disadvantage is that the IT department has to free up programmer time to respond to these requests.

Extracting is the kind of thing that makes life stressful for Security officers. On the one hand, you'd really rather not let the data go anywhere. On the other hand, your super-users are going to need to extract and examine subsets of data in order to do their jobs and keep the facility running smoothly.

Your facility should have a process for extracting data, and that process should include notifying you that the extraction has taken place. Make sure that your super-users understand and follow the process. Part of this may include making sure that the process isn't too difficult or time-consuming to follow. People are less likely to take shortcuts if they can do it the "right" way in a reasonable amount of time. Review your process, and make sure that all the steps really are necessary for security.

Complete information on the extract tool can be found in the *Fileman 22.2 Advanced User Manua*l.

Filegrams

A filegram is a data format, which can be used to transfer a record from one computer system to another. For example, a veteran who receives medical benefits through VA might visit a VA clinic in another city. That clinic could send the record of the veteran's visit to their home clinic by creating and sending a filegram (or, more likely, a small group of filegrams).

Filegrams are created by programmers using special Fileman tools. Once created, a filegram can be e-mailed to the appropriate facility. Once it has

arrived at the facility, a system manager can use other tools to incorporate the filegram into the appropriate record in that facility's VISTA system.

There are, of course, security and privacy implications to this process. When patient information is emailed, it needs to be encrypted and emailed securely. Before the system manager installs the filegram, they should check to ensure that it came from a known source, and that it has not been tampered with.

Your facility should have procedures for creating, sending, receiving, and installing filegrams. You as the Security or Privacy officer should work to ensure that these procedures are followed at all times.

Important Files

Fileman, of course, manages thousands of files. A few of these files are important enough to merit special attention:

File Name	File Number	Global
Data Dictionary	0	^DD
DD AUDIT File	0.6	^DDA
FILE File	1	^DIC
AUDIT File	1.1	^DIA
USER File	3	^DIC(3)
NEW PERSON File	200	^VA(200)

Many of these files have already been discussed. The AUDIT file and the DD AUDIT file store information on data audits and data-dictionary audits, respectively. The data dictionary itself is where files and fields are defined. Note that although almost everyone calls it the "data dictionary," its name for itself is the ATTRIBUTE file.

The FILE file lists all the files currently in the system, with the exception of

the data dictionary. The explanation for this is a little technical, but the gist of it is that by giving it a file number of 0, the Fileman developers "hid" the data dictionary from the FILE file.

Originally, the USER file was where each user's Fileman Access Codes were stored. At the time of this writing, only IHS is still using the USER file for this purpose, although it's possible an organization or facility new to VISTA could choose to do it this way as well.

Most sites keep Fileman Access Codes in the NEW PERSON file. For facilities using Kernel's File Access-by-User, that information is stored in the NEW PERSON file as well. Some sites have dispensed with the USER file altogether, and store all the relevant information in the NEW PERSON file.

Chapter 6: Best Practices

Strictly speaking, the material in this chapter is out of scope for a user manual on Fileman. However, your security interactions with Fileman will be driven by a larger set of security procedures, protocols, and regulations, so we felt it was worthwhile to discuss them here. Besides, in creating this manual, we had the assistance of a knowledgeable and experienced VA Information Security Officer, and we decided to take advantage of it.

Your Security Manual (You Do Have One, Right?)

To begin with, you should have a security manual. If your facility is part of a larger organization, such as VA or IHS, you may have a security manual from a national or central office, describing the organization's overall rules and procedures. That isn't the security manual we're talking about.

Your security manual should be specific to your facility. A centrally-produced manual would, for example, talk about physical access to the facility and how entrances can be controlled. But only your security manual could say, for example, that the Emergency entrance on Oak Street is open 24/7. Or that the west entrance is open during clinic hours, and accessible via key-card at other times. And so on.

Your security manual should include the names and contact information of the Security officer, the Privacy officer, and senior managers. It should also include any security-related procedures and protocols developed for your facility. In this manual, we have often said "you should have a process to do such-and-such." Your security manual is where you keep these processes.

Think back to when you first started your job as the Security officer at this facility. What are the five things you wish somebody had told you sooner? Make sure those things are in the security manual. Can you think of five more things? Make sure those are in the security manual, too. (Although if

one of your five things is, "I wish somebody had told me where to find the darned security manual," you'll need a different solution.)

Your Privacy Manual

Patient privacy is related to system security, but it is a separate issue. And, as more facilities convert from paper to electronic records, it is an issue of growing importance. You should view patient privacy as its own area of concern. The Security officer and the Privacy officer should be two different jobs, done by two different people (unless the facility is very small). And of course, the Privacy officer should have their own manual.

Like the security manual, the privacy manual should include the names and contact information of the Security officer, the Privacy officer, and senior managers. It should also include any privacy-related protocols and procedures developed for your facility. Privacy officers should also do the exercise where they ask themselves about five things they wish they'd known sooner, and include those in the manual.

As with the security manual, a "national" privacy manual, or one from a central authority, won't have enough detail for you to use at your facility. You need your own privacy manual that includes the specifics of your location.

Your Daily Checklist

A system manager, whose job is to keep a computer system up and running, typically has a daily routine of checks to perform. The checks tell the system manager things like how many users are logging into the system, how much memory is left, how many errors have been generated, and so on. The checks help the system manager identify potential problem areas before they blossom into massive problems.

You, too, should have a daily checklist of tasks you perform to check on the

security and privacy of your facility's VISTA system. For example, you might make it a daily practice to run an audit on the PATIENT file, or some portion of it. You might use the "monitor a user" option to see what a specific user has been up to. And of course, you want to follow up on any security- or privacy-related alerts or notifications you received through the system.

Sampling

So, when we say "run an audit on the PATIENT" file, do we mean that you have to look at every transaction that happened yesterday? Not at all. A much better idea is to look closely at a small sample of the data.

There are a few reasons for this. First, it is a much better use of your time. You have other things to do with your day; you can't spend it all looking at yesterday's audits. Second, you are most interested in problems that are widespread, and a widespread problem has a pretty good chance of showing up in a sample. And finally, you will be more alert and watchful with just a handful of audits than you would be if you tried to look at everything.

Therefore, to audit the PATIENT file, use the Print File Entries option as described in Chapter 4, but only look at a small chunk of the data (maybe an hour's worth). Scan the data, and choose two or three transactions that you find interesting, for whatever reason. Examine these transactions in more detail, until you understand them. What was changed, who made the change, and why? Don't be afraid to ask questions so you can understand the transactions.

Aside from the PATIENT file, other files that might be worth sampling include PAID, NEW PERSON, OPTION, KEY, TITLE, and DG SECURITY. The last of these, DG SECURITY, is part of VISTA's Registration package. More information on the file can be found in the Registration documentation suite.

Learning What Is Usual

A very important aspect of examining small samples of data is that you will learn what kinds of things are routine. As you become familiar with the kinds of transactions you are seeing, you will begin picking transactions that are "interesting" because they are out of the norm. You increase your chances of finding problems by becoming more familiar with what is to be expected.

Tracking Access

As we have discussed elsewhere, you should be kept apprised of any changes to users' levels of access. You should have a place where user access is tracked, possibly with your audit log, possibly in a file all its own.

It is not necessary for the Security officer to be part of the process for granting or changing access. Your facility can include you or not, as fits best with their overall operating strategy. Whether you are included or not, however, you *do* need to be notified of any changes to security access. It is especially important for you to be notified if a user's access has to be terminated.

You should revisit this list from time to time and make sure that the information you have is up to date, and that the access listed is still needed.

Tracking Local Files

Once your VISTA system has been up and running for a while, the odds are pretty good that your local programmers will begin adding files to the system. We outlined some reasons for this in the "Extracting" section of Chapter 5, but there are many reasons your facility might want to create its own files.

Your facility should have a procedure for how new files are created, and

that procedure should include making a record of who created the file, what it is to be used for, and who should have access to it. You should receive copies of all this information, so you can keep track of local files, especially those with security or privacy implications.

Appendix A: Fileman Access Codes

Fileman Access Codes have never been gathered in one manual before. And, let's be honest, they still haven't been. This is our best effort, but we're pretty sure we got some of this wrong. We have every hope that it will improve in future releases. If you have information that would help us improve this list, please contact us through OSEHRA.

^	no-one
@#	IT (programmer/system manager)
$%[]	IFCAP. (Note: IHS also uses these codes in Religion, Dental, and other packages). In IFCAP: [= read] = write % = delete $ = LAYGO
[space]	unassigned
~`	unassigned, except in IHS, where it is used for third-party billing
/\	unassigned, except in IHS, where it is used for Contract Information Services (CIS)
" '	unassigned
&*	unassigned
()	unassigned

{}	unassigned
<>	unassigned
,.	unassigned
:;	unassigned
!?	unassigned
+-	unassigned
|_	unassigned
=	unassigned
numerals	unassigned
Aa	unassigned, except in IHS, where it is used for the Financial Management System (FMS)
Bb	reserved for IHS
Cc	unassigned, except in IHS, where it is used for the Contract Health Management System (CHS)
Dd	PIMS/Admission/Discharge/Transfer/Registration/MAS.
Ee	Engineering
Ff	Dietetics/Nutrition/Food Handling
Gg	unassigned

Hh	unassigned
Ii	Imaging, but direct access is not recommended
Jj	unassigned
Kk	unassigned
Ll	Lab
Mm	unassigned, except in IHS, where it is used to denote Manager-level access
Nn	Nursing
Oo	Order Entry
Pp	Pharmacy
Qq	Quality
Rr	Radiology/Nuclear Medicine
Ss	Scheduling
Tt	TIU (Note: IHS also uses these codes for Timekeeping)
Uu	unassigned
Vv	unassigned
Ww	Women's Health
Xx	unassigned

Yy	Mental Health. Y = general y = supervisory
Zz	unassigned

Glossary

.01 Field	The most important field in a file, as identified by the file's designer. The .01 field of each record must contain information; if the value of the .01 field is deleted, the entire record is deleted.
Access code	See Access/Verify Code and Fileman Access Code. Using the phrase "access code" by itself can cause confusion.
Access/Verify Code	The codes users enter to sign in to VISTA.
Archiving	Moving inactive records into storage. This can free up space in the database for active records, and help keep the system running smoothly.
Audit	In Fileman, auditing a field means making a note in the corresponding audit log when certain transactions are carried out.
Audit access	One of the levels of access available for users of Fileman. Users with audit access can turn auditing on and off.
Audit value	The value that can be placed in the audit field of a particular data field. The possible values are: "yes, always," "edited or deleted," and "no." The field can also be left blank, which is essentially the same thing as a "no" value.
Browser	In VISTA, a tool for viewing data on your screen that has more options and better

	flexibility than a simple screen view. If it has been enabled on your system, the Browser can be selected at any Device prompt.
Data dictionary	A file that defines all the attributes of the various files in the Fileman database. Changing a file's data dictionary entry changes the way the file is set up.
Data-dictionary access	One of the levels of access available for users of Fileman. Users with data-dictionary access (usually programmers) can change how a field or file is set up.
Database key	One or more fields that uniquely identify a record in a database.
Delete access	One of the levels of access available for users of Fileman. Users with delete access can delete values in fields.
DUZ	A special variable that holds the user ID of any user in the VISTA system.
DUZ(0)	A special variable associated with DUZ. DUZ(0) stores the user's Fileman Access Codes.
Extracting	In Fileman, copying a file or part of a file and placing the contents in another file. Data extraction is often used for studies and statistical analysis.
File Access-by-User	A method for file security, managed through Kernel. File Access-by-User is an alternative to Fileman Access Codes.

Filegram	A data format designed for moving a record from one VISTA system to another.
Fileman Access Code	Fileman's native method for file security. Each file, field, and template can be given a Fileman Access Code, and users can be given corresponding codes so they can have access.
IHS	Indian Health Services, one of the primary developers of VISTA.
Kernel	Another of VISTA's infrastructure packages. Kernel handles much of VISTA's security.
Key	See Database Key and Security Key. Using the word "key" by itself can cause confusion.
LAYGO access	One of the levels of access available for users of Fileman. Short for learn-as-you-go, LAYGO access allows users to add records to files.
Line Editor	The older of VISTA's two native word-processing tools.
Local variable	In MUMPS, a variable being used by a programmer in his or her own programming space. Local variables do not affect the rest of the system.
Menuman	A Kernel module that can present different menus to different users, based on their job roles and security keys.
MUMPS	(Massachusetts General Hospital Utility Multi-Programming System) The computer language of VISTA and Fileman.

OSEHRA	(Open-Source Electronic Health Record Agent) OSEHRA is a non-profit organization tasked with coordinating the various open-source VISTA projects.
Programmer access	One of the Fileman Access Codes available. Users with this access (denoted by @) can access any field, file, or template, except those specifically designated off-limits to users (with ^).
Programmer mode	A special computer environment in which programmers can enter code and create programs. The security key XUPROGMODE is needed to enter programmer mode. Programmer mode is sometimes called direct mode.
Purge	To delete excess files in order to make room on the system. In Fileman, audit logs are not automatically purged, so manual purges must be conducted on a regular basis.
Read access	One of the levels of access available for users of Fileman. Users with read access can view information, but cannot change, add, or delete anything.
Screen Editor	The newer of VISTA's two native word-processing tools.
Security key	An electronic code that allows the user to access certain areas of the system. In Menuman, security keys are used to lock some menu options away from some users.

SQL	Structured Query Language, a popular language for relational databases. Fileman does not use SQL, but it does have tools that allow programmers to write SQL interfaces to Fileman.
Super-user	A user who manages VISTA and Fileman for their department. Different organizations have different terms for their super-users, including CAC, ADPAC, Application Coordinator, and Informaticist.
Symbol table	In MUMPS, the place where a programmer's local variables are stored. A programmer's symbol table does not affect the rest of the VISTA or Fileman system.
System file	Any file visible on the hard drive of a computer. Fileman files do not show up as system files.
Template	A Fileman feature that allows users to save queries, searches, and reports so that they can be run again. Templates can be given Fileman Access Codes.
VA	The U.S. Department of Veterans Affairs, the government agency that developed VISTA.
Write access	One of the levels of access available for users of Fileman. Users with write access can change data, but they cannot add or delete records.

Index